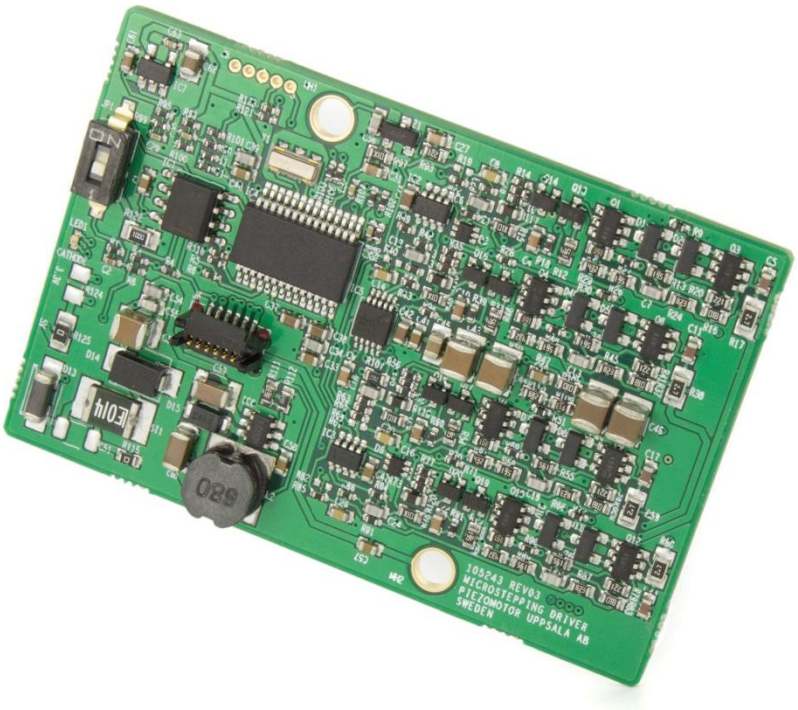
PiezoMotor®



Piezo LEGS® Controller PMD401

Technical Manual

Imprint

Revision 01
2017.05.15
Document number 150025-01

Stålgatan 14, SE-754 50 Uppsala, Sweden

Overview

This document is a guide to the technical aspects of installing and using ***Piezo LEGS® Controller PMD401***.
Detailed information about the different Piezo LEGS motors can be found on our website:
www.piezomotor.com

Content

1 - General Information

The PMD401 is a 1-axis controller/driver for use with Piezo LEGS motors from PiezoMotor. The two motor connectors provide identical output signals. Several boards may be stacked together to form multi-axis systems, and piggyback on customer's mainboard for close integration in various OEM applications. For linear motors it provides resolution down to sub-nanometer range.

PMD401 is the ideal choice for system designs where one or several Piezo LEGS motors are used. The board controls the Piezo LEGS motor by feeding waveform signals to each of the piezoelectric actuator legs. The waveforms are designed specifically to make the drive legs perform a precise walking motion. The motion of the drive legs is transferred via friction contact to a linear rod or a rotary disc.

Communication with the board is via 2-wire RS485. It can operate in closed loop using quadrature or serial encoders, or act as a Piezo LEGS amplifier for use with standard motion controllers (Servo Mode via SPI interface).

Running 8192 microsteps produces 1 full waveform cycle (wfm-step), which gives around 5 µm movement with a Piezo LEGS linear motor. Microstep resolution is better than 1 nanometer. The waveform update rate is 65 kHz and maximum cycle frequency (wfm-step rate) is 1500 Hz. Maximum cycle frequency will be set lower for motors with capacitance >0.6 µF.

The motor can be parked (powered down) to minimize position disruption at power off. Still, the position may change somewhat during the parking procedure. The Piezo LEGS motors are capacitive and do not consume any power standing still, so there is normally no reason to park while holding a position. Do however park when connecting and disconnecting motors.

PMD401 is not intended for critical applications such as life support. The user is responsible to prevent any unacceptable damage that may be caused by system malfunction.

PMD401 has been tested to comply with EN and FCC standards for radio emission.

2 - Quick Start

2.1 Connecting and Running Motor by Serial Command

1. If purchased together with a connector board, mount the PMD401 with the stacking connector and attach RS485 lines to the terminal block of the connector board. If you do not have the special connector board, read chapter 3.4, page 5, for pin-outs and selection of mating connectors.

2. If you have purchased the USB-to-RS485 serial converter cable, it should install when connecting to your computer. Otherwise, find the communication drivers on the website of the manufacturer (**http://www.ftdichip.com**). A virtual COM port is mounted once connected and installed properly.

3. Connect your Piezo LEGS motor to the 5-pole connector. If you are using a Piezo LEGS Linear Twin motor, it needs to be connected with two cables in parallel (use both 5-pole connectors).

4. If you want to be able to run the motor in closed loop, you will need to connect an encoder through the 6-pole sensor connector. The PMD401 supports quadrature encoders as well as serial SSI or BiSS encoders. Quadrature encoder is default, whereas other encoders need to be set by command at each power on.

5. Connect a 48V DC power supply (±5% and minimum 5W).

6. Download and install a data terminal software of preference. A simple freeware Windows program is *Terminal* (**https://sites.google.com/site/terminalbpp**). Connect with the PMD401 at 115200n81 and send ASCII commands.

7. The motor must be unparked (energized) in order to run. Selecting waveform with **M** command will automatically unpark the motor. Open loop run command **J** can be used to test motor function. Be careful not to let the drive rod (the shaft) escape the motor. See below examples of a few commands to start out testing (more commands and command syntax in chapter 4).

Send:	*Receive:*	*Comment:*
XM2<CR>	**XM2<CR>**	Command given to unpark motor with waveform Delta (**M2**).
XE<CR>	**XE:0<CR>**	If you have an encoder connected, you may read the position with **E**. Encoder reports position 0 counts.
XJ200,0,100<CR>	**XJ200,0,100<CR>**	Open loop run command of 200 wfm-steps forward. Speed 100 wfm-steps/second.
XJ-200,0,500<CR>	**XJ-200,0,500<CR>**	Open loop run command of 200 wfm-steps reverse. Speed 500 wfm-steps/second.
XE<CR>	**XE:63**	Encoder reports position 63 counts. Due to variation of step-length, driving the same number of steps in two directions will not take you back to the original position.

8. If you want to run closed loop, make sure to first check the Y-settings (chapter 4.2.2, page 9). Most importantly, make the correct settings of encoder type (**Y13**), target limits (**Y3** and **Y4**), encoder counting direction (**Y6**), and the SPC variable (**Y11**). Note that both **Y11** and **Y6** can be auto-configured using **Y25,1** (if encoder type is correctly set).

