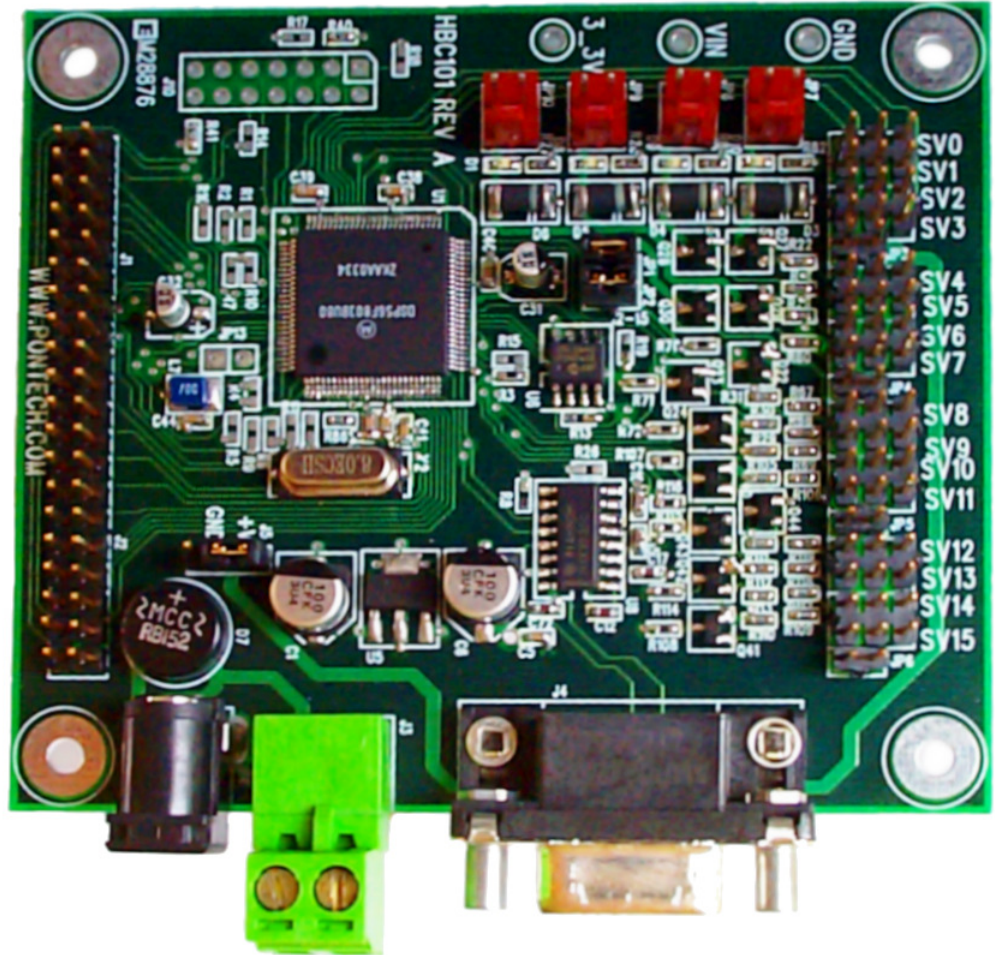


HBC101



Hybrid Motor Controller

Rev A (2004-April-2)
www.pontech.com

HBC101 - Table of Contents

HBC101 - Table of Contents	1
Introduction to the HBC100.....	4
Servo Basics.....	4
Power Supply and Pin-out	5
J3 / P1 Power Connections	5
J4 DB9 RS-232 Connector.....	6
HBC101 - Servo Connections.....	6
J1 / J2 Expansion Port.....	7
Jumper Pin Configurations	8
Example Interface	8
Operation of Board	10
Commands Descriptions	11
Board Control.....	11
BDn Select Board	11
WRm n Write to RAM.....	11
RRm Read from RAM.....	11
WEm n Write to EEPROM.....	12
REm Read from EEPROM	12
WSS Write System Settings to EEPROM	12
RC Servo Motor Control.....	13
SVn Select Servo	13
Mn Move to absolute position.....	13
In Incremental move relative to current position	13
Digital Input and Output	13
PSn Pin Set	13
PCn Pin Clear.....	13
PTn Pin Toggle.....	13
Analog to Digital Conversion	14
ADn Read a voltage on the A/D port.....	14
Serial Peripheral Control.....	14
SI Shift byte In the SPI port	15
SOn Shift byte Out the SPI port.....	15
Time and Timing.....	15
Dn Delay/Pause in milliseconds (ms).....	15
Stepper Motor Control	16
MIn Move Immediately to absolute position.....	16
MCn Move Cued to absolute position	16
IIn Increment Immediately relative to current position.....	16
ICn Increment Cued relative to current position.....	16
CU Cue to move or increment.....	16
PSn Pin Set	16
PCn Pin Clear.....	16
RPn Read Pin.....	17
ADn Read a voltage on the A/D port.....	14
RC Read Current motor position.....	17
RD Read Destination motor position	17
RT Read delTa motor position (Destination - Current)	17
HMn Re-home stepper	18
H0 Halt with deceleration (H-Zero).....	18

