PARALLAX INC

599 Menlo Drive
Rocklin, CA 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: (888) 512-1024; sales@parallax.com
Tech Support: (888) 997-8267; support@parallax.com

Parallax Servo Controller – USB (#28823) Rev B

16-Channel Servo Control with a USB Interface

Introduction

With the proliferation of servos used in technology today comes the obligation to manage multiple servos each executing complex motions. The original Parallax Servo Controller (#28023) was designed to remove the burden of driving the servo pulses from the controller and thereby expand the number of servos that one controller could be expected to handle. To that end the original PSC was quite successful, but for some of the more complex applications writing a textual program to control many servos executing compounded motions and actions can be tedious to say the least.

One way to make the whole application development process easier is to change the primary interface from a keyboard (text-based code) to that of a mouse (visual based program). By adding a USB port to the servo controller and providing the user with a free graphical user interface, a direct link between mouse and servo is made. Move the mouse, the servo moves in real-time. Now the task of developing a program to control intricate servo sequences becomes a palatable, fun process.

The concept of the frame is introduced to help keep movements and sequences organized and manageable. Sixteen servo positions and time delays make up a frame; two or more frames constitute a sequence. Sequences may be stored as files on your PC's hard drive to be retrieved for later use. Like a movie, a sequence may be played, paused, stopped, or edited repeatedly until you get it just the way you want it.

With animatronics customers in mind, the current PSCI software supports up to 128 frames. (Most animal walking sequences require fewer than 8 frames.)

Features

- Each Servo is individually controllable: Position, rate, offset, and post delay
- Speed Control: Each servo may have any of 63 run-time changeable rates
- 128 Frames: supported though most sequences require far fewer frames

Before You Begin

The processor used on the PSC-USB requires 5 VDC which is supplied by the USB port. Servos require far more power than the USB port can supply. Therefore you must use a separate power supply for your servos. In general, servos require 4-7.5 VDC. Be sure that the servo power source can supply ample current at the proper voltage and will not damage the servos.