9. Closed loop run commands are **T**, **R** and **C**, as described in chapter 4.2.1, page 8. For example:

Send:	*Receive:*	*Comment:*
XE<CR>	**XE:63<CR>**	Encoder reports position 63 counts.
XT20<CR>	**XT20<CR>**	Enter target mode and move to position 20.
XY23<CR>	**XY23:83,1<CR>**	Target timer indicates target was reached (**1**) in 83 milliseconds (**83**).
XE<CR>	**XE:21<CR>**	Encoder reports position 21 counts, and apparently the motor has moved to target and is now actively controlling around that position. Stop range setting (**Y5**) may be altered to improve the positioning, but should be set equal to the encoder jitter.
XS<CR>	**XS<CR>**	Stop command (**S**) will stop motor and exit target mode.
XM4<CR>	**XM4<CR>**	Park motor (power down).

10. When closed loop has been tuned, for example encoder limits and acceleration, don't forget to save to flash using **Y32** command.

2.2 Setting up Multiple Axes (Stacked Boards)

1. When you have one axis up and running and you want to introduce multiple other axes on the same RS485 line – start by only having one board connected and change the axis address of that first board. Every PMD401 delivered from factory will have default address **0**. Change the address to **1** using the **Y40** command. See below the sequence of commands to send and the expected responses from the PMD401:

Send:	*Receive:*	*Comment:*
X0Y40<CR>	**X0Y40:0<CR>**	To read the current address, which according to response is indeed **0**.
X0Y40,1<CR>	**X0Y40,1<CR>**	To set address to **1**. The next communication will have to be with the newly assigned address.
X1<CR>	**X1<CR>**	As seen the board is now responding on its new address **1**. Note that address is still not stored in flash memory.
X1Y32<CR>	**X1Y32:0, Flash OK<CR>**	The save command **Y32** is used to store the axis address (and other settings) into flash memory. The new address will stick after power cycle.

2. Now when the first board has been assigned a new address, you can add a second board to the stack. Remember to remove power before mounting the second board.

3. The newly added second board will have its default address **0**. Change the board address to **2** using the same procedure as before:

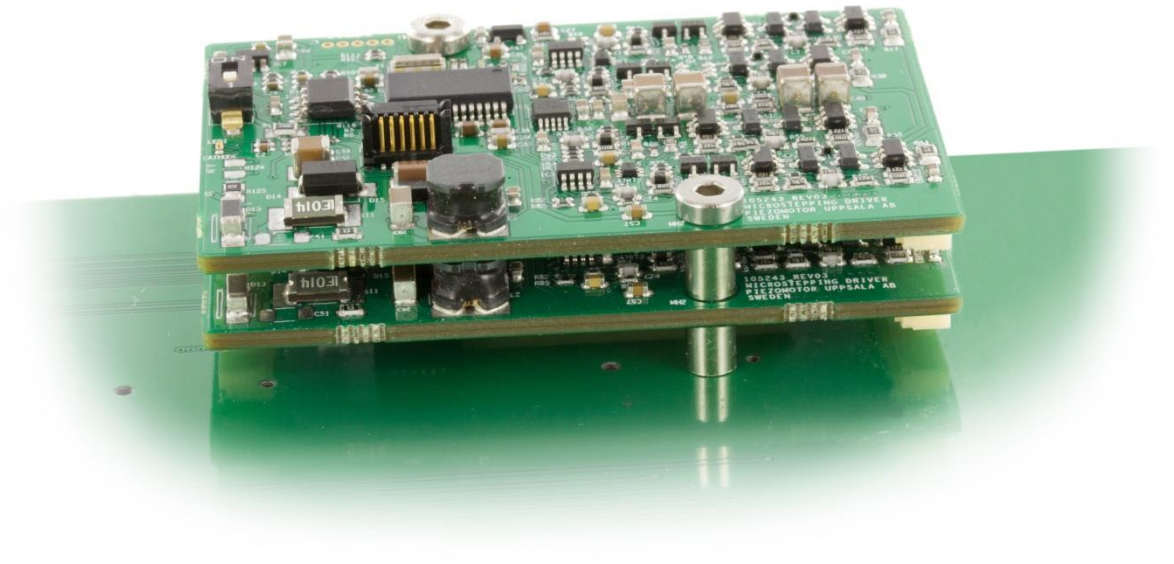
Send:	*Receive:*	*Comment:*
X0Y40,2<CR>	**X0Y40,2<CR>**	To set address to **2**. Next command will need to address axis **2**.
X2Y32<CR>	**X2Y32:0, Flash OK<CR>**	Store address into flash memory.

4. Now there are two boards connected with addresses **1** and **2**. More boards can be added with the same procedure. Always setup boards with consecutive numbering **1**, **2**, **3**,…,**n**. Address range goes from **0**…**126**, but it may be beneficial to number from axis **1** if more than one board is connected (especially when using chain command, see chapter 4.2, page 7).

3 - Installation

3.1 Mechanical

Use M3 screws (or screws of similar size), and mount board with appropriate spacer (included). Recommended spacer is 5 mm high, outer diameter Ø4 mm, and inner diameter Ø3.2 mm. See image below for example of two stacked boards mounted on a customer specific main board. Length of screws will depend on the number of boards stacked. Download CAD model from our website to see placement of mounting holes and stacking connector. The size of the board is 59 x 39 mm.



3.2 Electrical

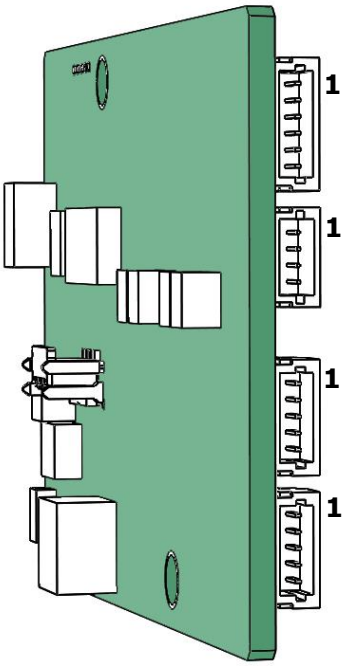
The board is powered with 48V DC power supply (±5%). Current consumption at 48V is roughly 6 mA when motor is stopped and maximum 100 mA when motor is running at maximum speed. For pinouts and connectors, see chapter 3.4 on page 5.

3.3 Host Communication

The PMD401 connects to host via 2-wire RS485. If connecting to a PC, one may use a USB-to-RS485 serial converter cable (sold separately) to open up a virtual COM port. Use data terminal software of choice and send commands in ASCII format.