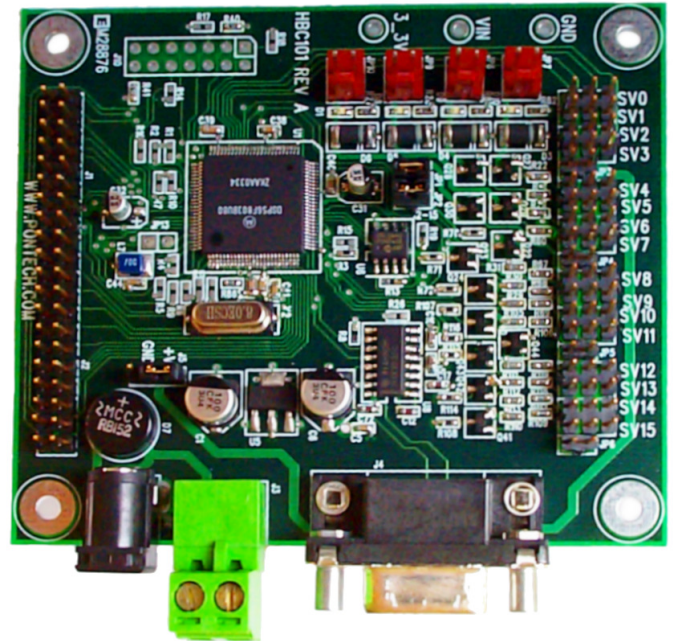
HI	Halt Immediately, do not decelerate	18
H+	Advance CW indefinitely	18
H-	Advance CCW indefinitely	18
SO	Stepper Off when not moving.....	18
SP	Stepper always Powered	18
SH	Step Half	18
SF	Step Full	18
SW	Step Wave	18
TCn	Test if pin Clear, on clear execute HI	19
TSn	Test if pin Set, on set execute HI	19
SDn	Step Delay	20
RSD	Read Step Delay.....	20
SAn	Set Acceleration/Deceleration Factor	20
RSA	Read Acceleration/Deceleration Factor	20
SMn	Set Minimum Step Delay Factor.....	20
RSM	Read Minimum Step Delay Factor	20
Changing Default Settings		21
Controlling Multiple Boards		22
Notes		23
Commands Listing		24
Warranty and Copyrights		28
Copyright and Trademarks		28

Introduction to the HBC100

The HBC101 is a Motorola DSP (Digital Signal Processor) based servomotor IO board. The board can both control as well as capture RC (Radio Control) servo PPM (Proportional Pulse Modulation) signals. The board has eight high-resolution PPM outputs and six high-resolution PPM inputs; additionally there are two low-resolution PPM inputs.

Unused servo output pins can be reconfigured as digital outputs for controlling on/off devices. Four outputs have been buffered so that small devices such as relays and solenoids may be directly driven.

There is an 8-channel 12-Bit ADC (Analog to Digital Converter) port for reading analog voltage between 0-3.3 volts, and a SPI (synchronous peripheral interface) port which allows data to be shifted in or out serially.



HBC101 Feature List

- 115,200bps true RS-232 level IO
- Eight 100ns resolution PPM outputs (500ns to 6.5us high pulse)
- Six PPM inputs with 100ns resolution (500ns to 6.5us high pulse)
- Two PPM inputs with 1us resolution (???us to ???us high pulse)
- Unused servo outputs can be reconfigured for digital output of which four of them can sink up to 300mA for driving small relays or solenoids
- Unused servo inputs can be reconfigured for digital input. All of which are buffered for up to a 40V input signal.
- Six 15-bit resolution 0% to 100% PWM outputs for driving H-Bridge circuits
- SPI bus with multiple chip selects
- Eight 12-bit ADC pins for measuring 0V-3.3V analog signals
- Field Upgradeable Firmware
- AC or DC Operation
- Ability to transmit and receive Futaba* trainer port signals (Expansion Board Required)

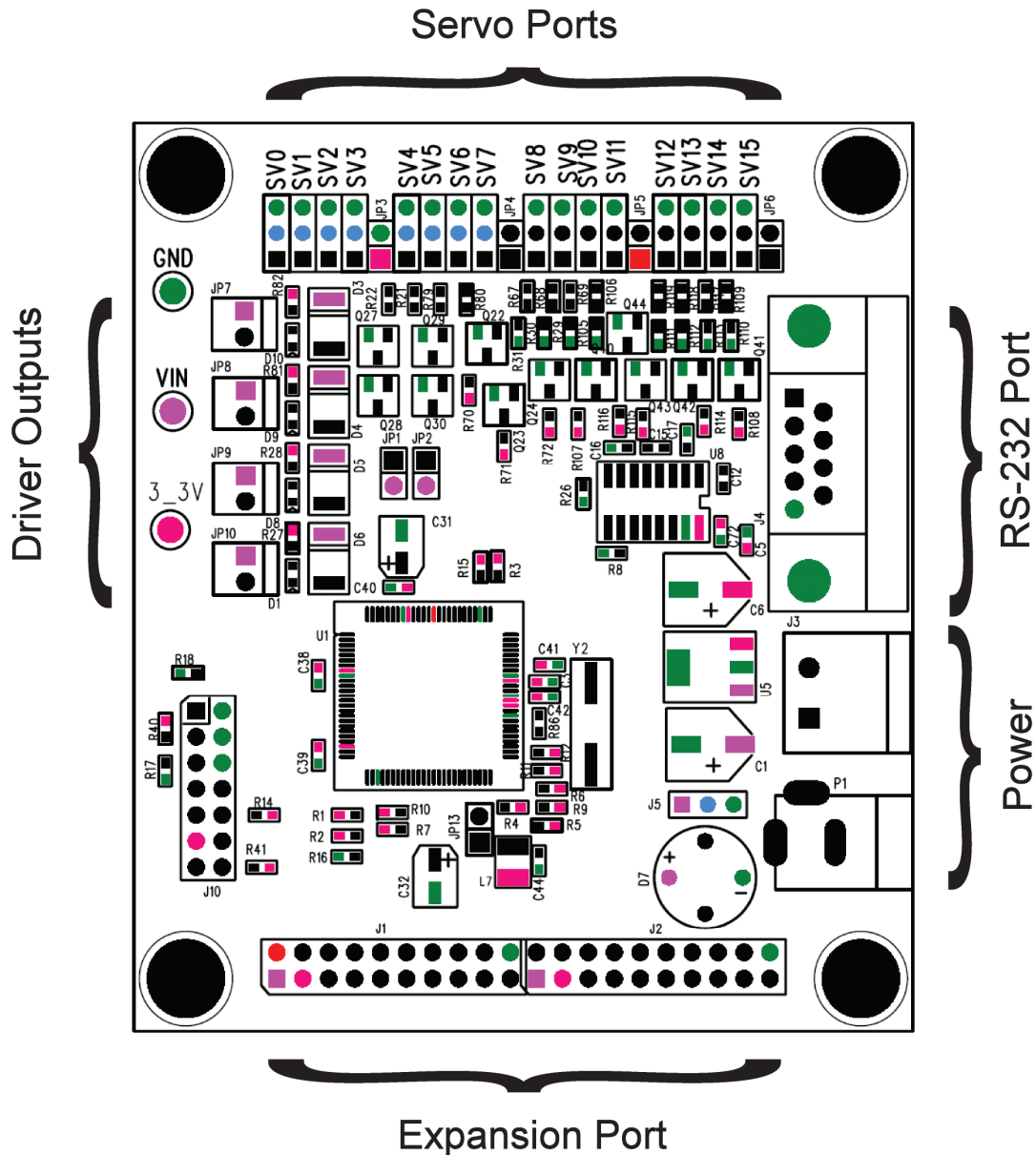
*Futaba – Manufacturer of RC transmitters, receivers and servos.