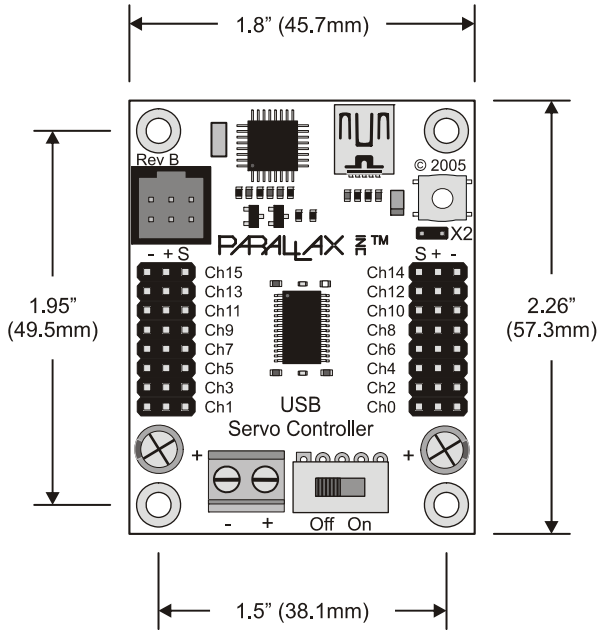
Getting Started

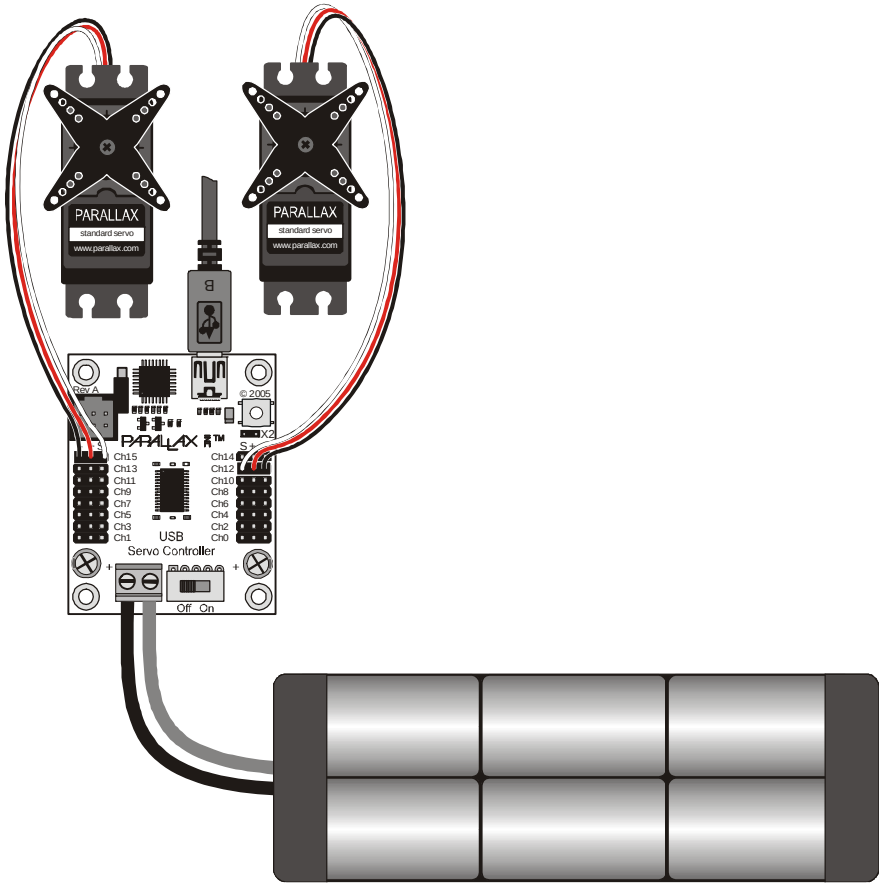
1. Locate and remove the PSC-USB from its protective anti-static bag. It should closely resemble the image shown (right)

2. Mount the PSC-USB using 4-40 screws (or equivalent). Ensure that none of the connections on the PCB bottom are inadvertently shorted.

3. Ensure the servo power switch on the Parallax Servo Controller is off, and then connect the power source for the servos to the screw terminals observing proper polarity.

4. Servos require more power than the USB port can supply. For this reason, you must connect a separate power supply for the servos, as shown below. * Note: Though a Rev A board is depicted here, the connections for the Rev B board are exactly the same.