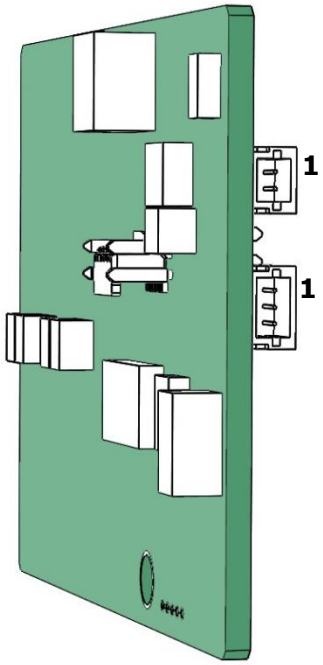
Serial Communication					
Baud Rate	**Start Bit**	**Data Bits**	**Parity**	**Stop Bits**	**Handshaking**
115200 bits/s	1	8	None	1	None

3.4 Connections



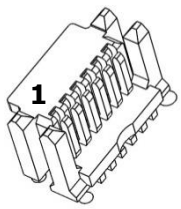
Pin	Function	Alt. Function	I/O	Notes	
Pin 1	I+	SSI Data / in3	In	Index or SSI data in; 10k pull-up	Sensor
Pin 2	GND	-	GND	Signal ground	
Pin 3	5V Out	-	Out	Max 0.2A	
Pin 4	B+	out0	In/Out	Input B or digital out; ~20k pull-up	
Pin 5	A+	SSI CK	In/Out	Input A or SSI clock; 10k pull-up	
Pin 6	(screen)	in0	In	22k pull-up and 10nF to GND	
Pin 1	GND	-	GND	Signal ground	Limit Switch
Pin 2	5V Out	-	Out	Max 0.2A	
Pin 3	Limit Reverse	in1 / out1	In/Out	External limit reverse; ~20k pull-up	
Pin 4	Limit Forward	in2 / out2	In/Out	External limit forward; ~20k pull-up	
Pin 1	GND	-	GND	Signal ground	Motor
Pin 2	Motor Phase 4	-	Out	-	
Pin 3	Motor Phase 3	-	Out	-	
Pin 4	Motor Phase 2	-	Out	-	
Pin 5	Motor Phase 1	-	Out	-	
Pin 1	GND	-	GND	Signal ground	Motor
Pin 2	Motor Phase 4	-	Out	-	
Pin 3	Motor Phase 3	-	Out	-	
Pin 4	Motor Phase 2	-	Out	-	
Pin 5	Motor Phase 1	-	Out	-	

- 3.3 to 5V signal levels.
- Signals are single ended without EMC protection – not intended for long cables. Suitable signal protection must be implemented externally, for example series resistors and TVS diodes.
- Connectors are from manufacturer JST, part numbers SM06B-SRSS-TB, SM04B-SRSS-TB, and SM05B-SRSS-TB. Mates with connectors from JST series *SH* (crimp style) or *SR* (IDC style).
- The two motor connectors are connected in parallel.



Pin	Function	I/O	Notes	
Pin 1	GND	GND	Signal ground	Power
Pin 2	+48V	In	48 VDC ±5%, 5W	
Pin 1	GND485	GND485	100Ω to GND	RS485
Pin 2	Data–	In/Out	Idle low; 23k to 2.5V	
Pin 3	Data+	In/Out	Idle high; 47k to 5V	

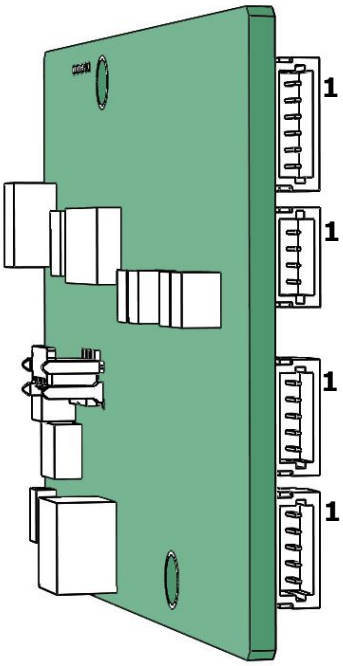
- GND485 is connected to signal ground (GND) via a 100Ω/0.5W resistor.
- There is a manual switch to control termination (AC-termination 120Ω, 1nF).
- The RS485 transceiver is robust regarding EMC and may be used with long cables. For implementing EMC surge protection, refer to Texas Instruments datasheet SN65HVD82.
- Connectors are from manufacturer JST, part numbers SM02B-SRSS-TB and SM03B-SRSS-TB. Mates with connectors from JST series *SH* (crimp style) or *SR* (IDC style).



Pin	Function	I/O	Notes	
Pin 1	Data+	In/Out	Idle high; 47k to 5V	Power RS485
Pin 2	Data–	In/Out	Idle low; 23k to 2.5V	
Pin 3	+48V	In	48 VDC ±5%, 5W	
Pin 4	+48V	In	-	
Pin 5	GND	GND	Signal GND	
Pin 6	GND	GND	-	

- This is the board stacking connectors (one connector on each side of board).
- Connectors are from manufacturer ERNI, series *MicroStac*, part number 114711. Connector mates with the same part.

3.5 Connections in Servo Mode



Pin	Function	I/O	Notes	
Pin 1	SPI_MOSI	In	16-bit signed speed data; 10k pull-up	Servo
Pin 2	GND	GND	Signal ground	
Pin 3	5V Out	Out	Max 0.2A	
Pin 4	SPI_SS	In	Slave select; Set low during transmit; ~20k pull-up	
Pin 5	SPI_SCLK	In	Serial clock, Max 500 kbps; 10k pull-up	
Pin 6	AEN	In	1=Amplifier enable; 0=Park; 22k pull-up and 10nF to GND	
Pin 1	GND	GND	Signal ground	Status
Pin 2	5V Out	Out	Max 0.2A	
Pin 3	Ready	Out	1=Overheat or parked; 0=Ready to run; ~20k pull-up	
Pin 4	Fault	Out	1=Error (other than overheat); ~20k pull-up	
Pin 1	GND	GND	Signal ground	Motor
Pin 2	Motor Phase 4	Out	-	
Pin 3	Motor Phase 3	Out	-	
Pin 4	Motor Phase 2	Out	-	
Pin 5	Motor Phase 1	Out	-	
Pin 1	GND	GND	Signal ground	Motor
Pin 2	Motor Phase 4	Out	-	
Pin 3	Motor Phase 3	Out	-	
Pin 4	Motor Phase 2	Out	-	
Pin 5	Motor Phase 1	Out	-	

- 3.3 to 5V signal levels.
- Signals are single ended without EMC protection – not intended for long cables. Suitable signal protection must be implemented externally, for example series resistors and TVS diodes.
- Connectors are from manufacturer JST, part numbers SM06B-SRSS-TB, SM04B-SRSS-TB, and SM05B-SRSS-TB. Mates with connectors from JST series *SH* (crimp style) or *SR* (IDC style).
- The two motor connectors are connected in parallel.

4 - Operation

4.1 Introduction

The PMD401 is a versatile controller, designed to work in closed loop with encoder feedback. In this chapter, the command structure and commands are described so that a programmer can build software to control one or several controllers via the serial communication interface (RS485).