Servo Basics

RC servos operate using feedback to compare the current position to an input pulse width, which typically repeats every 14 to 20 ms (milliseconds). If the pulse width lasts for approximately 0.6 ms, the servo will rotate to a maximum position. If the pulse width is increased to approximately 2.4 ms, the servo will rotate to the opposite maximum position (Figure 1). A 1.5 ms pulse will set the servo in the middle (neutral) position.

The HBC101 controller is designed to the specifications of a Futaba servo model FP-S148. These servos have a neutral position at 1.52 ms. -90 degrees at 0.6 ms, and 90 degrees at 2.4 ms. Other servos may have slightly different values for these positions.

Power Supply and Pin-out



(Figure 1 – HBC101 Port Diagram)

J3 / P1 Power Connections

The HBC101 requires a minimum of 5V for proper operation. This supplied voltage can either be AC or DC.

The internal logic of the HBC101 is entirely 3.3V.

A 6-Volt DC source powers the board, either from 4 alkaline batteries or 5 NiCad cells. An AC adapter can also be used: 6VDC, at 300mA. If using NiCads, a 4-cell pack might be easier to find than a 5-cell pack. The board will operate fine with 4 cells, but may not last as long as 5 cells.

J4 DB9 RS-232 Connector

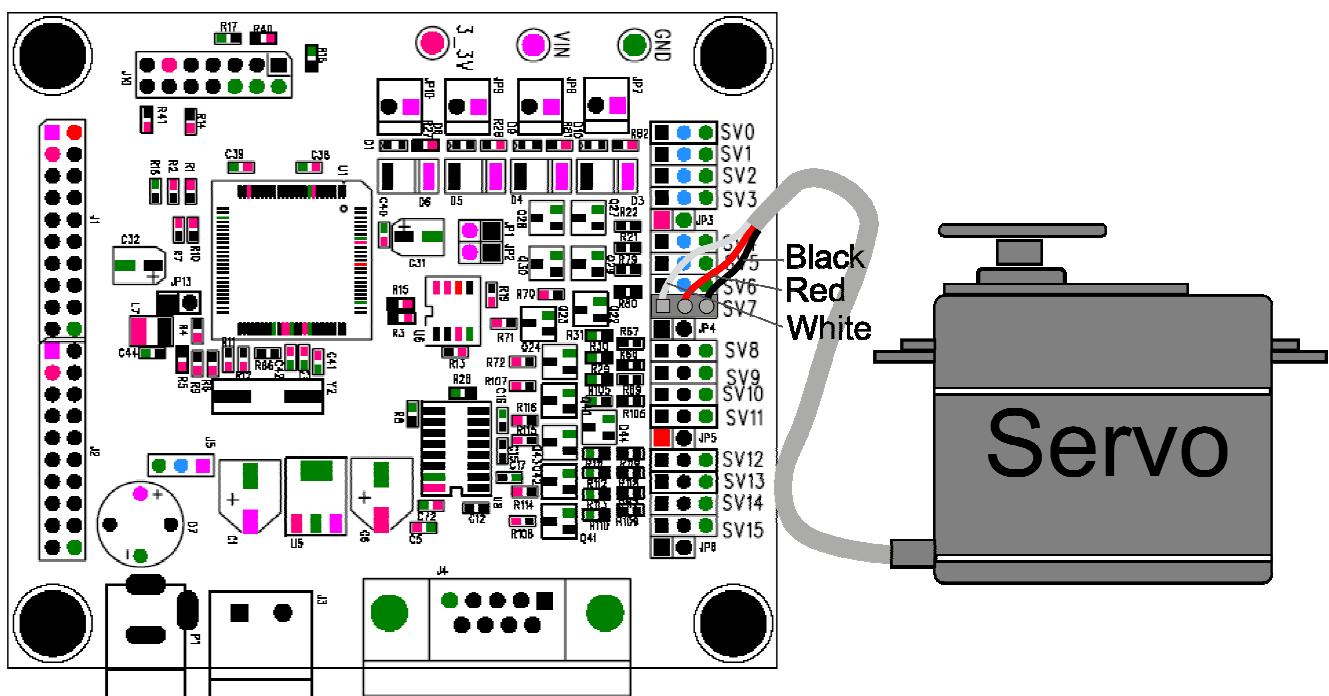
DB-9	DB-25	Macintosh	DB-9
1	--	3	1
2	5	2	2
3	3	--	3
4	--	--	4
5	8, 4	7	5
6	--	--	6
7	--	--	7
8	--	--	8
9	--	--	9

(Figure 2a – DB cross-referenced chart)

HBC101 - Servo Connections

The servo port connectors use a 3-pin male sip (single inline pin) connector (0.1-inch spacing). The servo connector is designed for use with Futaba-type servos with J-type connectors. The servos have three colored wires, Black for ground, Red for power, and White for signal.

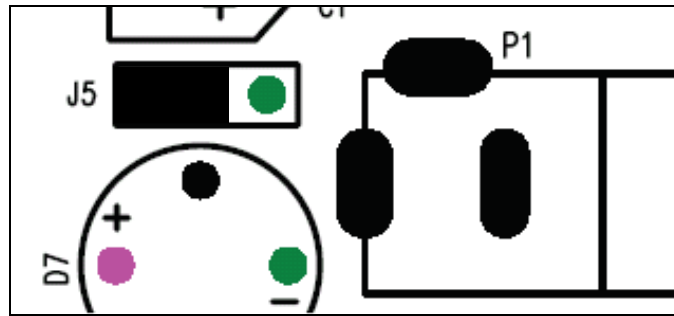
Connect the cable with the **black wire** to the pin of the connector closet to the silk screen labels (S0...S7) on the PCB, and the **white wire** away from the outside edge of the board as shown below.