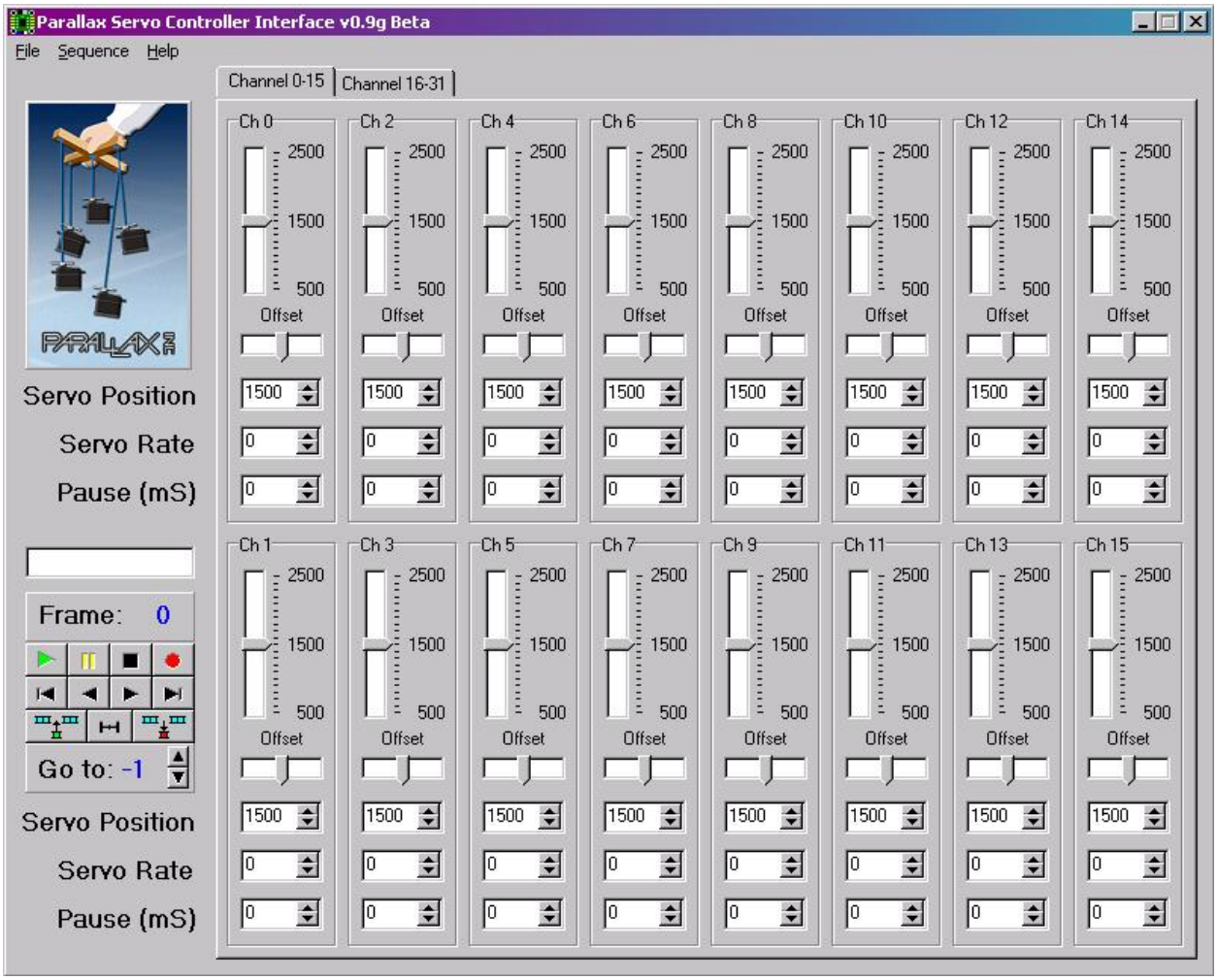


Installing the PSCI software

Go to www.parallax.com and search for 28823. On the 28823 product page under Downloads, find the link for the PSC Software for PC. Once the program is downloaded, double-click on it and follow the installation instructions.

Using the PSCI software

Once installed, launch the PSCI software. It should start immediately and the following screen will be displayed:



USB Drivers

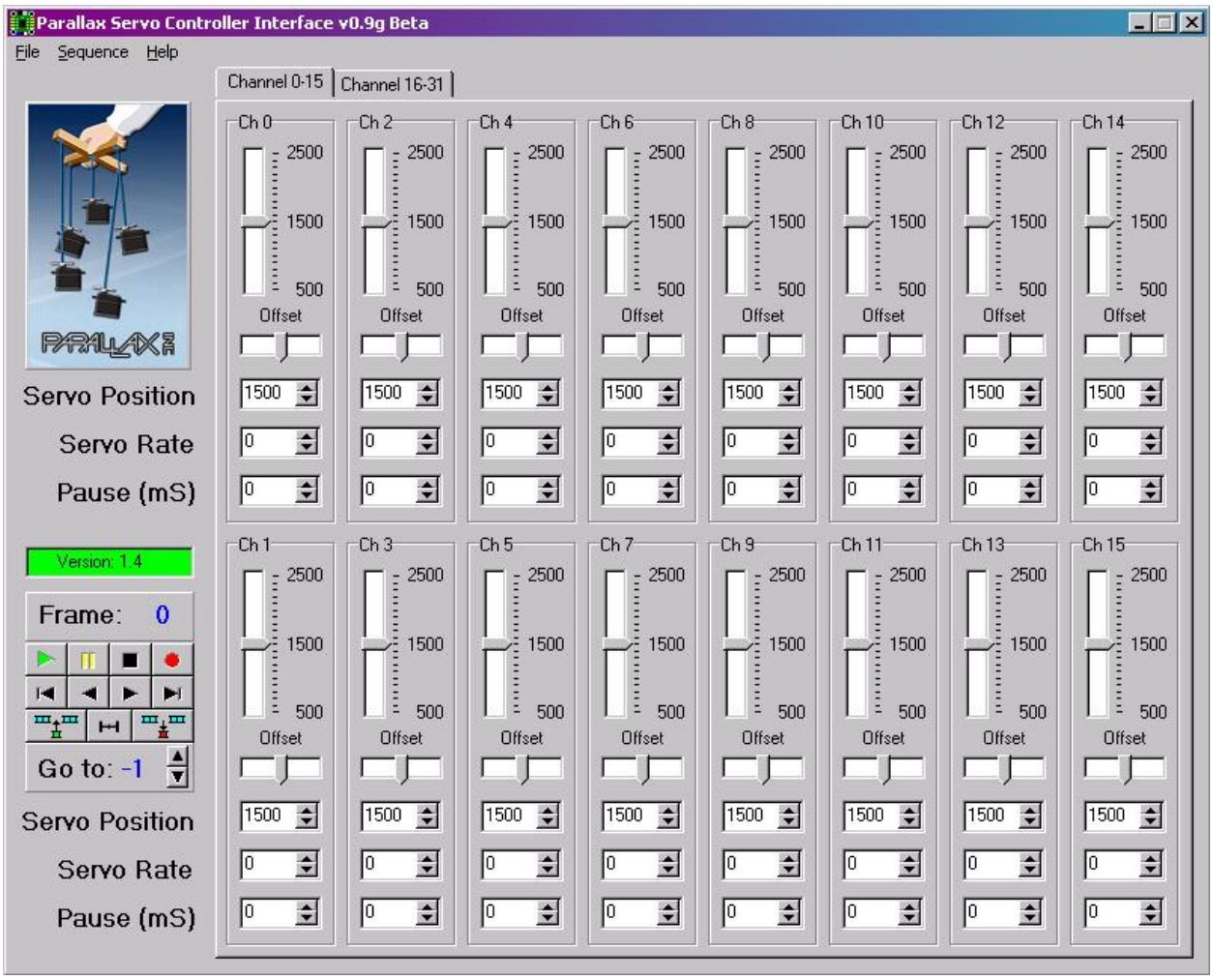
The FTDI chip used by the PSC-USB requires you to install USB drivers on your PC before using it. FTDI drivers for most Windows operating systems are included with the BASIC Stamp Editor and Propeller Tool software installers. You may also download them from the link on the home page of www.parallax.com, or go to www.ftdichip.com and check for updated drivers for your particular OS. After downloading has completed, close your web browser and double click on the driver file you just downloaded to start the installation process. Follow the prompts as directed. If you have any difficulty, simply call Parallax Tech Support toll-free at (888) 99-STAMP, [888-997-8267]. Now you are ready to use your new hardware.

