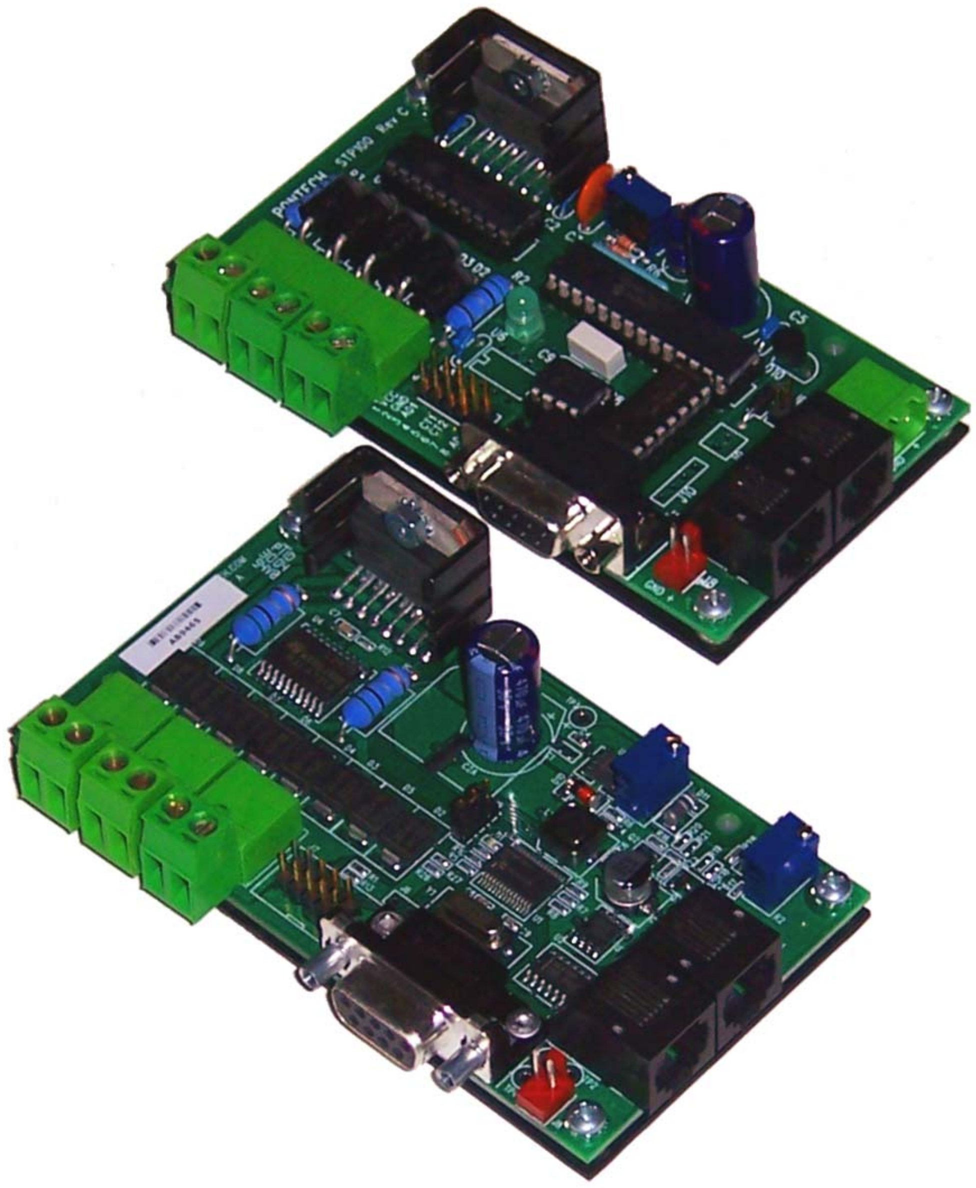
STP100/STP101

Stepper Motor Controller



October 16, 2004

www.pontech.com

Preface

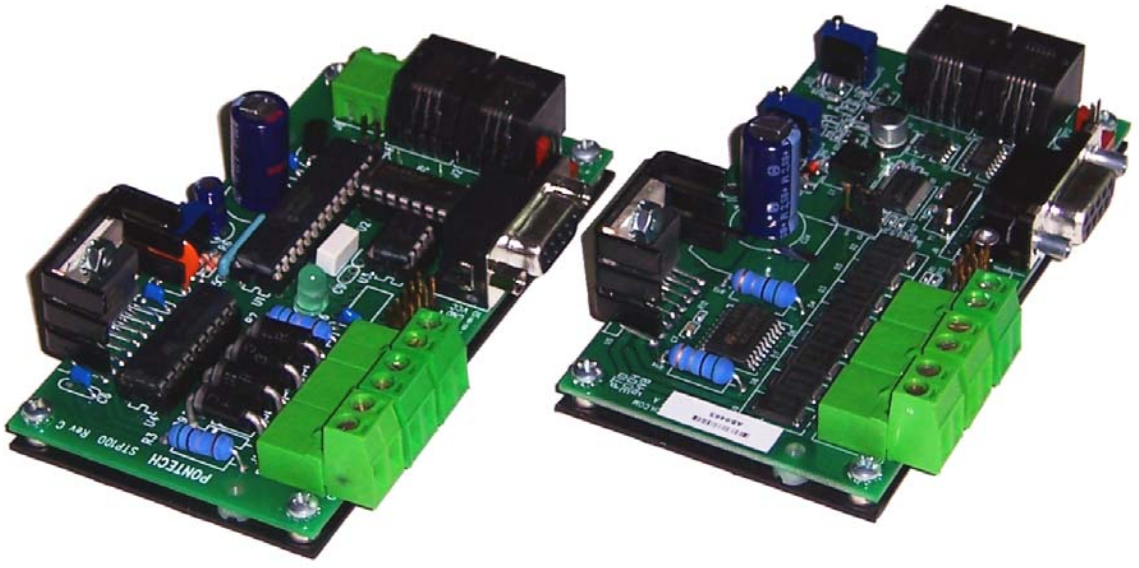
This documentation reflects the technical specifications of the STP100 & STP101 Firmware V2.3, hereby referred as STP10x. Differences between models will be noted to a specific model, ie. STP100 or STP101.

Table Of Contents

Preface **2**
Table Of Contents **3**
Introduction to the STP10x **5**
Stepper Motor Basics **6**
Bipolar, Unipolar 6
Motor Construction 6
Bipolar Operation 7
Operating a Unipolar as Bipolar 7
Board Connections **8**
STP100 Board Connections: 8
STP101 Board Connections: 9
Connecting a host computer to the STP10x board(s) 10
RS-232(Serial) 10
RS-485 Network 10
Connecting a Stepper Motor to the STP10x 11
Connecting power to the STP10x 12
Powering the STP100 12
Unregulated Logic power 12
Motor power 12
Single Supply Operation 12
Powering the STP101 12
Digital I/O 13
Board Operation **14**
The motor current limit 14
Current Limiting on the STP100 14
Current Limiting on the STP101 14
Single mode Adjustment 15
Double mode Adjustment 15
Command Structure 16
Returned Values 16
Command Examples 17
Command List **18**
Board Control Commands 18
BDn Select Board 18
PSn Pin Set 18
PCn Pin Clear 18
RPn Read Pin 19
ADn Read a voltage on the A/D port 19
WRm n Write to Ram 20
RRn Read from Ram 20
WEm n Write to EEPROM 20
REn Read from EEPROM 20
EEPROM Memory Map: 20