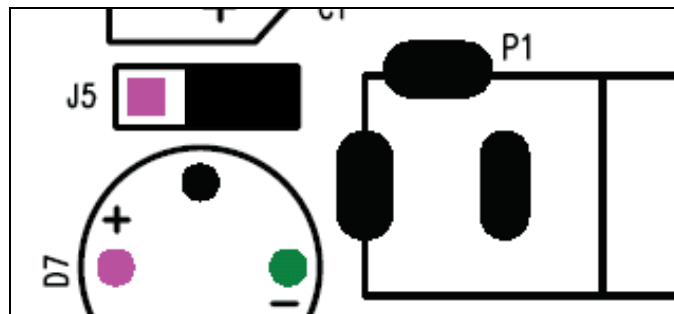
(Figure 3 - servo connected to SV7 of the HBC101)

J1 / J2 Expansion Port

Jumper Pin Configurations



Short pins 1-2 of J5 for the center pin of the servo outputs to be tied to +V-IN



Short pins 2-3 of J5 for the center pin of the servo outputs to be tied to GROUND

Example Interface

```
OPEN "COM1:9600,N,8,1,CD0,CS0,DS0" FOR OUTPUT AS #1
DO
  PRINT:PRINT "*****SV203 Servo Controller *****"
  INPUT "Enter Board ID Number:"; ID$
  INPUT "Enter Servo # to control:"; Servo$
  INPUT "Enter Position of Servo:"; Pos$

  PRINT #1, "BD";ID$;"SV";Servo$;"M";Pos$
  INPUT "Quit (y/n)"; Q$: IF Q$ = "y" THEN EXIT DO
LOOP
```

(Listing 1 - Sample interface program using QBASIC)

```

***** SV203 Servo Controller *****"
Enter Board ID Number:? 1           Board width ID = 1
Enter Servo # to control:? 2       move servo #2 to
Enter Position of Servo:? 200     position 200
Quit (y/n):? n

***** SV203 Servo Controller *****"
Enter Board ID Number:? 0           Any Board regardless of
Enter Servo # to control:? 4       ID, move Servo #4 to
Enter Position of Servo:? 254     position 254
Quit (y/n):? n

***** SV203 Servo Controller *****"
Enter Board ID Number:? 1           Board with ID = 1
Enter Servo # to control:? 2       Turn off Servo #2
Enter Position of Servo:? 0
Quit (y/n):? n

***** SV203 Servo Controller *****"
Enter Board ID Number:? 1           This command is invalid because
Enter Servo # to control:? 9       there is no Servo #9 and also
Enter Position of Servo:? 256     256 is out of the range of the
Quit (y/n):? y                     servo position.

```

(Listing 2 - Example screen when program in Listing 1 is running)

Listing 1 is a simple program written in QBASIC that requests the user to input a board ID number, a servo number to control, and the position of the servo.

Operation of Board

Connect the servos to the board; plug in the RS-232 cable to a COM port of a PC and the other end to the SV203. Power the board and the servos should return to the neutral position. Run the example programs.

The SV203 processes information one ASCII string command at a time. Each command string follows the format:

```
L n L n ... <enter> (maximum of 20 characters/line)
```

Where *L* is the command letter(s) in caps, *n* is a decimal integer number(s), and *<enter>* is ASCII 13. Please refer to Command Listing Page (p.26) for a complete listing.

For example, the commands to select a board, select a servo and move to a position are BD, SV, and M, respectively. If your want to move servo #3 of a board with an ID number equal to 1 to position 85, you would send the flowing command string.

```
BD1SV3M85 <enter>
```

Spaces or commas for ease of reading can also separate the commands.

```
BD1 SV3 M85 <enter> or  
BD1,SV3,M85 <enter> or  
BD 1 SV 3 M 85 <enter>
```

A terminal program may be used to test the functions of the board. The default setting on the board is 9600 baud, N81, with echo off. (A program called TERM.EXE is provided as a simple terminal like software to test the SV203 functions).

Once a board or servo is selected, it will stay selected until power is removed or another select command is received. For example, if the following commands were sent:

```
BD1 SV2 M100 <enter>
```

The next command will still move servo #2:

```
M150 <enter>
```

More then one servo can be moved at the same time in one line of a command string, just make sure you don't exceed the 20 characters per line limit, including spaces and commas:

```
BD1 SV1 M30 SV2 M104 SV3 M25 <enter>
```

The commands above will select Board #1, Move servo #1 to position 30, then move servo #2 to position 104 and move servo #3 to position 25 all at about the same time.

Any parameter value for the command not in the range of the command will be ignored. (See Command Listing Page – p.26.)

The board will start processing the command string when it receives the *<enter>* or ASCII 13 character. The host computer talking to the board should insert a delay of about 3 milliseconds between each command string to allow time to process the commands.

Commands Descriptions

The HBC101 command set is divided up into functional sub systems. The list of sub systems and a brief description is given below:

BD - Board Control
SV - RC Servo Motor Control
STP - Stepper Motor Control
DIO - Digital Input and Output
ADC – Analog to Digital Conversion
TIME - Time and timing
SPI - Serial Peripheral Control

Board Control

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 (*n*). 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 (see Commands Descriptions Page – p.20). This allows multiple boards of different ID number to be connected to the same serial port.