4.2 Command Syntax and Controller Functionality

- The serial communication interface is 2-wire RS485 (115200n81) with addressed commands. The host must allow responses to complete before sending next command.
- The protocol is plain text (ASCII), starting with **X** and axis number **0**...**126**, followed by the command and terminated with one of the command delimiters listed below.

Command Delimiters			
Delimiter	**Hex ASCII**	**Name**	**Description**
<CR>	0D	Carriage Return	Terminates a command, response will follow from controller
<LF>	0A	Line Feed	Terminates a command, response will follow from controller
;	3B	Semicolon	Terminates a command, suppress response from controller, useful for sending several commands together without reply

- The axis number can be excluded for axis **0**, e.g. **X0E<CR>** or **XE<CR>** are both valid for checking encoder position on axis **0**.
- Set-commands will be echoed back by default.
- Axis **127** is broadcast address with no responses except for "the empty command" **X127<CR>**, in which case each axis responds with its axis number at 2 ms delay per axis, useful to detect all present units. Host should wait 300 ms before sending a new command.
- Command timeout is 300 ms. Host must disable the transmitter within 20 µs after transmission (default) and allow responses to complete before sending the next command.
- Data is in decimal format except when interpreted on a bit level in some status reports.
- Tilde (**~**) after the axis number will instead address the next higher axis number. For instance, **X0~U<CR>** will ask status from axis **1**. The response will trigger yet the next axis to give a chain response. If the axes are consecutive and a chain command is given to address **0~**, all axes will respond except axis **0**. For example, with three controllers connected (**1**, **2** and **3**) a command **X0~U<CR>** will give the response **X1~U:aaaa<CR>X2~U:bbbb<CR>X3~U:cccc<CR>** If a syntax error is detected for a chain command, then the **~** is omitted to prevent errors from consecutive axes.
- A syntax error is indicated by inserting **_??_** in the response, e.g. **X1_??_Q5**. Any syntax error in the stop command (**S**) is ignored.
- If the command was correct but could not be executed, then the command is echoed with a trailing '**!**', e.g. trying to run when motor was parked (will instead unpark).
- A target command (**T**, **R** or **C**) will enter Target Mode (closed loop) and a stop command (**S**) will end Target Mode. An open loop run command (**J** or **I**) will end Target Mode as well. The closed loop normally runs every 1 ms, except for some slow SSI encoders where the target loop iterates every 2 ms. There is no error checking for 32-bit encoder position rollover, but there are position limits (**Y3** and **Y4**) that can be used to prevent this situation in closed loop.

4.2.1 Commands

Commands	
Command	**Description**
?	***Read identification string*** Read controller type and firmware revision. Example response: **X0?:PMD401 V13**
U{StatusType}	***Read status*** Read controller status. See details in chapter 4.2.3, page 11. {StatusType}: **0**,**1**,**2**,**3**,**4**
S	***Stop*** Stop motor and exit target mode.
M{Waveform} **M**	***Waveform and parking state*** Set-command to select waveform will also unpark the motor. Read-command gives both waveform and parking state (**M:5** if parked with *Rhomb* and **M:6** if parked with *Delta*). {Waveform}: **1** Waveform *Rhomb* **2** Waveform *Delta* **4** Park (power-off)
T{TargetPos}**,**{Speed} **T**{TargetPos} **T**	***Target*** Closed loop move to new target position. Motor should not be parked. Response to read-command is **T:100** if last active target was encoder position 100. {TargetPos}: New target position [*encoder counts*] {Speed}: Stepping rate; will set **Y8** [*wfm-steps/second*; *Hz*]
R{RelActive}**,**{Speed} **R**{RelActive} **R**	***Target relative to latest active target*** Closed loop move to new position relative to latest active target. Beware of i32 overflow. Read-command reports the current target position. {RelActive}: Distance relative to latest active target [*encoder counts*] {Speed}: Stepping rate; will set **Y8** [*wfm-steps/second*; *Hz*]
C{RelCurrent}**,**{Speed} **C**{RelCurrent} **C**	***Target relative to current encoder position*** Closed loop move to new position relative current position. **C0** to hold current encoder position. Beware of i32 overflow. Read-command reports the current target position. {RelCurrent}: Distance relative to current encoder position [*encoder counts*] {Speed}: Stepping rate; will set **Y8** [*wfm-steps/second*; *Hz*]
E{Position} **E**	***Encoder position*** Read or set encoder position. Beware that setting position in target mode will move motor. {Position}: Position to set [*encoder counts*]
…**b**	***Predefine command*** Any command with a trailing **b** can be predefined in memory and later executed with **B**. For example **X1T100b<CR>** and **X2T200b<CR>** to predefine targets for axis 1 and 2.
B{Parameter} **B**	***Execute a predefined command*** To execute a command which has been predefined in memory. Useful for simultaneous execution of all axes using broadcast command: **X127B1<CR>**. Read more in chapter 4.8. {Parameter}: **0** To clear a predefined command **1** To begin execution of a predefined command.
D{outX}**,**{State} **D**	***Digital I/O*** To set output pins or read their status. See chapter 4.7 on page 15 for more details.
J{wfmStep}**,**{µStep}**,**{Speed} **J**{wfmStep}**,**{µStep} **J**{wfmStep} **J**	***Run motor (Jog)*** Open loop stepping. Read more in chapter 4.6. {wfmStep}: Number of waveform-steps [1 wfm-step = 8192 µsteps] {µStep}: Number of microsteps {Speed} : Stepping rate [*wfm-steps/second*; *Hz*]
H{Speed} **H**	***Speed for open loop*** Speed setting for open loop run commands **J** and **I**. {Speed}: Stepping rate [*wfm-steps/second*; *Hz*]
N{Mode} **N**	***Index mode*** When enabled, use open loop jogging (**I**) to search for quadrature index. Read more in ch. 4.5. {Mode}: **0** Disabled **1** Position was reset at index (read only) **2** Stop at index **4** Stop and reset position at index
I{wfmStep}**,**{µStep}**,**{Speed} **I**{wfmStep}**,**{µStep} **I**{wfmStep} **I**	***Run motor (Index Jog)*** Same as **J** command, but will only run if index mode is activated and index has not been detected. Read more in chapter 4.6.