WSS Write System Settings to EEPROM 21
Motion Commands 21
MIn Move Immediately to absolute position....................21
MCn Move Cued to absolute position 21
IIn Increment Immediately relative to destination position 21
OIn Offset Increment Immediately relative to current position....................21
ICn Increment Cued relative to current position 22
CU Cue to move or increment....................22
RC Read Current motor position....................22
RD Read Destination motor position 22
RT Read delTa motor position....................22
RX Read Direction Sign....................23
HMn Re-home stepper 23
H0 Halt with deceleration(H-Zero) 23
HI Halt Immediately, do not decelerate....................23
H+ Advance CW indefinitely 23
H- Advance CCW indefinitely....................23
SO Stepper Off when not moving....................23
SP Stepper Powered when not moving 23
SH Step Half 24
SF Step Full....................24
SW Step Wave 24
TCn Test if pin Clear, on clear execute HI 25
TSn Test if pin Set, on set execute HI....................25
TTn Set Test Mode 25
Acceleration and Speed Commands....................26
SDn Step Delay....................26
RSD Read Step Delay....................26
SAn Set Acceleration/Deceleration Factor 27
RSA Read Acceleration/Deceleration Factor....................27
SMn Set Minimum Step Delay Factor....................27
RSM Read Minimum Step Delay Factor 27
Commands Summary 29
Warranty and Copyrights....................30
Warranty 30
Copyright and Trademarks 30
Disclaimer of Liability 30

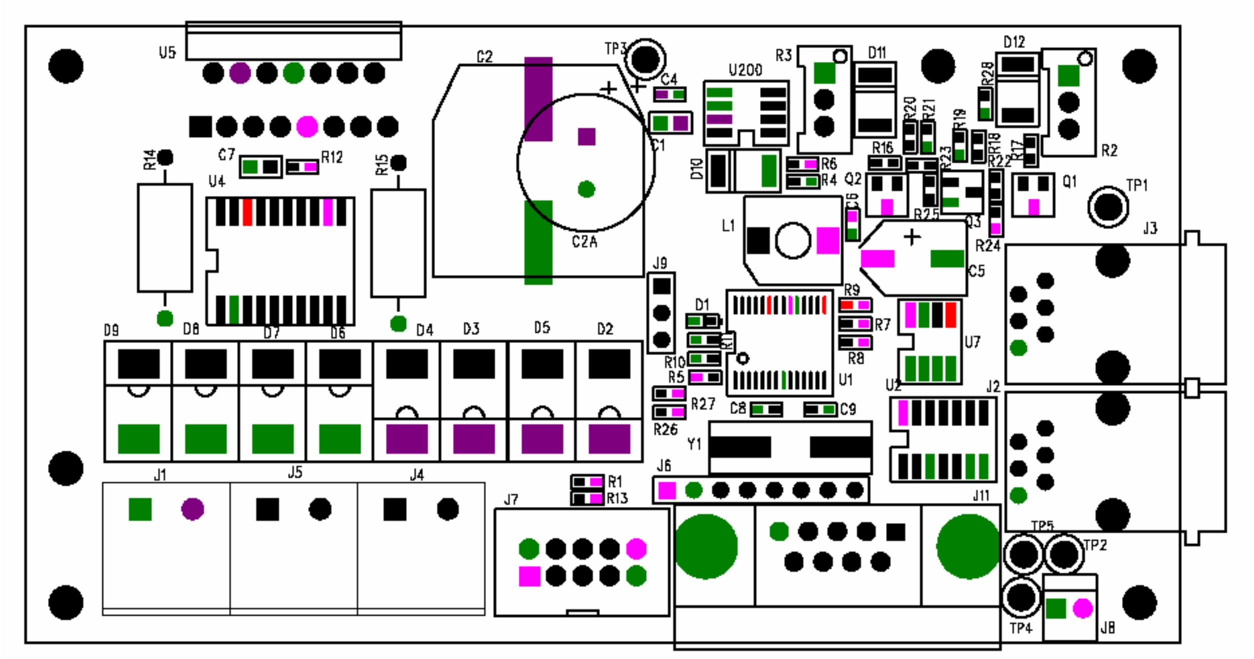
Introduction to the STP10x



The STP10x line of motor controllers is the perfect choice for small to medium stepper motor applications where step precession to 32 bits is required. The STP10x is divided into four functional blocks (Embedded Microprocessor, Stepper Logic, Motor Driver Circuit, and Serial Line Drivers). The embedded microprocessor gives the STP10x the ability to keep track of an absolute motor position. Among other things, the position is a signed 32-bit number. Other features of the STP10x that are microprocessor controlled include:

Motor acceleration and deceleration
Human readable command set
Multi board command processing
Cued commands
STP10x Board status queries (Motor Position, Pin Conditions)

The stepping logic for the STP10x is supplied by a SGS Thompson Stepper chip with built in current limiting. The driver chip supplies up to two Amps per stepper motor winding. The STP10x has the ability to drive both RS-232A and RS-485 serial lines.



The STP101, depicted above, has all of the functionality of the STP100 with the added functionality of single supply operation and both running and holding current limit potentiometers.

Stepper Motor Basics

Stepper motors are the motors for the digital age. The rotor of a stepper motor moves in discrete increments or steps (hence the name stepper motor). Because of there discrete nature, the control of a stepper motor lends its self nicely to microprocessor technology. The distance of a stepper motor step is given in degrees per step. Some typical values are shown below:

1.8 degrees / step (200 steps / revolution)
7.5 degrees / step (48 steps / revolution)

Bipolar, Unipolar

There are two common winding configurations for stepper motors: bipolar and unipolar. Bipolar motors have two independent windings while unipolar motors have two independent center tapped windings that tend to get connected together at the center tap while in operation.

Motor Construction

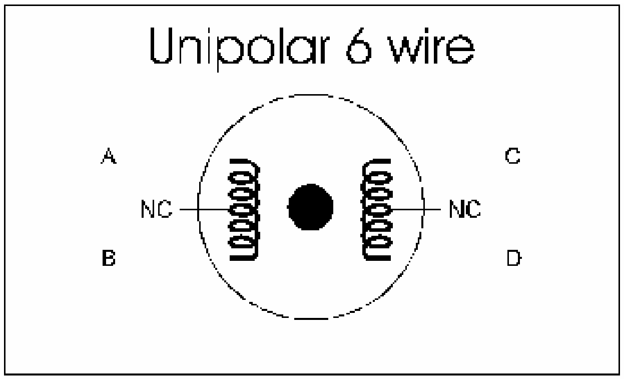
Stepper motors consists of two major components: the stator and the rotor. The stator is the stationary portion of the motor and the rotor is the portion of the motor that rotates. The stator consists of electromagnets that are alternately powered to attract permanent magnets on the rotor. When a magnet is pulled toward an electromagnet, motion is induced into the rotor.

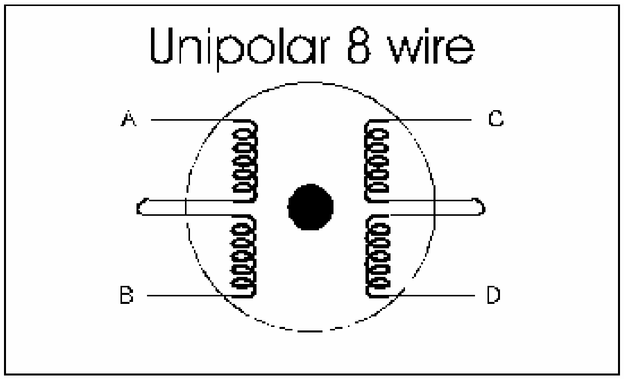
Bipolar Operation

Motion is accomplished by alternately powering one winding then the next and then reversing the power supply and repeating the process. There are several different control schemes that are commonly used for bipolar motors: full-stepping, half-stepping, wave stepping and micro-stepping.