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, such as:

```
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.

There is a configuration register at location 60 in RAM. The value of the register is initialized by the contents of EEPROM in location 11. The register configures the shift function for MSF (most significant first) or LSF (least significant first), data valid on clock going high or low, and the number of bits to shift in/out.

WRm n Write to RAM **RRm Read from RAM**

Where *m* is the memory address (see table on next page),
n is the value to be stored.

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 contents will be erased.

i.e. WR51 20 <enter>
 Servo #1 will move to position 20, this command is equivalent to SV1 M20 <enter>

i.e. RR52 <enter>
 The position of servo #2 will be returned.

RAM Memory Map:

Address (m)	Usage	Note
5	Port A	
6	Port B	
7	Port C	
14	TMR1L	
15	TMR1H	
16	TICON	
27	CCPR2L	
28	CCPR2H	
29	CCP2CON	
51 to 58	Current Servo Position	
59	Servo Select	
60	Shift Config. Register	
133	TRIS A	
134	TRIS B	
135	TRIS C	

Note: All other RAM locations not listed are used by the system and should not be used.

WEm n Write to EEPROM
REm Read from EEPROM

Where *m* is the memory address (see table on next page),
n is the value to be stored.

These commands allow you to modify and read the contents of the external EEPROM connected to the processor. The EEPROM is a non-volatile memory storage, so any information written to it will stay even when power is removed.

i.e. WE0 2 <enter>
 Change the board ID number to #2

i.e. RE1 <enter>
 The SV203 returns the initial servo value of servo #1

EEPROM Memory Map:

Address (m)	Usage	Factory default	Note
0	Board ID #	1	
1	Initial Servo #1 Value	128	0 - off or Digital
2	Initial Servo #2 Value	128	0 - off or Digital
3	Initial Servo #3 Value	128	0 - off or Digital
4	Initial Servo #4 Value	128	0 - off or Digital
5	Initial Servo #5 Value	128	0 - off or Digital
6	Initial Servo #6 Value	128	0 - off or Digital
7	Initial Servo #7 Value	128	0 - off or Digital
8	Initial Servo #8 Value	128	0 - off or Digital
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	Shift Config. Register	0	MSB, valid on Rising, 8-bit

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.

RC Servo Motor Control

SVn Select Servo

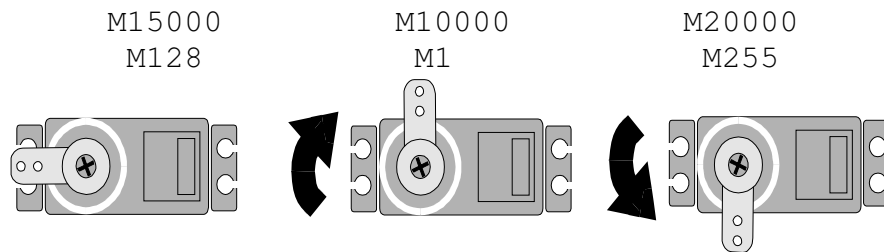
On power-up, servo #1 is pre-selected. To select another servo or to make sure the servo is selected, send “SV” followed by the servo number. The servo number must be between 1 and 8 because there is a maximum of eight servos that the board can control.

i.e. SV3 <enter>

 Servo #3 will be selected. Any move commands that follow will operate on servo #3.

Mn Move to absolute position

This command will move the selected servo to an absolute position. The range of the position is between 1 and 255. On a RC-type servo, the maximum mechanical movement is about 180 degrees. The 1 to 255 position ranges gives a precision of a little under one degree. The figure below shows servo position at respective value.



Sending a zero to the servo can turn off the pulse-width command signal to the servo, which causes the servo to remove power from the motor:

 M0 <enter>

In Incremental move relative to current position

This command will move the servo relative to its current position by adding or subtracting the value entered to the current position.

i.e. M100 <enter>

 I10 <enter>

 I-20 <enter>

 The selected servo will first move to position 100, then to position 110 (100+10), and then finally to position 90 ((100+10) - 20).

Digital Input and Output

PSn Pin Set

PCn Pin Clear

PTn Pin Toggle

These commands allow you to use the servo port as a digital output by setting, clearing or toggling individual bits of the servo port. In order to use the port as a digital output, the servo PWM must first be turned off by sending a M0 command to each servo pin that you want to use as digital output.

i.e. SV7 M0 PS7 <enter>
Pin S7 of the servo port will be set high (5 Volts).

i.e. SV8 M0 <enter>
PT8 <enter>
Pin S8 of the servo port will be toggled/flipped
(Set high if pin was low, or cleared if pin was high).

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.

Analog to Digital Conversion

ADn Read a voltage on the A/D port

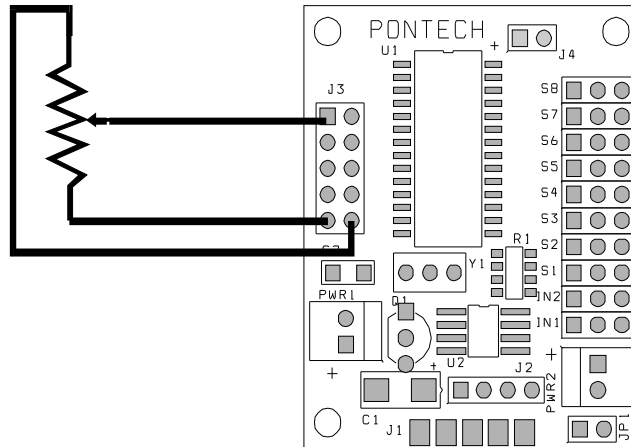
Port A (see figure on previous page) is a 5-Channel, 8-bit A/D input that can read an analog voltage. The n is a number between 1 to 5, which tells which pin on the A/D port to request. When the board receives this command, it will read the specified voltage on the pin and return a value between 0 to 255, which represents a voltage between 0 to 5 Volts.

i.e. AD1 <enter>
If wires were connected as in the figure below and the potentiometer as in the middle position, the board would return a value close to 28 followed by <ASCII 13>, which is about 2.5 Volts.

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 receives this command, it will read the voltage on the pin specified and return a value between 0 to 255 that represent a

5k to 10k Pot

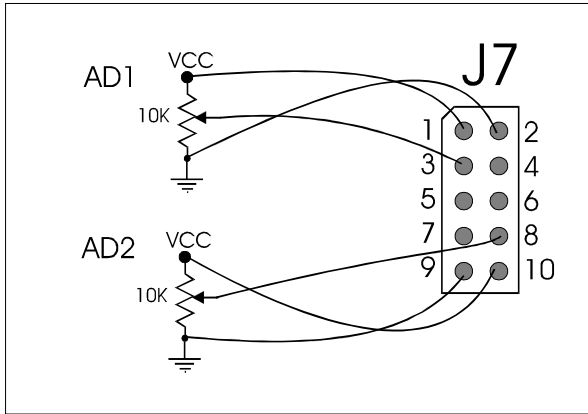


voltage between 0 to 5 Volts.

$$\text{Voltage} = \text{value} / 255 * 5 \text{ Volts}$$

i.e. 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.



Serial Peripheral Control

SI **Shift byte In the SPI port**

SOn **Shift byte Out the SPI port**

These commands allow you to use Port C (see figure on p.14) on the board as a synchronous serial port to shift in or out a byte of data. The figure below shows how to connect shift register chips (74HC164 or 74HC165) to the board to make a serial-to-parallel or parallel-to- serial converter.

Use the **SO** command to make a serial-to-parallel converter

i.e. `S03 <enter>`

Pins 3 and 4 of the '164 will be set high, while all others are low (3 decimal → 00000011 binary).

Use the **SI** command to make a parallel-to-serial converter

i.e. `SI <enter>`

The board will return a number between 0 to 255.

Time and Timing

Dn **Delay/Pause in milliseconds (ms)**

Delay commands may be added to the string to pause between movements:

```
SV1 M20 D1000 M100 <enter>
```

Servo #1 will move to position 20. There is a one-second (1000 ms) pause, and then servo #1 will move to position 100.

Caution: When using the Delay command, the board will not receive input from the serial port during the delay state. The host computer that is talking to the board has to wait at least the same amount of time before another command string can be sent. Any commands sent during the delay will be ignored.

Stepper Motor Control

MI_n 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 STP100 is set for half-stepping, MI moves the stepper in half steps.

MC_n 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 STP100. This is useful for synchronizing multiple boards connected to the same network.

II_n Increment Immediately relative to current position

This command will move the stepper motor relative to its current position by adding or subtracting the value entered to the steppers motor destination position.

i.e. MI1000 <enter>
II100 <enter>
II-200 <enter>

The steppers destination will first be set to position 1000, then to position 1100 (1000+100), and then finally to position 900 (1100 - 200). If the stepper is in motion the II command will not wait for the motor to stop before adjusting the destination position. If the STP100 is set for half-stepping, II moves the stepper in half steps.

IC_n 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 STP100. 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 BD0 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 BD0CU string is received. The last command just makes sure only one board stay selected.

```
BD1MC200 <enter>
BD2MC400 <enter>
BD3IC1000 <enter>
BD0CU <enter>
BD1 <enter>
```

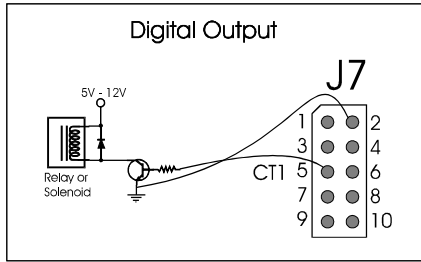
PS_n Pin Set

PC_n 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.

i.e. 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.

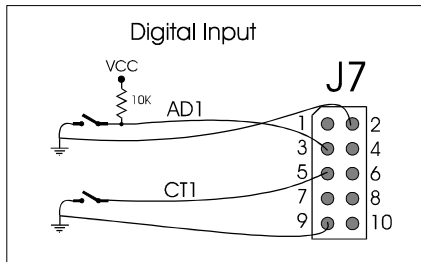


RPn Read Pin

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.

i.e. 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>.



RC Read Current motor position

RD Read Destination motor position

Read the current or destination motor position.

i.e. 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.

Performing this command is the same as doing a RD command and a RC command and then taking the result of RD and subtracting the result of RC (RD - RC).

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 than what it should be.

i.e. RT <enter>

The board will return a value between -2147483648

and 2147483647 followed by <ASCII 13> and <ASCII 10>..

HMn **Re-home stepper**
(Current position = destination position = n)

Set a new home position.

i.e. HM0 <enter>
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 HX 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 HX commands except HI will accelerate and decelerate the motor appropriately.

- SO** **Stepper Off when not moving**
- SP** **Stepper always Powered**

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.

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 STP100 into full step, half step or wave step mode. When any one of these commands is executed the STP100 may move up to a step from its current position.

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

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			+	-

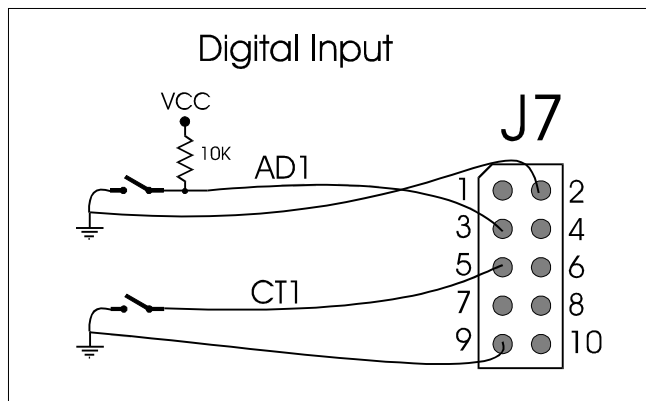
TCn **Test if pin Clear, on clear execute HI**

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 STP100. 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 are used as home limit sensors, external pull-up resistors must be used.



Acceleration and Speed Commands

The STP100 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 “Step Delay”, “Minimum Step Delay Factor” and “Acceleration Factor”. The delay used between steps while the motor is accelerating up to its operating speed is calculated as such:

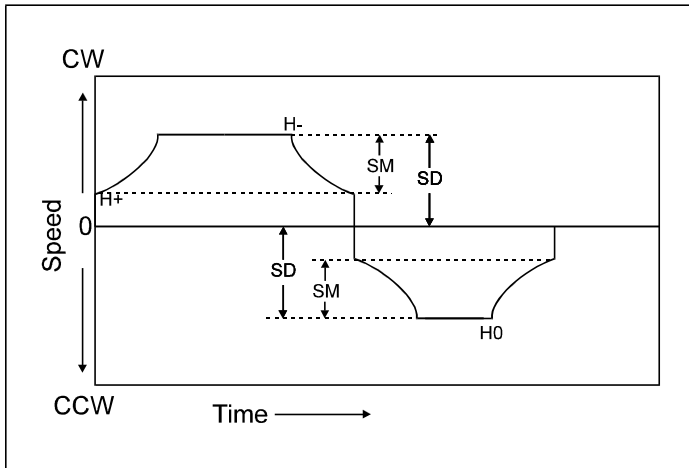
$$\text{“Current Step Delay”} = \text{“Step Delay”} + \text{“Minimum Step Delay Factor”} - (\text{“Acceleration Factor”} * \text{steps})$$

Where: “Current Step Delay” >= “Step Delay”

The board starts with “Current Step Delay” equal to “Step Delay” plus “Minimum Step Delay Factor”. Then “Current Step Delay” is subtracted each step by a factor of “Acceleration Factor” until “Current Step Delay” is greater than or equal to “Step Delay”.

To disable the acceleration/deceleration feature use “Minimum Step Delay Factor” = 0 (SM0 command).