Commands	
Command	**Description**
L{SampleDelay} **L-**{SampleDelay}	***Position logging*** Logging of 100 encoder positions. {SampleDelay}: Delay time between samples [millisecond] Example: **L5** starts logging directly after command is given (with 5 ms delay between samples), whereas **L-5** arms logging to start when receiving next command (for example a run command).
L0 **L**	***Read stored log data or status*** Response is: **L0:**{StartTime}**,**{SampleDelay}**,**{StopTime}**,**{samples}**,** {pos0}**,…,**{pos99} Response is: **L:**{StartTime}**,**{SampleDelay}**,**{StopTime}**,**{samples}**,** {pos0}
Y{#}**,**{Setting} **Y**{#}	***Settings command*** See chapter 4.2.2 below.
…**<esc>**	Escape cancels the command line (command delimiter still required). No response.
<empty>	Empty command is just echoed back (useful as a ping), e.g. **X1<CR>** will echo axis 1. Empty command to broadcast address, **X127<CR>**, will trigger each axis to respond with its axis number at 2 ms delay per axis (useful to detect all present units).

4.2.2 Settings and Miscellaneous (Y-commands)

- Set using **Y**{#}**,**{x} or alternatively **Y**{#}**=**{x}
- Read using **Y**{#} or alternatively **Y**{#}**?**
- Command **Y**{#} will most often only report the parameter whereas **Y**{#}**?** also adds a short description.
- Undocumented **Y**{#} are reserved for future use. A trailing '**!**' indicates unimplemented **Y**{#}, for example **Y99:!**

Settings and Miscellaneous		
Command	**Description**	**Notes**
Y0	***Read microstep counter*** Reads back the microstep within the waveform (8192 microsteps per cycle). Example response: **Y0:0,4096** first digit is not used, second is 0…8191.	0,U16
Y1,{Initiate}	***Initiate from flash*** {Initiate}: **2** Initiate from flash (Y3…12, Y38…40) **3** Initiate factory default values (Y3…12)	**Y32** saves to flash
Y1	***Compare current flash settings*** Responses: **Y1:0, Flash equal** Y3…12, Y38…40 **Y1:1, Flash differ** Y3…12, Y38…39 **Y1:2, Axis differ** Y40	
Y2,{xLimitEn} **Y2**	***External limit enable*** {xLimitEn}: **0** Disabled **1** Enabled, active high (pull-up) **2** Enabled, active low (pull-up)	*Default* **0** U2
Y3,{LimitA} **Y3**	***Target mode position limit A*** Target mode will stop when encoder count <A	*Default* **-10000** I32
Y4,{LimitB} **Y4**	***Target mode position limit B*** Target mode will stop when encoder count >B	*Default* **10000** I32
Y5,{StopRange} **Y5**	***Target mode stop range*** Number of encoder counts from target where it is optimal to stop the motor. This dead band should be set equal to the encoder jitter.	*Default* **1** U16
Y6,{EncDir} **Y6**	***Encoder counting direction*** {EncDir}: **0** for positive counting in forward direction **1** for negative counting in forward direction	*Default* **0** U1
Y7,{MinSpeed} **Y7**	***Target mode minimum speed*** Stepping rate [*wfm-steps per second*; *Hz*]	*Default* **1** U16
Y8,{TargetSpeed} **Y8**	***Target mode speed*** Stepping rate [*wfm-steps per second*; *Hz*] Measured capacitance limits actual maximum speed (see **U3** command)	*Default* **1500** U16

Settings and Miscellaneous

Command	Description	Notes
Y9,{SpeedRampUp} **Y9**	***Target mode speed ramp up*** Acceleration in target loop. Max 800 [*Hz/ms*].	*Default* **20** U16
Y10,{SpeedRampDown} **Y10**	***Target mode speed ramp down*** Deceleration in target loop. Max 800 [*Hz/ms*].	*Default* **20** U16
Y11,{SPC} **Y11**	***Steps Per Count in target mode*** {SPC} can be calculated ≈ 50·{EncoderResolution}, where encoder resolution is given in nanometers. You may use script **Y25,1** to auto-configure this setting. Read more in chapter 4.3, page 13.	*Default* **250** U32
Y12,{Model} **Y12**	***Target mode model*** {Model}: **0** to reach target as fast as possible **1** to avoid overshoot only in forward direction **2** to avoid overshoot only in reverse direction **3** to avoid overshoot in both directions	*Default* **0** U8
Y13,{EncType} **Y13**	***Encoder type*** {EncType}: **0** No encoder; Host may report positions (32-bit) **1** Quadrature (32-bit, max 15 MHz counting) **2** Reserved **3** Servo; SPI slave to external motion controller **4**/**5**/**6** BiSS 18/26/32 bit **8**…**30** SSI 8…30-bit; 330 kbps; Position is extended to 32-bit by lap counter. **38**…**60** SSI 8..30-bit; 130 kpbs; Position is extended to 32-bit by lap counter; Target loop runs at half speed (every 2 ms) The quadrature input signal A is used as output to SSI encoders and also in servo mode. Only encoder type 0…3 is saved to flash with Y32 whereas serial encoders will revert to 0 (none) at power on.	*Setting at delivery* **1** U8
Y14,{QuadOffset} **Y14**	***Quadrature offset*** For example: **Y14,1000** to set index position = 1000	I32
Y19	***Read analog input*** [not implemented]	U12
Y21	***Read time*** [*ms*] Free running 32 second timer. Max value 32762 [*ms*]	U15
Y22	***Read xLimit time*** [*ms*] Example: **Y22:3650,1** if stop detected at time 3650 ms	U15
Y23	***Read target timer*** [*ms*] Examples: **Y23:650,1** if target reached in 650 ms **Y23:20,0** if target not yet reached, running 20 ms	U15
Y25,{RunScript} **Y25**	***Run Script*** {RunScript}: **0** Stop **1** Script for auto-configuration of **Y6** and **Y11** Read more in chapter 4.4 on page 13.	**Y25:1,1** when executing **Y25:1,0** when finished
Y30	***Read target mode parameters*** Y2…13	csv multiple data types
Y32	***Save to flash*** Parameters Y2…13, Y38…40 are saved to flash. Will not store serial encoders for Y13, but instead reverts to no encoder at power on. Save takes ~60 ms.	**Y32:0 Flash OK** when save is done
Y38	[not used]	
Y39	[not used]	
Y40,{Address} **Y40**	***Axis address*** {Address}: **0**…**126**	*Setting at delivery* **0** U8
Y41	***Software reset*** Reboot takes about 2.5 seconds	**Y41:0, Reset** When rebooted
Y42	***Read unit serial number***	U32
Y44,{ResponseDelay} **Y44**	***Command response delay*** Delay time for RS-485 response [*µs*]	*Default at power on* **20** U8

4.2.3 Controller Status (U-command)

U0

U0:{d1}{d2}d3}{d4}

This command will give status information for many different controller functions. There is a LED on the PMD401 board which is normally OFF, but turns ON in case of voltageError or close to overheat (fanRequest). LED also makes a short blink when executing a serial command.