Operating a Unipolar as Bipolar

Here are some illustrations on configuring unipolar motors to operate as bipolar motors. Often it is difficult to determine winding configurations by looking at the motor. Refer to the manufacture specifications for exact information. In a six wire unipolar motor, the center taps are not used and in an eight wire unipolar motor, the windings are tie together in pairs. Because the center taps are not used or the two windings are tie together in series, the voltage rating of the motor is now doubled.

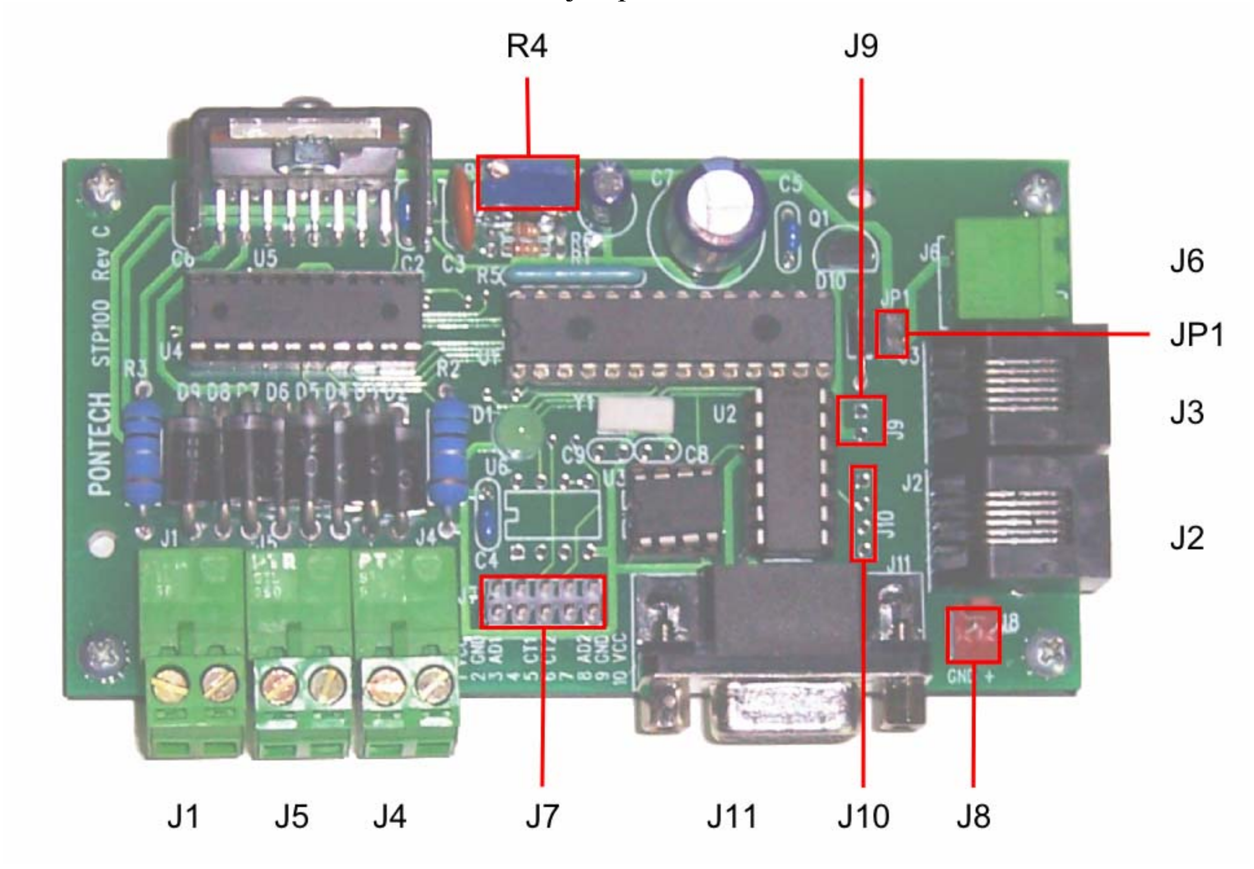




Board Connections

STP100 Board Connections:

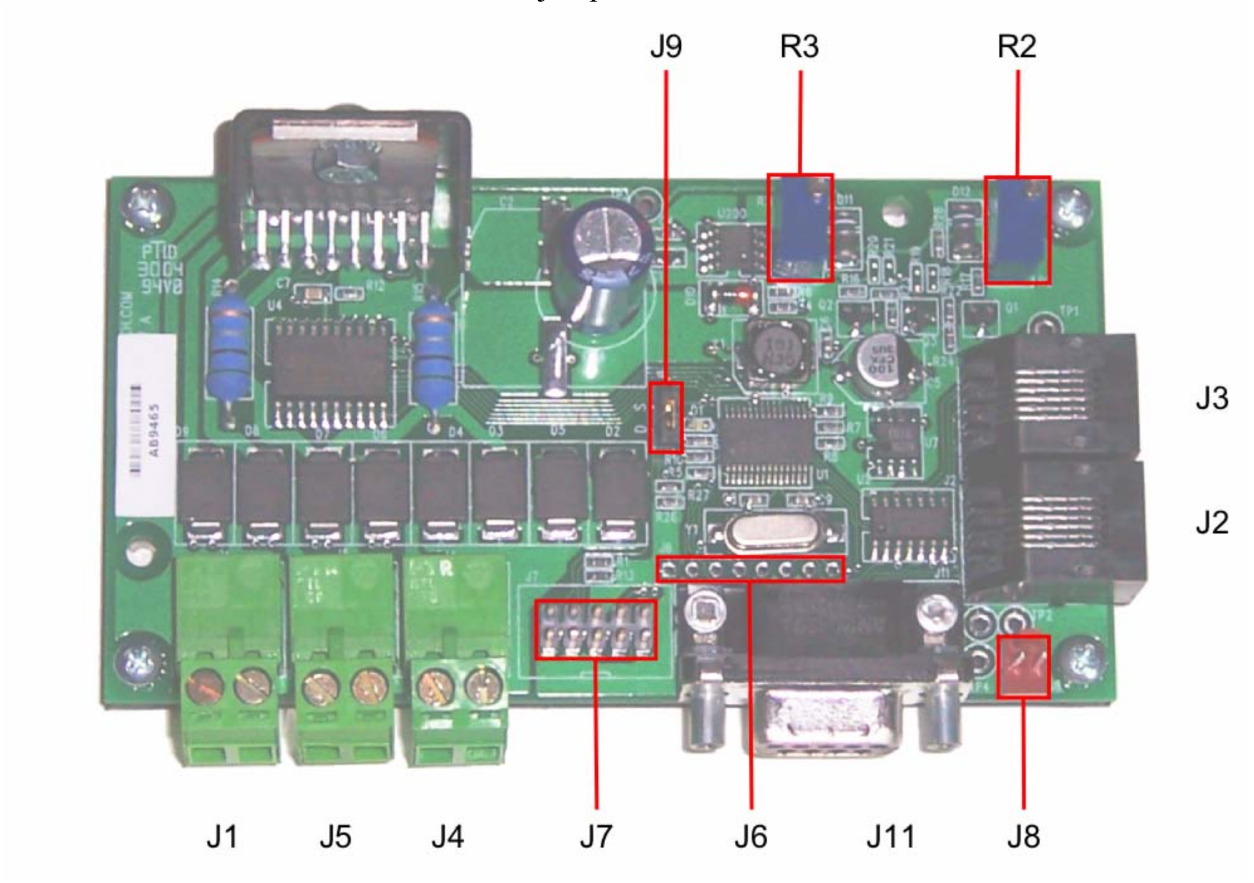
The STP100 has eleven connectors and one jumper.