To use the acceleration/deceleration feature, start with a “Minimum Step Delay Factor” = 2000 (SM2000) and “Acceleration Factor” = 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 RSD command is used to read the Step Delay value.

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

SA n 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.

SM n 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.

i.e. SM1000 <enter>

If SD was previously set to 2000 and SA was previously set to 10,
 The “Current Step Delay” is initially assigned to 3000 (SD+SM). As the motor moves during each step, “Current Step Delay” is subtracted by 10

until “Current Step Delay” is greater or equal to SD. The “Current Step Delay” stays at 2000 until it’s time to decelerate. When decelerating “Current Step Delay” is added by 10 until it reaches the Destination Position.

Changing Default Settings

When the board is powered-up, it first reads the external EEPROM to get its Board ID number, Initial positions for the servo, Baud rate, and some other initial flags. These initial settings can be changed by using the WE command (for WE command see p.20).

To change ID:

i.e. WE03 <enter>
#3 is the new ID number

To have the board be pre-enable on power-up:

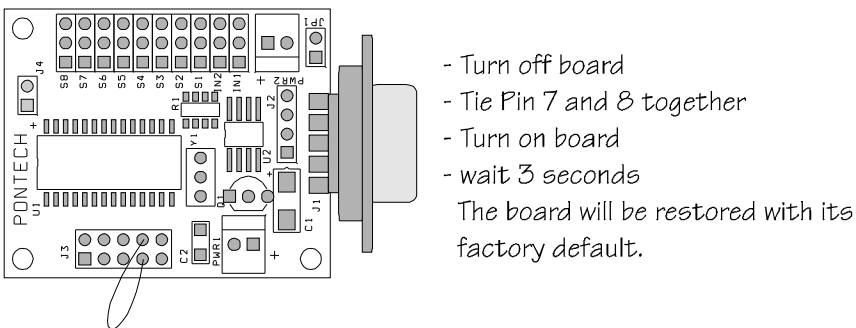
i.e. WE101 <enter>

To change Servo initialization:

i.e. WE20 <enter>
Servo #2 will be off when the board is turned on.

See the Memory Map on previous page for other default settings.

The board can be reset to its factory default settings by tying pins 7 and 8 of Port C together, power-up the board, and wait for about 3 seconds.



You can also use the enclosed program on disk called DEFAULT . EXE to reset the default settings of the board.

If the baud rate of the SV203 has been altered, either on purpose or inadvertently, the DEFAULT.EXE program cannot be used. In this case, the above method of shorting pins 7 and 8 of connector J3 is preferred.

Notes

Commands Listing

Commands	Parameter (n)	Description
BD_n	0 to 255	Board Select
SV_n	1 to 8	Servo Select
M_n	0 to 255	Move to an absolute location
I_n	-128 to 127	Move relative to current position
D_n	1 to 65535	Delay (ms)
PS_n PC_n PT_n	1 to 8	Put, set single pin of servo port b is a number between 1 and 8, which tells what pin on the port is being used. If $n = 0$ → clear Bit $n = 1$ → Set Bit $n = 2$ → Toggle Bit
AD_n	1 to 5	Get A/D value, the board will return a value between "0" to "255" followed by ASCII 13 which represent a voltage between 0 to 5 Volts.
SO_n	0 to 255	Shift a byte out to the SPI port
SI	None	Shift a byte in form the SPI port
WR_{m n}	$m = 0$ to 255 $n = 0$ to 255	Write to internal RAM m is the memory location n is the value to write
RR_m	$m = 0$ to 255	Read the contain of internal RAM m is the memory location to read
WE_{m n}	$m = 0$ to 8190 $n = 0$ to 255	Write to external EEPROM m is the memory location n is the value to write
RE_m	$m = 0$ to 8190	Read the contents of external EEPROM m is the memory location to read
?	None	Help, return summary of command listing
V?	None	Returns the firmware version

Command Set

C1	C2	C3	Sub System	Command	Parameter (n, m)	Return Value	Description
			?	RPS			
			?	RSC			
			?	RY			
-	-	-	ADC	ADn	n = 1 to 8	0 to 4095	Get A/D value, the board will return a value between "0" to "4096" which represents a voltage between 0 to 3.3 Volts.
-	-	-	BD	?	None		Help, return summary of command listing
-	-	-	BD	?V			Query Firmware Version
-	-	-	BD	\0'			Return Prompt
-	-	-	BD	BDn	n = 0 to 255		Board Select
-	-	-	BD	BR			Set Baud Rate
-	-	-	BD	DE			Flash Erase
-	-	-	BD	DP			Flash Write
-	-	-	BD	DR			Flash Read
-	-	-	BD	JP			reset the processor
-	-	-	BD	LU			Lookup binary command equivalent
-	-	-	BD	RE	M = 0 to 8190		Read the contain of external EEPROM, m is the memory location to read
-	-	-	BD	REm	m = 0 to 8190		Read the contents of external EEPROM m is the memory location to read
-	-	-	BD	RR	M = 0 to 255		Read the contain of internal RAM, m is the memory location to read
-	-	-	BD	RRm	m = 0 to 255		Read the contain of internal RAM m is the memory location to read
-	-	-	BD	V?	None		Returns the firmware version
-	-	-	BD	WE m n	M = 0 to 8190 N = 0 to 255		Write to external EEPROM, m is the memory location n is the value to write
-	-	-	BD	WR m n	M = 0 to 255 N = 0 to 255		Write to internal RAM, m is the memory location n is the value to write