Bit Value	{d1}	{d2}	{d3}	{d4}
8	comError	reset	servoMode	parked
4	encError	xLimit	targetLimit	overheat
2	voltageError	script	targetMode	reverse
1	cmdError	index	targetReached	running

Underlined — Means the specific flag will stay active until reported.

comError — Error in communication has occurred; wrong baudrate, data collision, or buffer overflow.
encError — Encoder error (serial communication or reported error from serial encoder).
voltageError — Supply voltage or motor fault was detected.
cmdError — Command timeout occurred or a syntax error was detected when response was not allowed.

reset — Power-on/reset has occurred.
xLimit — Is set if the last motor movement was stopped by external limit switch.
script — Flag indicates that the SPC test or some other script is running.
index — Indicates that index signal was detected since last report.

servoMode — If servo mode is selected.
targetLimit — Is set if the position limit is reached.
targetMode — If target mode is active (closed loop).
targetReached — If target was reached; may still be regulating in closed loop.

parked — Motor is powered down.
overheat — Controller board output stage is overheated.
reverse — Last motion was in reverse direction.
running — Motor is running.

Send	Example Response	Comment	
U0	**U0:0808**	{d2}: **8** {d4}: **8**	Reset (8) has occurred; normal at power on. Motor is parked (8).
U0	**U0:0162**	{d2}: **1** {d3}: **6** {d4}: **2**	Index was detected since last report (1). Target mode (2) stopped by encoder limit (4). Last motion was in reverse direction (2).

U1

U1:{out}{in}

This command will give status of output and input pins (I/O). It also gives a fan request status when board is close to overheat and could use some cooling (on board LED will also light up). Note that bit response is hexadecimal.

Bit Value	{out}	{in}
8	fanRequest	in3
4	out2	in2
2	out1	in1
1	out0	in0

Send	Example Response	Comment	
U1	**U1:dc**	{out}: **d** {in}: **c**	**out0** high (1), **out2** high (4) and **fanRequest** (8). **in3** is high (8) and **in2** is high (4).

U2

U2:{5V}**,**{3.3V}**,**{48V}**,**{M23}**,**{Temp}

This command will give status information for board voltages. Errors previously detected are indicated with an asterisk (*****). Motor will stop on voltage error or overheat.

Parameter	Description	Nominal Value	Unit	Error Limit
{5V}	Internal 5V	**5.00**	V	±10%
{3.3V}	Internal 3.3V	**3.30**	V	±5%
{48V}	Internal 48V	**48.0**	V	±5%
{M23}	Test signal motor failure	**~23**	-	*≤14*
{Temp}	Temperature on PCB	-	°C	*≥74°C*

Send	Example Response	Comment		
U2	**U2:5.05,3.32,47.2*,23,56C**	{5V}:	**5.05**	Measured 5V level.
		{3.3V}:	**3.32**	Measured 3.3V level.
		{48V}:	**47.2***	Measured 48V level. The star indicates that an error was detected, but may no longer be present.
		{M23}:	**23**	Indicating test signal is okay.
		{Temp}:	**56C**	Indicating 56°C on PCB board.

U3

U3:{cap}**,**{freq}

This command will give status regarding measured motor capacitance and maximum allowed drive frequency.

Send	Example Response	Comment		
U3	**U3:2064nF,457Hz Delta**	{cap}:	**2064nF**	Indicating estimated motor capacitance of roughly 2064 nF.
		{freq}:	**457Hz Delta**	Calculated maximum drive frequency and information about the selected waveform (Delta or Rhomb).

U4

U4:{d1}{d2}{d3}{d4}**,**{out}{in}

This command will give status information same as U0 and U1 together.

4.3 Target Mode (Closed Loop)

A target run command (**T**, **R** or **C**) will enter Target Mode (closed loop) and a stop command (**S**) will end Target Mode. Giving an open loop run command (**J** or **I**) will also end Target Mode. The closed loop normally runs every 1 ms except for extra slow SSI encoders where the target loop iterates every 2 ms. There is no error checking for 32-bit encoder position rollover, but there are position limits (**Y3** and **Y4**) that can be used to prevent this situation in closed loop. Note that setting the encoder position while operating in Target Mode may have the effect that the motor starts to move if the position is no longer equal to target.

Target Mode requires that an encoder (position sensor) is connected. PMD401 supports quadrature encoders and serial SSI or BiSS encoders. Encoder type is selected with setting **Y13**. If an unsupported encoder is required, one may let host computer handle encoder and send position updates to PMD401 at regular intervals (for example every 10 ms) so that the controller can perform target loop.

There are a few Target Mode settings parameters (**Y3**…**Y12**) that specifically decide the closed loop behavior. Most importantly settings must be made for target limits (**Y3** and **Y4**), encoder counting direction (**Y6**), and the SPC variable (**Y11**). Note that both **Y11** and **Y6** can be auto-configured using **Y25,1** (if encoder type is correctly set).

The **Y11** setting for Steps Per Count (SPC) tells the controller how to convert a distance (encoder counts) into microsteps. The motor step length is rather approximate, so the SPC parameter does not have to be very precise for obtaining a stable target loop. The SPC has a scaled integer representation, and needs to be calculated. SPC can often be approximated to 50·[encoder resolution in nanometers]. For instance, if the encoder resolution is 5 nm, we can enter **Y11**=250. One may also run a number of open loop steps and determine the change in encoder counts per wfm-step [*counts/wfm-step*]. For example, run 16 wfm-steps (**J16**) and note the change in encoder counts. Divide by 16 to get a value *X* in unit *counts/wfm-step*. Now calculate **Y11**=65536·4/*X*.

If you do not want to calculate SPC manually, there is a script **Y25,1** which will auto-configure **Y11** described in the next section (4.4).

Target Mode will adjust to the acceleration and deceleration parameters set by **Y9** and **Y10**. For example, the ramp down deceleration parameter **Y10** defines the behavior when approaching target. Overshooting target can be prevented if deceleration is soft (a low value for **Y10**). There is also a parameter **Y12** to slow down target approach in one or both directions.

4.4 Script Command (Y25)

The only script implemented at this time is **Y25,1** which is a script to determine the relationship between encoder resolution and motor step length, and to set parameters **Y6** and **Y11** accordingly. The script will run the motor ±16 wfm-steps (roughly ±0.1 mm) and check encoder positions to do calculations. A read command will show **Y25:1,1** when executing and **Y25:1,0** when finished. An error gives a negative value (for example **Y25:1,-1**) and **Y11** is set very low.

4.5 Index Mode Command (N)

The alternatives are **N2** (to stop at index), **N4** (to stop and reset position at index), or **N0** (deactivate Index Mode). **N4** enters index mode which will reset position when the index signal comes (when ABI=111).

The read command **N** reports current index mode status and last detected index position with response **N:**{mode}**,**{position}. Mode **1** will be reported when position has been reset at index, for example **N:1,132**. A question mark after the read command (**N?**) will add a short description, e.g. **N:1,132., indexed** (meaning position has been reset at index). A dot '**.**' after the logged index position indicates that the position was logged since the last report. The index position as given by read command is logged by index signal alone, so it may be logged before position reset and can deviate somewhat from the true index position where position reset occurs (ABI=111).