Name	Description
J1	Stepper motor power **(GND VIN)**
J2	RS-485 multi drop network connection
J3	RS-485 multi drop network connection
J4	Stepper motor winding connection point **(B A)**
J5	Stepper motor winding connection point **(D C)**
J6	Unregulated Logic power
J7	Digital I/O
J8	Unregulated Logic power
J9	Unregulated Logic power **(**Unpopulated**)**
J10	RS-485 multi drop network connection **(**Unpopulated**)**
J11	RS-232 9-pin DCE connector
JP1	Tie Logic power to motor power when jumper present
R4	Current Limit Adjust

STP101 Board Connections:

The STP101 has nine connectors and one jumper.



Name	Description
J1	Board Power **(GND VIN)**
J2	RS-485 multi drop network connection
J3	RS-485 multi drop network connection
J4	Stepper motor winding connection point **(B A)**
J5	Stepper motor winding connection point **(D C)**
J6	JTAG Header **(**Unpopulated**)**
J7	Digital I/O
J8	Regulated Output **(GND +5V)**
J9	Current Limit Mode Jumper **(S**ingle/**D**ouble**)**
J11	RS-232 9-pin DCE connector
R2	Current Limit Adjust **(S:** No Operation, **D:** Moving Current**)**
R3	Current Limit Adjust **(S:** Holding/Moving, **D:** Holding Current**)**

Connecting a host computer to the STP10x board(s)

(Connecters: J2, J3, J10, J11)
The STP10x parses input from a serial interface and responds with requested information. This allows for easy interface to a host computer with an off the shelf terminal program or custom program.

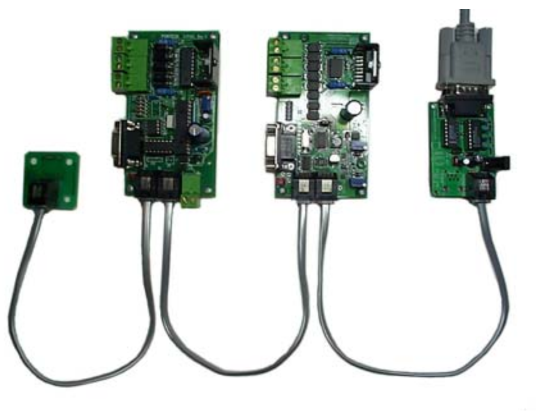
RS-232(Serial)

(Connecter: J11)
The STP10x is equipped with a 9-pin female RS-232 interface connector configured as DCE (data communications equipment). When using only one board, the STP10x will connect directly to a host computer serial port through J11 without the need of a RS-232 to RS-485 converter (Pontech part # PP232-485f).
Note: When using the J11 connector, only one device can be connected to the host computer serial port. The STP10x cannot drive additional STP10x boards when J11 is used.

The J11 connection is best suited for testing individual boards and applications requiring one stepper motor. When using RS-232, the cable length is limited to a range of 20 to 30 feet. Cables longer than these lengths tend to pickup noise that can effect the operation of the STP10x.

RS-485 Network

(Connecters: J2, J3, J10)
The STP10x is also equipped with two 4-pin RJ11 RS-485 interface connectors that will allow for multiple boards to be chained together in a network. The RS-485 connectors allow for full duplex multi drop communication with a host computer when used in conjunction with the PP232-485f.

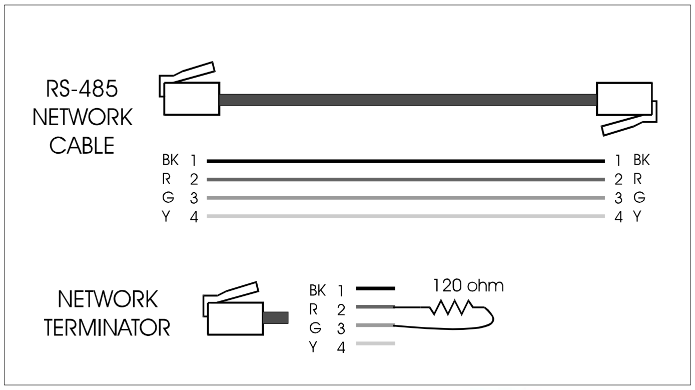


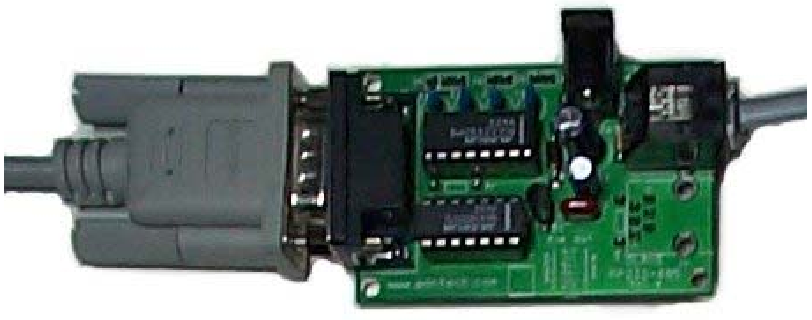
WARNING: The cables for the RJ11 are straight through type. Telephone cables have different pin-outs and will not work.

When using the RS-485 interface, up to 31 devices can be connected to a host computer using only one serial port. A PP232-485f is needed to convert the RS-485 to RS232. Up to 255 devices can be connected to the host computer with additional RS-485 signal relays or repeaters.

The RS-485 network setup is suited for applications requiring multiple stepper motors connected to one host computer and longer cable ranges of up to 2000 feet. When using an RS-485 network, termination is required at the end of the network chain. This is done by tying a 120 OHM resistor to the two middle pins of the RJ11.

The RJ11 cable used is connected in a straight through manner. A standard phone cable cannot be used. The cable used should be wired as such…




PP232-485f

Connecting a Stepper Motor to the STP10x

(Connectors: J4, J5)

Most bipolar motors come with four wires. They are usually labeled A, B, C and D, where A, B belong to one winding and C, D belong to the other. Connect the wires accordingly to J4 and J5 on the STP100. J4 and J5 may be interchanged if the phase sequence causes the motor to move counter-clockwise when the intended movement is clockwise. If A and B are flipped or if C and D are flipped, the stepper may not operate correctly and cause the motor to oscillate when issued a move command.

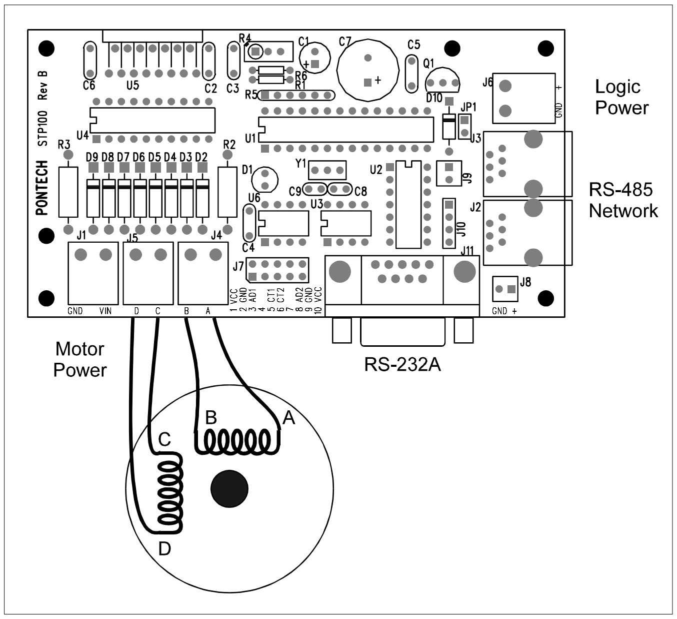
Connecting power to the STP10x

Powering the STP100

(Connectors: J1, J6, J8, J9, JP1)

The STP100 requires two power supplies to operate. One supply is used for the control logic and the other for the motor power.