-	-	-	BD	WSS			Write System Settings to EEPROM (current value of SD, SM, SA, SH, SF, SW, SP, SP are stored to EEPROM)
-	-	-	DIO	?Pn	n = 8 to 15	0 or 1	read Pin on servo input connectors
-	-	-	DIO	PCn	n = 0 to 7		Pin Clear n (servo port) if PWM is disabled (see SPE)
-	-	-	DIO	PSn	n = 0 to 7		Pin Set n (servo port) if PWM is disabled (see SPE)
-	-	-	DIO	PTn	n = 0 to 7		Pin Toggle n (servo port) if PWM is disabled (see SPE)
-	-	-	DIO	RPn	N = 3, 5, 6, 8 or None		Read Pin (If n is not specified, a ?-bit value is returned representing all pins)
-	-	-	HB	?DCn	n = 0 to 2	0 to 10,000	read Duty Cycle for h-bridge n
-	-	-	HB	DA			Duty Cycle
-	-	-	HB	DB			Duty Cycle
-	-	-	HB	DC			Duty Cycle
-	-	-	HB	DCn m	n = 0 to 2 m = 0 to 10,000		set Duty Cycle for H-Bridge n to m 0.01%
-	-	-	SPI	SI	None		Shift a byte in form the SPI port

C1	C2	C3	Sub System	Command	Parameter (n, m)	Return Value	Description
-	-	-	SPI	SOn	0 to 255		Shift a byte out to the SPI port
-	-	-	STP	CU			Cue Cued Commands
-	-	-	STP	H-	None		Set stepper motor in motion in negative direction forever
-	-	-	STP	H+	None		Set stepper motor in motion in positive direction forever
-	-	-	STP	H0 (H-Zero)	None		Halt stepper motor, with deceleration
-	-	-	STP	HI	None		Halt Immediately, set destination position to current position and do not decelerate
-	-	-	STP	HM	N = - 2147483648 to 2147483647		Set new stepper motor home position, current position = destination position = n
-	-	-	STP	IC	N = - 2147483648 to 2147483647		Move Cued to a position relative to the current stepper destination position
-	-	-	STP	II	N = - 2147483648 to 2147483647		Move Immediately to a position relative to the current stepper destination position
-	-	-	STP	MC	N = - 2147483648 to 2147483647		Move Cued to an absolute stepper position
-	-	-	STP	MI	N = - 2147483648 to 2147483647		Move Immediately to an absolute stepper position
-	-	-	STP	RC	None		Read stepper motor Current position
-	-	-	STP	RD	None		Read stepper motor Destination position
-	-	-	STP	RL	None		Read deceleration point
-	-	-	STP	RSA	None		Read Step Acceleration/Deceleration Factor
-	-	-	STP	RSD	None		Read Step Delay
-	-	-	STP	RSM	None		Read Minimum Step Delay Factor
-	-	-	STP	RT	None		Read stepper motor delTa position (DestinationPosition - CurrentPosition)
-	-	-	STP	RX	None		Read stepper motor direction sign ('+', '-', '0')
-	-	-	STP	SAn	n = 1 - 255		Acceleration/Deceleration Factor
-	-	-	STP	SDn	n = 6 to 65535		Step Delay (n x 1.6us)
-	-	-	STP	SF	None		Stepper Full step mode
-	-	-	STP	SH	None		Stepper Half step mode
-	-	-	STP	SMn	n = 0 to (65535 - SD)		Minimum Step Delay Factor (Start Step Delay = StepDelay + MinimumStepDelayFactor) if = 0 then no acceleration
-	-	-	STP	SO	None		Stepper coils Off when not moving (Remove power from stepper, no holding torque)
-	-	-	STP	SP	None		Stepper coils always Powered (even when motor is not moving)
-	-	-	STP	SW	None		Stepper Wave step mode
-	-	-	STP	TCn	?		Test pin n, on Clear (logic 0) execute H0
-	-	-	STP	TSn	?		Test pin n, on Set (logic 1) execute H0
-	-	-	STP	TTn	0 or 1		Limit Switch Mode

C1	C2	C3	Sub System	Command	Parameter (n, m)	Return Value	Description
X	X	X	SV	?B			Read Servo Positions (Binary)
X	X	X	SV	Mn	n = 0 to 255 (Mode 0) n = 500 to 65,535 (Mode 1)		Move servo to an absolute location
X	X	X	SV	?M	n = 0 to 15 (+1 in Mode 0)		Read absolute location of servo n
X	X	X	SV	MSn	n = 0 to 1		0 = SV203 Mode(C2 default) 1 = Raw 100nS(10000 = 1ms) per count Mode(C1, C3 default)
X	X	X	SV	?MS		0 to 1	Read Current Scaling Mode
X	X	X	SV	SVn	n = 0 to 15 (+1 in Mode 0)		Servo Select
X	X	X	SV	SSn m	n = 0 to 15 (+1 in Mode 0) m = 0 to 15 (+1 in Mode 0)		Set Servo n output routing from servo m
X	X	X	SV	In	n = -255 to 255 (Mode 0) n = -32,767 to 32,767 (Mode 1)		Move servo relative to current position
X		X	SV	SPEn	n = 0 to 65,535		Servo PWM Enabled
X		X	SV	?SPE		0 to 65,535	Read Servo PWM Enabled
	X		SV	STn	n = 0 to 7		Trainer Signal Select
			TIME	Dn	n = 1 to 65,535		Delay (ms)

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 not apply if product has been damaged by accident, abuse, or misuse.

Copyright and Trademarks

Copyright © 2004 by ProLinear/PONTECH, Inc. All rights reserved.

Motorola is a registered trademark of Motorola, Inc.

PIC is a registered trademark of Microchip Technology, Inc.

FUTABA is a registered trademark of FUTABA Corporations.

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 of recovering, reprogramming, or reproducing and data stored in or used

with ProLinear/PONTECH, Inc. products.