The open loop command **I** can be used when searching for index. The **I** command is similar to **J** command, with the difference that it only runs when Index Mode is active, i.e. when index has not been found.

Setting quadrature offset **Y14** is a way to alter the position while keeping the index position valid. Setting the quadrature position directly (e.g. **E0**) will introduce an uncertainty to the exact index position and the Index Mode will therefore revert to zero (unindexed) and furthermore reset the offset (**Y14**=0). A position reset at index will set the position equal to **Y14**. Setting **Y14** afterwards will simply redefine the index position without the need to go back and find it.

4.6 Jog Commands (J and I)

Open loop jog command **J** can move motor in full wfm-steps, by microsteps, or by a combination of wfm-steps and microsteps, as well as set the speed. 8192 microsteps constitutes 1 wfm-step.

All input values are signed integers. A negative value for any of the parameters will give reverse movement.

J{wfmStep}**,**{µStep}**,**{Speed}	Set command:	wfm-steps + microsteps at given speed
J{wfmStep}**,**{µStep}	Set command:	wfm-steps + microsteps, previous speed
J{wfmStep}	Set command:	wfm-steps, previous speed
J	Read command:	Returns **J:1** if motor is running Returns **J:0** if motor is stopped.

Send Examples:	*Comment:*
XJ-16,4096,256<CR>	Run 16 wfm-steps and 4096 microsteps, reverse direction. 16 + 4096/8192 = 16.5 wfm-steps. Speed is 256 wfm-steps/second. Will complete in 16.5/256 seconds ≈ 64.5 ms
XJ0,128,5<CR>	Run 128 microsteps, forward direction. 0 + 128/8192 = 0.015625 wfm-steps. Speed is 5 wfm-step/second. Will complete in 0.015625/5 seconds ≈ 3.1 ms
XJ-978<CR>	Run 978 wfm-steps, reverse direction. Speed will be depending latest open loop speed (stored in **H**)

The **I** command works the same but will only run if index mode is activated and index has not been detected. Useful to avoid running if index is detected just before the run command is given.

4.7 Digital I/O Command (D)

Depending on encoder type and use of limit switch, there are some unused digital inputs and outputs available for general purpose. As can be seen in the pinout table (chapter 3.4, page 5), **out0** is not available when quadrature encoder is selected and **in3** cannot be used together with SSI encoders or quadrature index.

Note that **out0** and **in0** use different pins , whereas **in1**/**out1** and **in2**/**out2** share the same pins as external limit switch inputs. Outputs **out1** and **out2** must be set high (open drain) to read the inputs. Setting **out1** or **out2** low when limit switch function is enabled will produce a pulse and return to high level (input). External limit switch signals should then be open drain or connected via series resistors, for example 1.2kΩ.

D

D:{out2}{out1}{out0}**,**{in3}{in2}{in1}{in0}

Read command will give status of input and output pins. Pin high indicated by **1**, and pin low indicated by **0**.

Send	Example Response	Comment
D	**D:110,1100**	{out2}: **1** Indicating out2 is high {out1}: **1** Indicating out1 is high {out0}: **0** Indicating out0 is low {in3}: **1** Indicating in3 is high {in2}: **1** Indicating in2 is high {in1}: **0** Indicating in1 is low {in0}: **0** Indicating in0 is low Since {in1} is **0** while {out1} is **1**, this indicates that an external signal drives the pin low.

Set command **D**{outX}**,**{State} will set output pins, where {outX} is **2**/**1**/**0** for **out2**/**out1**/**out0**, and {State} is **1** for pin high and **0** for pin low.

Example Send	Comment
D0,1	out0 is set high (1).

4.8 Predefining a Command (B)

Any command given that ends with a trailing **b** will be stored in the **B** command and can later be executed with **XB1**. For instance, **X1T100b<CR>** and **X2T200b<CR>** will store commands **T100** for axis 1 and **T200** for axis 2. Both commands can then be executed simultaneously by sending command **B1** to the broadcast address, **X127B1<CR>**. Sending **X0~B1<CR>** will also execute the predefined commands almost simultaneously (chain command for consecutively numbered axes).

…**b**	Stores a command, for example **X1T200b<CR>** will store command **T100** for axis 1.
B0	Clears a stored command.
B1	Begins execution of stored command. The response from executed command is not returned. If however the executed command generates an alert, then the alert indicator '**!**' is returned, i.e. **XB1!**
B	Reports the stored command string. Example response: **B:T100b**

5 - Servo Mode

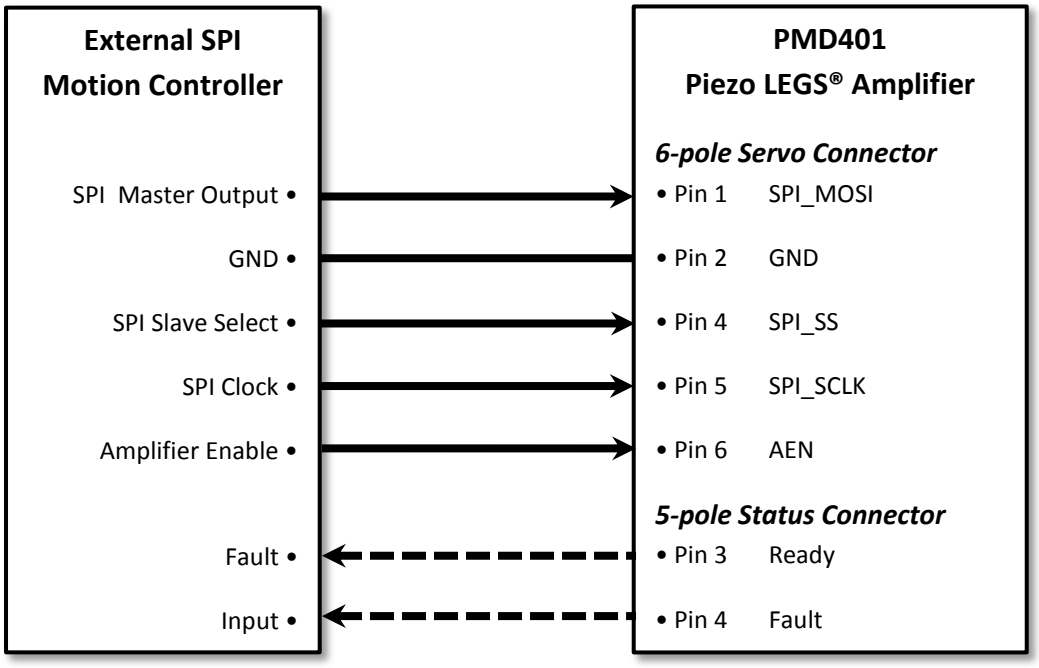
5.1 Slave to an External Motion Controller

PMD401 can act as a Piezo LEGS amplifier for an external servo motion controller with SPI interface. Note that suitable signal protection must be implemented externally, for example series resistors and TVS diodes.