Selecting the USB Port

Unlike serial ports, USB ports are dynamic. That is to say that they are manifested when the PSC-USB is connected, and extinguish themselves upon disconnecting. If you are so inclined to witness this phenomenon, simply open up the properties of the My Computer icon, then click on the Hardware tab, then click on the Device Manager button. Once in the Device Manager, click on the small + sign next to the Ports icon. Now, connect and disconnect the PSC-USB while watching the Ports list. For proper operation, connect your PSC-USB to the PC USB port **before** launching the PSCI software.

To select a COM port, simply click on the File tab and hover the mouse over the Select Comm Port entry to invoke the list of found comm ports. To choose a comm port, simply click on that comm port. If all goes well, a message will report the successful opening of the comm port.

If you wish to confirm the presence of the PSC-USB, you may then click on File->Get PSC version. If the PSC-USB is found, the version number will appear in the message window on the left side of the PSCI screen. See below.



Controls Description

To control the servos in real-time mode, connect your servos to the PSC-USB and simply slide the power switch on the PSC-USB to the ON position. Now, moving the corresponding servo position slide bar in the PSCI software positions the servo.

Offset

Not all servos are alike so the offset control allows you to vary the center point of each servo. This may not matter too much for standard servos, but is very helpful when controlling servos modified to rotate 360 degrees.

Position

You may use either the slide control or the up/down arrows, or enter the servo position numerically to set the servo position. Please note that by clicking on the numbers 2500, 1500, or 750, the servo position will be set to that number. These numbers correspond to the width of the servo command pulse, in microseconds.

Rate

Each servo may be set to a rate of rotation. Furthermore, each servo may have a different speed for each frame. A servo rate of 0 will cause the servo to move as quickly as possible; its fastest rate. A servo position of 63 will cause the servo to move to its destination position at its slowest rate. At speed 63, the servo takes about 45 seconds to complete a 180 degree motion.

Delay

When a frame is executed, as in animate mode, a servo command is sent to the PSC-USB for each servo in the order 0,1,2,3... After each servo command is sent, the delay time is observed before sending the next servo command. Even if a servo has not reached its destination before a new servo command is sent, it will immediately start executing the new command. The delay parameter should be long enough to let the servo travel to its destination. The Delay parameter has no effect when operating in real-time mode.

Frame

The number to the right of the word “Frame” indicates the current frame.

Go to

The number to the right of the phrase “Go to” shows the frame number that will be jumped to when all servo commands have been sent and all delays have been observed for the current frame. Note: no servo commands are sent for servos that appear to be unused.