The STP100 logic and motor power connectors share a common ground. The ground connections are labeled GND on the STP100 printed circuit board.



Unregulated Logic power

(Connectors: J6, J8, J9)

An unregulated power supply operating in the range of 7-15VDC is to be connected to one of the supplied connection points (J6, J8 or J9). Only one connection of the three connectors is required

Motor power

(Connector: J1)

A regulated power supply operating in the range of 5-46VDC is required to supply power to the motor.

Single Supply Operation

(Connector: JP1)

A single supply can be used for both logic and motor power when the motor power supply operates in a voltage range of 7VDC to 15VDC. To use a single supply, connect a jumper to JP1 and apply power through J1.

WARNING: If separate power supplies are used for motor and logic power for the STP100, be sure that JP1 is not installed.

Powering the STP101

(Connector: J1)

The STP101 is powered similarly to the STP100 configured for Single Supply Operation. With the difference that connection point J1 of the STP101 can accept a voltage range of 8VDC to 46VDC. The voltage you apply to J1 should be the voltage required by the stepper motor you intend to use.

Note: Connector J1 will power both the stepper motor and control logic.

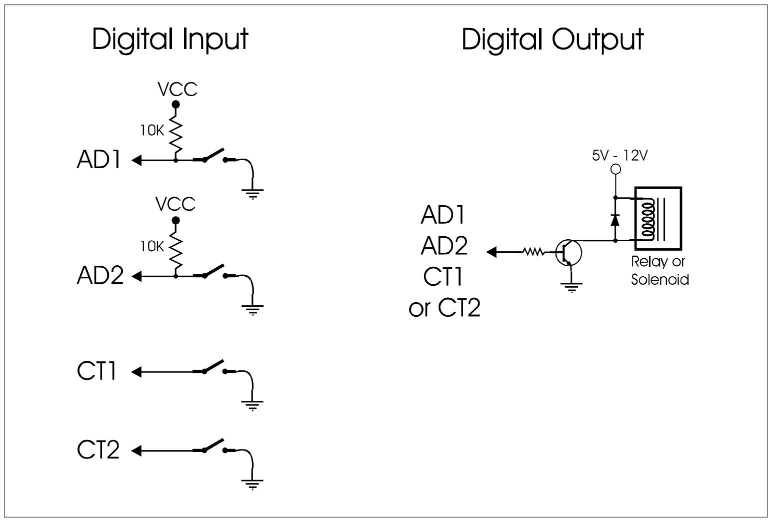
Digital I/O

(Connector: J7)

The STP10x has the ability to sense four digital I/O lines. When configured for input they can be used for motor shut off and home detection applications. These pins can also be used as digital output lines. In addition, two of these pins can be used as 8-bit analog to digital converter inputs.

Pins CT1 and CT2 can be connected directly to a single pole single throw micro switch without the addition of any other circuitry. If connecting AD1 or AD2 to a single pole single throw micro switch, an external pull-up resistor as shown below is required.

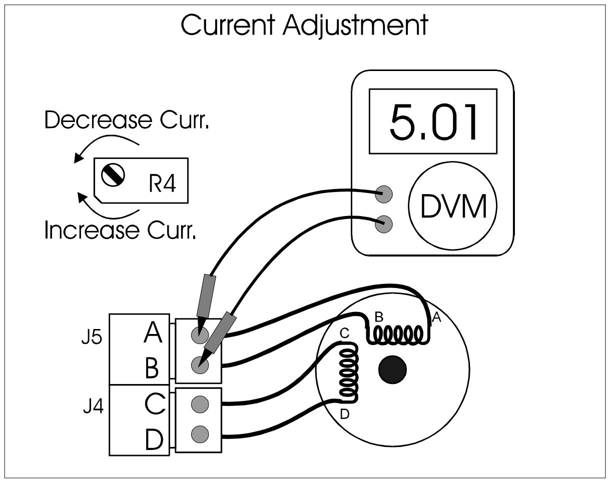
Pin	Description
1	VCC Regulated logic power 5VDC
2	GND Board ground (common to logic and motor)
3	AD1 Analog input, Digital input/output (STP101 has 100KΩ pull-down to GND)
4	STP100: No Connect / STP101 200Ω resistor to VCC (+5V)
5	CT1 Digital input (pulled to 5VDC)
6	CT2 Digital input (pulled to 5VDC)
7	STP100: No Connect / STP101 200Ω resistor to VCC (+5V)
8	AD2 Analog input, Digital input/output (STP101 has 100KΩ pull-down to GND)
9	GND Board ground (common to logic and motor)
10	VCC Regulated logic power 5VDC



Board Operation

The motor current limit

To prevent potential damage of your motor. The voltage across each phase should not be above the rating of the stepper motor being used when the motor is not moving. To test the voltage across the winding, connect the stepper motor, serial port and power the STP10x. Using an RMS voltmeter set for VDC, place the test leads to terminals A and B of the stepper motor connectors, (shown in the diagram below). Current limiting should be adjusted to the maximum current required to prevent the motor from slipping during normal operation. Read the sections below for details on current limiting on your STP100/STP101. If excessive torque is required and the motor can handle greater then two amps per winding, the STP10x may not be capable of driving your motor.



Note: If current limiting is not adjusted properly the stepper motor may operate at excessively high temperature. This can lead to heat damage of the permanent magnets, there by reducing the motor's torque.

Current Limiting on the STP100

Make sure the board is at full phase by issuing a **SF** command. Then issue a **SP** command to power the stepper motor. Connect a RMS voltmeter set for VDC, and place test leads to terminals A and B of the stepper motor connectors. Adjust the pot labeled R4 until the voltage is close to the voltage rating of the stepper motor.

Current Limiting on the STP101

The STP101 has two Current Limiting Modes settable by jumper J9, **D**ouble and **S**ingle. In **D**ouble mode, Pot R3 controls the Holding current, (while the motor is halted), while Pot R2 controls the Moving current, (while the motor is in motion). In **S**ingle mode, Pot R2 is disabled and Pot R3 controls both the Moving and Holding currents.

Single mode Adjustment

To adjust in **S**ingle mode, make sure the board is at full phase by issuing a **SF** command. Then issue a **SP** command to power the stepper motor. Connect a RMS voltmeter set for VDC, and place test leads to terminals A and B of the stepper motor connectors. Adjust the pot labeled R3 until the voltage is close to the voltage rating of the stepper motor.

Double mode Adjustment

To adjust in **D**ouble mode, make sure the board is at full phase by issuing a **SF** command.

Adjust the moving current limit by issuing a **SP** command to power the stepper motor windings, simulating a moving state. Connect a RMS voltmeter set for VDC, and place test leads to terminals A and B of the stepper motor connectors. Adjust the pot labeled R2 until the voltage is close to the voltage rating of the stepper motor.

Adjust the holding current limit by issuing a **SO** command to turn off the stepper motor windings, simulating a halted state. Connect a RMS voltmeter set for VDC, and place test leads to terminals A and B of the stepper motor connectors. Adjust the pot labeled R3 until the desired holding voltage is chosen.

Command Structure

The STP10x processes one ASCII string at a time. The string must be capitalized and terminated with a carriage return or <ASCII 13>. The board will not process the command string until it encounters the <ASCII 13>. The default communication setting is 9600 baud, N81, with echo off. It will not send anything back unless it is a read command. Any terminal programs such as Procomm, Qmodem or HyperTerminal may be used to test the feature of the STP10x board.