Servo mode is enabled by connecting with the PMD401 by serial RS485 interface, setting **Y13** parameter to **3**, and saving the setting to flash memory by **Y32** command. Note that the serial commands to run/stop the motor will disable servo mode until reboot or **Y13** parameter is set to **3** again.

See chapter 3.5 on page 6 for connections in Servo Mode. A low level on **AEN** input signal will power down (park) the motor. It normally takes 250 ms to power down and at least 80 ms to power up the motor (longer with larger motors). The **Ready** signal will be low when PMD401 is ready to run. A high **Ready** signal indicates amplifier overheat or motor powered down (parked). A high level on **Fault** indicates a critical error, e.g. motor or voltage failure (not overheat). Fault condition is also indicated by LED. A fault will stay for at least 250 ms.

5.2 SPI Interface



Amplifier enable signal from the motion controller connects to **AEN** input and unparks motor on high level. The **SPI_MOSI** controls the motor speed. The motor will stop when input data is =0. The PMD401 expects regular speed updates and will stop the motor if no SPI data has been received for about 10 ms. The 16-bit signed SPI data gives linear speed control (stepping rate). Note that motor step length is not constant and a position sensor (encoder) is necessary for feedback to the motion controller.

SPI_SCLK idle state is high and PMD401 samples data after positive edge (max 500 kbps). Serial command **E** can be used to report the latest received SPI data (useful for debugging).

Maximum motor stepping rate is 1500 Hz. However, a capacitance check is done at the time of unpark and may select a lower speed limit for motors with capacitance >0.6 µF.

5.3 Waveform Selection

The default waveform at power ON is *Delta*. The other alternative is waveform *Rhomb*. Waveform must be selected with serial command.

The *Delta* waveform gives much higher positioning accuracy and is normally preferred. The *Rhomb* waveform may give longer step length for light loads, i.e. faster max speed.

5.4 External Motion Controller PID Settings

The external motion controller should normally be setup for a brush-type DC servo amplifier. However, the motor command will alter the motor stepping rate (frequency) rather than the torque, so the PID settings will differ from a normal servo. It is recommended to have D=0 and I=0 and only use P. If there is a “velocity feed forward”, this may improve motion control.

A servo controller normally requires feedback from a position sensor (encoder). Using a motion controller without an encoder often results in continuous run at maximum speed unless a limit has been set on PID output. Limiting the PID output can be convenient when testing for example the encoder direction. Take away the PID limit when the system is functioning, and set the desired motion speed, acceleration, and deceleration.

6 - Safety Instructions

Note!

The PMD401 driver is a high-end product intended for use with PiezoMotor's Piezo LEGS product line. In order to get best performance and reliability it is important the driver unit is handled according to the instructions given in this manual and other delivery documents.

Caution!

The piezoceramic elements in a Piezo LEGS motor act as capacitors and can sometimes hold substantial electrical charge.

- Make sure motors are discharged through suitable discharge resistors.

Caution!

Incorrect installation using improper mounting materials or methods can cause damage to the PMD401 driver unit.

Caution!

Depending on its use the PMD401 unit can get very hot.

- The PMD401 should be installed in a clean and dry environment with access to proper ventilation. On installation, ensure that air can flow around the board without obstruction.

Caution!

Electrostatic discharges can cause irreparable damage to the electronics.

- Note and follow the ESD protective measures

Caution!

Incorrect connection of motor leads may cause irreparable damage to both motor and PMD401.

- Connect in accordance with the specified pin assignment.

7 - Maintenance

7.1 General Maintenance Instructions

The PMD401 does not require any regular maintenance. Follow given safety instructions.

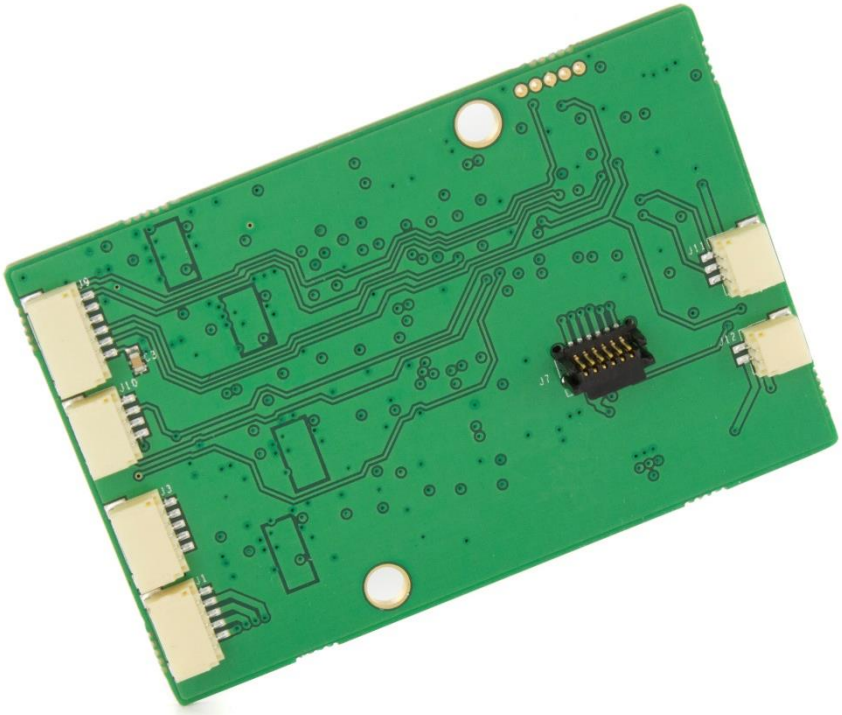
7.2 Trouble Shooting

If problem arise, the status command (**U**) should provide useful information. Check wirings to motor and encoder. Try running in open loop to see if you can get movement. Regarding closed loop operation, make sure the Target Mode settings are correct. If you get readings from the encoder you can see if you have the correct resolution by taking a number of open loop wfm-steps and checking how many encoder counts you have traveled. A wfm-step is typically 5±3 µm for a standard Piezo LEGS linear motor.

7.3 Firmware Updates

When new features are added, the user can update the firmware over the serial interface using bootloader software. PiezoMotor will make updates available on **www.piezomotor.com**.

When firmware is updated, you should also look for the latest revision of this manual to learn about the updates, or read the change log document.



PiezoMotor Uppsala AB
Stålgatan 14
SE-754 50 Uppsala, Sweden

Telephone: +46 18 489 5000
Mail: info@piezomotor.se
Web: www.piezomotor.com