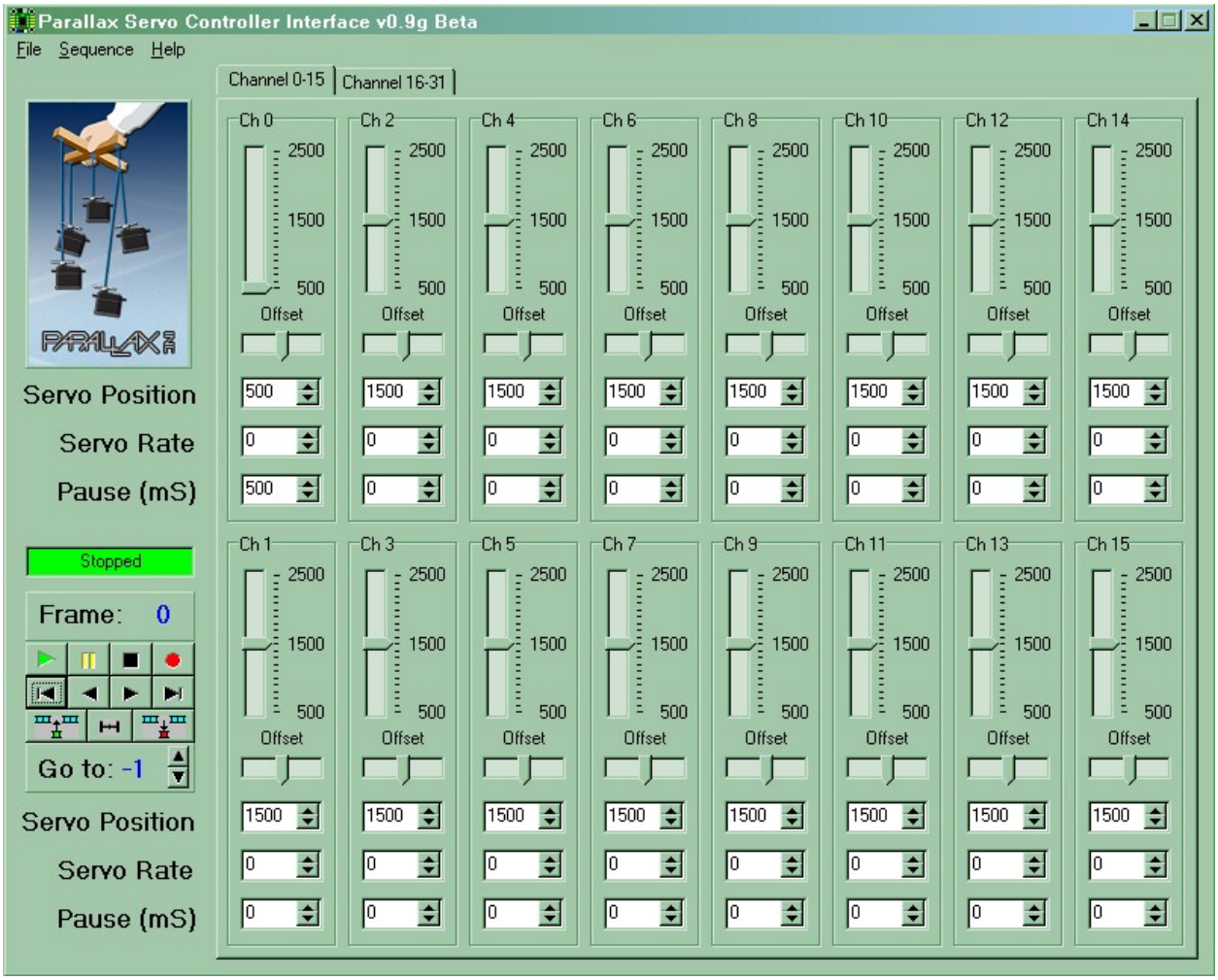
Animation Controls

Controls are provided to facilitate the creation, controlling, and editing of animation sequences. By hovering the mouse over each control button (without clicking) a hint will pop up to help clarify the purpose of the button.

Creating a Sequence

Creating a sequence is easy. The following example shows you how to create a simple sequence.

1. Click on the 500 position for servo channel 0 and edit the delay for that channel to 500 ms as shown.



2. Click the red circle (Record) to record the frame. Note: clicking the record button on the last frame of a sequence caused the current frame to be copied to the next frame (this helps with animation sequences). If you were to click the record button after editing an existing frame, the current frame would be updated, and the next frame would not be disturbed.
3. Click the number 2500 for channel 0 and click the record button again.
4. Click the number 500 for channel 0 and click the record button again.
5. Click the number 1500 for channel 0, increment the Go to number to 0 and click the record button.

That completes one simple animation sequence. Now you may use the CD-like control buttons to run, stop, and pause your animation sequence.