When a board is selected, an empty command (carriage return only string) will cause the board to return its board ID number and a greater-than prompt (i.e. 001>). The board will stay selected until power is removed or another board select command is received. For example, the following commands select board one, set the step delays to 800, and move the stepper to absolute position 100. The <enter> is ASCII value 13.

```
BD1 <enter>
SD800 <enter>
MI100 <enter>
```

Multiple commands may be issued on one command string. Make sure the total length per line including the <ASCII 13> is less than or equal to 20 characters long.

```
BD1MI100 <enter>
```

Comma or space can separate multiple commands.

```
BD1,MI100 <enter>
     or
BD1 MI100 <enter>
```

Any command or parameter value not in the range of the command will be ignored. The host computer talking to the board should delay a minimum of three milliseconds between each command string sent to give the STP10x time to process the command.

Returned Values

All commands that invoke the STP10x to return a value are formatted as such:

```
<Number String> <ASCII 13> <ASCII 10>.
```

Command Examples

Homing the stepper motor

The commands below will select board number one, set a flag to stop the stepper if pin 5 of the J7 connector is pulled low, then move the stepper motor until pin 5 reads 0 Volts.

```
    BD1 <enter>
    TC5 <enter>
    H- <enter>
```

Then use the **RT** command the see if the motor is still moving. When it hit the limit switch, the motor will stop and the **RT** command will return “0”. Then home the position to 0 by sending:

```
    HM0 <enter>
```

Issue commands while motor is moving

The speed or a new position may be adjusted on the fly, as the motor is moving.

```
    II200 <enter>
    II300 <enter>
```

The commands above will move the motor a total of 500 steps from its initial position.

Multi-Axis Control

When multiple board are connected on the same line. Each board must have a unique ID number (the ID number of the board can be changed by using the **WE0** <ID#> command).

```
    BD1 MI4000 <enter>
    BD2 MI-2600 <enter>
    BD3 MI600 <enter>
```

The commands above move the stepper motor on board 1 to position 4000, then board 2 to position –2600, and finally board 3 to position 600. If the application requires that all three motors start moving at the same time, then use cue moves (see **CU** command).

Command List

Board Control Commands

BDn Select Board

Before the board will accept any commands, it must first be enabled. To enable the board, you must send the **BD** command followed by the board ID number. The default ID number of the board is 1. So simply send the following to enable the board:

```
BD1 <enter>
```

The board ID number can be user redefined by using the **WE** command. This allows multiple boards of different ID number to be connected on the same network.

You can enable the board in two other ways: You can pre-enable the board at power-up by changing the default-settings. (see **WSS** command); or you can enable the board by sending an ID number 0, as such:

```
BD0 <enter>
```

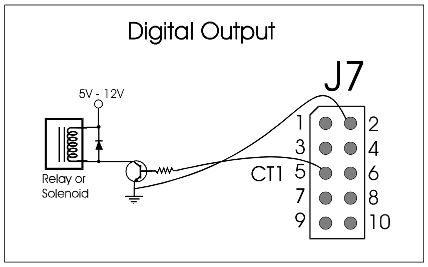
This will override the ID number checking and any boards connected to the network will be enabled regardless of the ID number of the board. No board will respond with a prompt to a Carriage Return <ASCII 13> if a **BD0** command has been sent. This prevents collision on the RS-485 line.

PSn Pin Set
PCn Pin Clear

These commands allow you to use the pins 3, 5, 6, and 8 on the 10-pin header (J7) as digital output by setting or clearing individual bits of this port.

```
PS8 <enter>
```

Pin 8 of the 10-pin header will be set high (5 Volts).
The pins can drive and sink up to 25 mA, a driver circuit such as the one below may be required to drive anything that uses more current such as a relay or a solenoid.

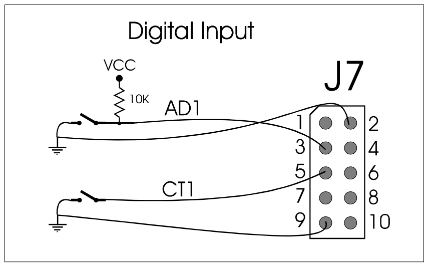


RPn Read Pin

This command will read the current state of pins 3, 5, 6, and 8 on the J7 connector. If n is a number 3,5,6 or 8, the board will return 0 or 1 corresponding to the specified pin. If n is nothing, the board will return a 4 bit number with 3, 5, 6 and 8 being bits 0, 1, 2 and 3 respectively.

```
    RP5 <enter>
```

If nothing is connected to pin 5 of J7, the board will return a value "1"
followed by <ASCII 13> and <ASCII 10>.



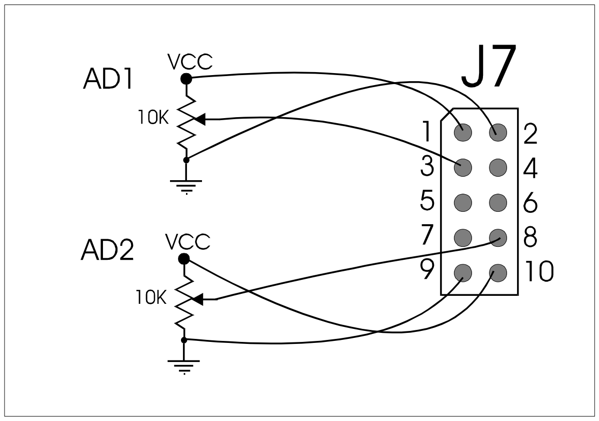
ADn Read a voltage on the A/D port

The 10-pin header has two 8-bit ADC input channels (pins 3 and 8) that read an analog voltage. The n is number 1 or 2 that tells what channel on the 10-pin header (J7) to request. When the board receive this command, it will read the voltage on the pin specified and return a value between 0 to 255 that represent a voltage between 0 to 5 Volts.

Voltage = value / 255 * 5 Volts

```
    AD1 <enter>
```

If wires were connected as the figure below and the pot was in the middle position, the board will return a value close to 128 followed by <ASCII 13> and <ASCII 10>, which is about 2.5 Volts.



WRm n Write to Ram
RRn Read from Ram

These commands allow you to modify and read the contents of the internal register or RAM of the processor. The internal RAM is volatile memory storage, so when power is removed the content will be erased.

WEm n Write to EEPROM
REn Read from EEPROM

This command allows you to modify and read the content of the external EEPROM connected to the processor. The EEPROM is non-volatile memory storage, so any information written to it will stay even when power is removed.

```
    WE0 2 <enter>
```
Change the board ID number to #2

```
    WE10 0 <enter>
```
Board won't be enable on power up

```
    RE9 <enter>
```
Read the baud read setting of the board.

EEPROM Memory Map:

Address (m)	Usage	Default (n)	Note
0	Board ID #	1	
1	Reserved	1	Should not be 255
2	Step Delay Lo-byte	208	Default = 2000
3	Step Delay Hi-byte	7	
4	Minimum Step Factor Lo-byte	0	Default = 0
5	Minimum Step Factor Hi-byte	0	
6	Step Acceleration Factor	10	
7	Step Mode	0	0 – Half 1-Full other-Wave
8	Step Powered on hold	0	0 – off 1 – on
9	Baud Rate	50 (9600 baud)	24 (19200 baud) 100 (4800 baud) 200(2400 baud)
10	Pre Enable Flag	1	1-Yes 0-No
11	Reserved		
12	Test Mode	0	0=Mode 0, 1=Mode 1

When the board is power-up, it first reads the external EEPROM to get its Board ID number, baud rate, initial stepper settings, and some other initial flags. These initial settings can be changed by using the **WE** or **WSS** command.

WSS Write System Settings to EEPROM

Use this command to save the current values for step delay, minimum step delay factor, acceleration factor, step modes (half, full, wave), and step power to EEPROM. When the board is re-powered, it will be initialize to these values.

Motion Commands

MIn Move Immediately to absolute position

This command will move the stepper to an absolute position. The range of the position is between –2147483648 and 2147483647 steps. If the stepper is in motion the **`MI`** command will not wait for the motor to stop before adjusting the destination position. If the STP10x is set for half-stepping, **`MI`** moves the stepper in half steps.

MCn Move Cued to absolute position

This command is similar to the **`MI`** command, except the move will not execute until the **`CU`** (cue) command is sent to the STP10x. This is useful for synchronizing multiple boards connected to the same network.

IIn Increment Immediately relative to destination position

This command will move the stepper motor relative to its destination position,

```
MI1000 <enter>
II500 <enter>
II-1000 <enter>
```

The steppers destination will first be set to position 1000, then to position 1500 (1000+500), and then finally to position 500 (1500 - 1000). If the stepper is in motion the **`II`** command will not wait for the motor to stop before adjusting the destination position. If the STP10x is set for half-stepping, **`II`** moves the stepper in half steps.

OIn Offset Increment Immediately relative to current position

This command will move the stepper motor relative to its current position.

```
MI1000 <enter>
OI500 <enter>
OI-1000 <enter>
```

Compared to the **`II`** command, the **`OI`** command functions similarly with the exception that the final destination may vary. Depending on the current position the command was parsed which is affected by the speed of the stepping and the amount of time that passes before a next **`OI`** command is parsed. If the STP10x is set for half-stepping, **`OI`** moves the stepper in half steps.

ICn Increment Cued relative to current position

This command is similar to the **II** command, except the move will not execute until the **CU** (cue) command is sent to the STP10x. This is useful for synchronizing multiple boards connected to the same network.

CU Cue to move or increment

This command cues a move or increment move that was previously set by a **MC** or **IC** command. This is normally used with the **BD**0 command to enable all board before issuing a cue.
The following example will tell three boards connect in a network to move all at the same time. Board 1 to position 200, board 2 to position 400, and board 3 increment by 1000. They will not execute the command until the **BD**0 **CU** string is received. The last command just makes sure only one board stay selected.

```
BD1 MC200 <enter>
BD2 MC400 <enter>
BD3 IC1000 <enter>
BD0 CU <enter>
BD1 <enter>
```

RC Read Current motor position
RD Read Destination motor position

Read the current or destination motor position.

```
RC <enter>
```

The board will return a value between –2147483648 and 2147483647 followed by <ASCII 13> and <ASCII 10>.

RT Read delTa motor position

(Destination - Current)

Read the difference between destination and current motor position. A value of 0 means the motor is not moving. A value greater than 0 means the motor is moving clockwise, and a value less than 0 means the motor is moving counter-clockwise.
This command is the same as doing a **RD** command and a **RC** command and then taking the result of **RD** and subtract the result of **RC, i.e.** (**RD** - **RC**).

```
RT <enter>
```

The board will return a value between –2147483648 and 2147483647 followed by <ASCII 13> and <ASCII 10>.

Note: (**RD** – **RC**) may be greater than the signed 32 bit number that is returned by **RT**. If this is the case, the value will overflow and the result returned will be smaller then what it should be.

RX Read Direction Sign

This command returns +, - or 0 <ASCII 48> depending on the motion state of the motor.

+ => Motor moving in positive direction (destination > current position)
- => Motor moving in negative direction (destination < current position)
0 => Motor not moving (destination = current position)

HMn Re-home stepper

(current position = destination position = n)
Set a new home position.

```
    HM0 <enter>
```

Issuing **HM0** will Re-home the stepper to zero. A **MI200** command after this will move the motor clockwise 200 steps.

H0 Halt with deceleration(H-Zero)
HI Halt Immediately, do not decelerate
H+ Advance CW indefinitely
H- Advance CCW indefinitely

The **H**x commands allow for spinning and halting the stepper continuously with regard for it's absolute position. Once a **H+** or **H-** command is invoked, the destination position is set to the maximum or minimum respectfully. If the limit is reached the current position will be set to the opposite limit and the motor will continue in its current direction. All **H**x commands except **HI** will accelerate and decelerate the motor appropriately.

SO Stepper Off when not moving
SP Stepper Powered when not moving

These two commands are used to set the state of the stepper coils when the stepper is not moving. If the application requires no holding torque, such as a lead screw, the **SO** command should be executed in order to save wear on the stepper and power consumption. If holding torque is required on power up the **SP** command will keep the winding always powered even when the motor is not moving.
Note: The **WSS** command may be used to save the current state to EEPROM so the state is restored on next power up.

SH Step Half
SF Step Full
SW Step Wave

These three commands are used to set the STP10x into full step, half step or wave step mode. When any one of these commands is executed the STP10x may move up to a step from its current position.

Half Step Mode

Phase	A	B	C	D
1	-	+		
2	-	+	-	+
3			-	+
4	+	-	-	+
5	+	-		
6	+	-	+	-
7			+	-
8	-	+	+	-

Full Step Mode

Phase	A	B	C	D
1	-	+	-	+
2	+	-	-	+
3	+	-	+	-
4	-	+	+	-

Wave Step Mode

Phase	A	B	C	D
1	-	+		
2			-	+
3	+	-		
4			+	-

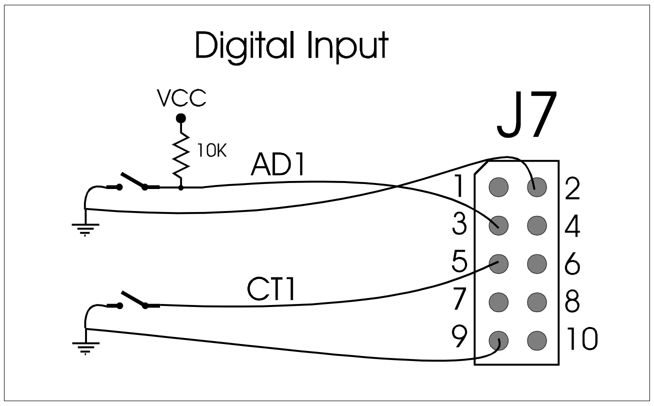
Note: The `WSS` command may be used to save the current state to EEPROM so the state is restored on next power up.

TCn Test if pin Clear, on clear execute HI
TSn Test if pin Set, on set execute HI

These two commands can be used to sense limits by using a micro switch tied to ground and to pins 3, 5, 6, or 8 of the STP10x. When a condition is met, a **`HI`** command is executed to stop the stepper, the absolute position can then be set to zero with the home command (**`HM`**).

This command can be started before or after a move command. It will stay in its checking state until the condition is met. Then will perform a **`HI`** command and stop the motor. The **`HM0`** command may then be required to home the position to zero if so desired.

Pins 6 and 8 have built-in pull-up resistors to VCC. If pins 3 and 8(5?) are used as home limit sensors, external pull-up resistors must be used.



TTn Set Test Mode

Mode 0:
`TC` and **`TS`** commands behave as described above.

Mode 1:
`TC` and **`TS`** commands are direction specific.
AD1 and CT1 now will halt the motor moving in the – (**`CCW`**) direction.
AD2 and CT2 now will halt the motor moving in the + (**`CW`**) direction.
`TC` and **`TS`** commands do not need to be re-issued between bumps.

Acceleration and Speed Commands

The STP10x keep tracks of three internal values that are used in determining how the motor should accelerate, decelerate and the nominal operation speed. These values are StepDelay, MinimumStepDelayFactor and AccelerationFactor. The delay used between steps while the motor is accelerating up to its operating speed is calculated as such:

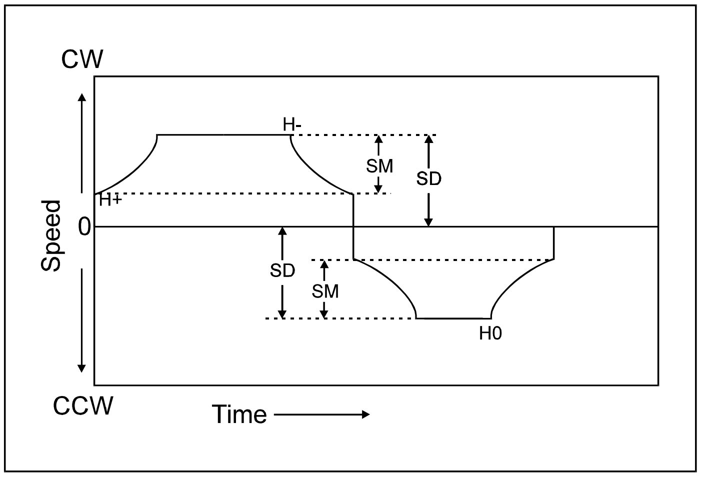
CurrentStepDelay =
StepDelay + MinimumStepDelayFactor –
(AccelerationFactor * steps)

Where: CurrentStepDelay >= StepDelay

The board starts with CurrentStepDelay equal to StepDelay plus MinimumStepDelayFactor. Then CurrentStepDelay is subtracted each step by a factor of AccelerationFactor until CurrentStepDelay is greater than or equal to StepDelay.

To disable the acceleration/deceleration feature use MinimumStepDelayFactor = 0 (`SM0` command).

To use the acceleration/deceleration feature, start with a MinimumStepDelayFactor = 2000 (`SM2000`) and AccelerationFactor = 10 (`SA10`) and then trim the values to suite the application.



SDn Step Delay
RSD Read Step Delay

This command is used to set delay between steps (half, full or wave). The smaller the value, the faster the motor moves. The fixed time per Step Delay is **2.04uS**. The **`RSD`** command is used to read the Step Delay value.

You can calculate the value of SD for a desired RPM using the formula:
1*(duration of 1 second)* **/ sps***(steps we need in that duration)* ***/ 2.04uS****(constant time per step delay)*

A complete example how to calculate the SD value for a desired 300 rpm:
First convert rpm to revolutions per second.
300 rpm / 60 = 5 rps

Then take the number of steps to complete a revolution for a given stepping motor & mode(Full stepping/Half stepping)
For a 1.8 degree stepper motor: **1 rev. = 200 Full steps = 400 Half steps**
// I am going to be using Full stepping for the rest of this example

Find the total steps required a second to achieve the desired rps, by multiplying the rps by the steps per revolution.
5 rps * 200 Full steps = 1000 sps(steps per second)

Using the fixed **2.04uS** per Step Delay, we can complete the formula to find the value of SD for the desired rpm/rps.
1 / 1000 sps / 2.04uS = 490.196 = SD

The **WSS** command may be used to save the current state to EEPROM so the state is restored on next power up.

SAn Set Acceleration/Deceleration Factor
RSA Read Acceleration/Deceleration Factor

The **SA** command is used to set the acceleration and deceleration factor of the stepper when large loads are being moved. The **RSA** command is used to read the Acceleration Factor value.

The **WSS** command may be used to save the current state to EEPROM so the state is restored on next power up.

SMn Set Minimum Step Delay Factor
RSM Read Minimum Step Delay Factor

The Minimum step delay factor is used for setting the initial step delay at the beginning of a move. If the Minimum Step Delay Factor is set to zero then no acceleration or deceleration will occur. The **RSM** command is used to read the Minimum Step Delay Factor value.

The **WSS** command may be used to save the current state to EEPROM so the state is restored on next power up.

> **SM**1000 *<enter>*
> If **SD** was previously set to 2000 and **SA** was previously set to 10,
> The CurrentStepDelay is initially assigned to 3000 (**SD** + **SM**). As the motor moves during each step, CurrentStepDelay is subtracted by 10 until CurrentStepDelay is greater or equal to **SD**. The

CurrentStepDelay stays at 2000 until it’s time to decelerate. When decelerating CurrentStepDelay is added by 10 until it reaches the Destination Position.

Commands Summary

Command	Parameter	Description
BDn	N = 0 to 255	Board Select
MIn	N = -2147483648 to 2147483647	Move Immediately to an absolute location
MCn	N = -2147483648 to 2147483647	Move on Cue to an absolute location (must be followed by CU command)
IIn	N = -2147483648 to 2147483647	Move Immediately relative to current position
ICn	N = -2147483648 to 2147483647	Move on Cue relative to current position cue (must be followed by CU command)
CU	None	Cue a move
PCn	N = 3, 5, 6, or 8	Pin clear on 10-pin header
PSn	N = 3, 5, 6, or 8	Pin set on 10-pin header
RPn	N = 3, 5, 6, 8 or None	Read Pin on 10-pin header (If n is not specified, a four bit value is returned representing all pins)
ADn	N = 1 or 2	Get A/D value, the board will return a value between "0" to "255" which represents a voltage between 0 to 5 Volts.

WRm n	M = 0 to 255
	N = 0 to 255
		Write to internal RAM, m is the memory location n is the value to write

WEm n	M = 0 to 8190
	N = 0 to 255
		Write to external EEPROM, m is the memory location n is the value to write

WSS	None	Write System Settings to EEPROM (current value of SD, SM, SA, SH, SF, SW, SP, SP are stored to EEPROM)
RRm	M = 0 to 255	Read the contain of internal RAM, m is the memory location to read
REm	M = 0 to 8190	Read the contain of external EEPROM, m is the memory location to read
RC	None	Read Current Position
RD	None	Read Destination Position
RT	None	Read delTa Position (DestinationPosition - CurrentPosition)
RX	None	Read Direction Sign (Returns +, - or 0)
HMn	n = -2147483648 to 2147483647	Set new home position, current position = destination position = n
HI	None	Halt Immediately, set destination position to current position and do not decelerate
H0 (H - Zero)	None	Halt, Set speed to 0 and decelerate
H+	None	Move Clockwise forever
H-	None	Move Counter-Clockwise forever
SP	None	Stepper Always powered
SO	None	Stepper Off when not moving (Remove power from stepper, no holding torque)
SH	None	Set Half Step mode
SF	None	Set Full Step mode
SW	None	Set Wave Step mode
TCn	n = 3, 5, 6, or 8	Test if pin clear, on clear execute H0
TSn	n = 3, 5, 6, or 8	Test if pin set, on set execute H0
TTn	n = 0 or 1	Set Test Mode
RSA	None	Read Step Acceleration/Deceleration Factor
RSD	None	Read Step Delay
RSM	None	Read Minimum Step Delay Factor
SAn	n = 1 - 255	Acceleration/Deceleration Factor
SDn	n = 6 to 65535	Step Delay (n x 2.04µs)
SMn	n = 0 to (65535 - SD)	Minimum Step Delay Factor (Start Step Delay = StepDelay + MinimumStepDelayFactor) if = 0 then no acceleration

Warranty and Copyrights

Warranty

ProLinear/PONTECH, Inc. warrants its products against defects in materials and workmanship for a period of 90 days.

If you discover a defect, ProLinear/PONTECH, Inc. will, at its option, repair, replace, or refund the purchase price. Simply return the product with a description of the problem and a copy of your invoice (if you do not have your invoice, please include your name and telephone number).

The warranty does no apply if product has been damaged by accident, abuse, or misuse.

Copyright and Trademarks

Disclaimer of Liability

ProLinear/PONTECH, Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of property, and any costs or recovering, reprogramming, or reproducing and data stored in or used with ProLinear/PONTECH, Inc.