1. To save your sequence, simply click on Sequence->Save and specify a file name.
2. To retrieve your saved sequence, simply click on Sequence->Open and specify the filename.

Notes concerning the connection of the PSC-USB to your microcontroller:

For the Board of Education, Rev C:

1. The power source for the servos must be a separate power source from that of the Board of Education.
2. Connect the three-pin cable from the PSC-USB to X4, slot 15.
3. Ensure that the Vservo jumper selector, (between X4 and X5), is set to the Vdd position. Connect your servos to the 3-pin terminals provided.
4. To power your PSC-USB, the power switch must be placed to the "2" position.

For all others:

1. The power source for the servos must be a separate power source from that of the 5 VDC used to supply the on-board logic.
2. Connect the three pin cable from the PSC-USB to: VDD (Red), VSS (Black), and the white wire to your choice of I/O pin. VDD must be 5 VDC. Note: the idle state of the serial line is 5 VDC.

Hardware Reset Pushbutton

The pushbutton on the PSC-USB is connected to the reset circuit of the on-board microcontroller. Pushing this pushbutton resets the microcontroller thereby resetting all parameters to their default values.

Hardware/Firmware Revision List

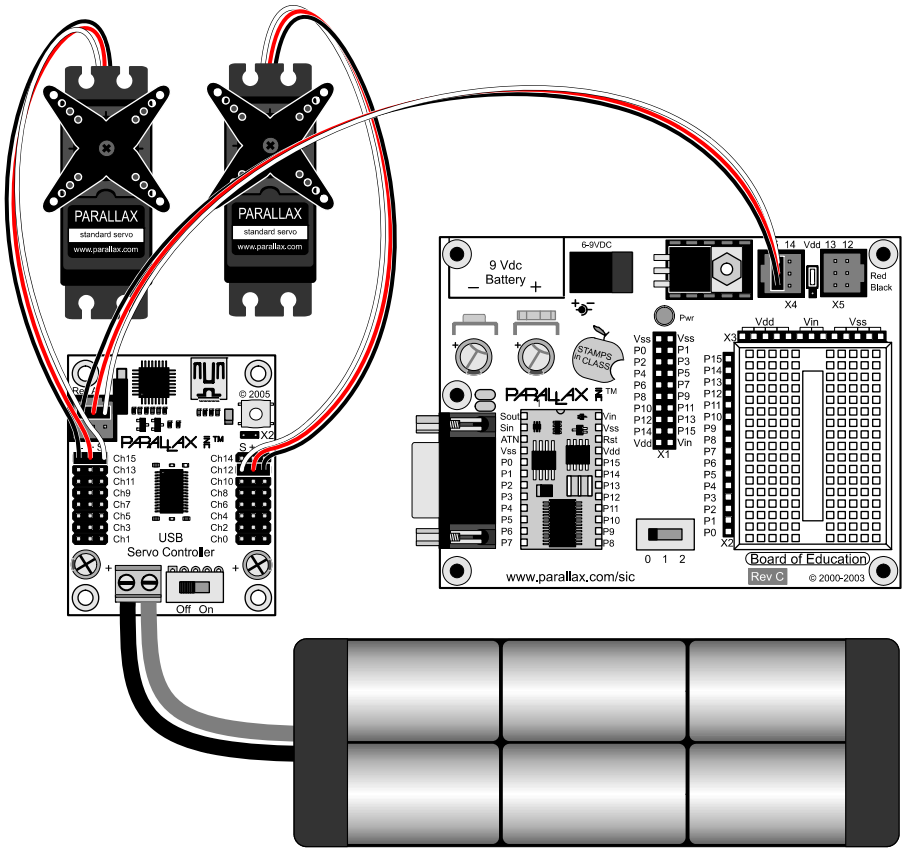
Hardware	Firmware	
Rev A	1.4	(Internal prototype, never released)

No known issues with this version of firmware. Though the PSC – Serial was released with earlier version of firmware, the PSC-USB was only released with v1.4 firmware.

Hardware	Firmware	
Rev B	1.4	(Initial release)

No known issues with this version of firmware. Though the PSC – Serial was released with earlier versions of firmware, the PSC-USB was only released with v1.4 firmware.

Figure 3: Parallax Servo Controller, Typical Connections



Follow the instructions in this document when making connections to the PSC. To power up your PSC, slide the power switch on the Board of Education (BOE) to the "2" position. You should see the red LED illuminate on the PSC as well as the green LED light on the BOE.

Serial Command Format

The PSC supports several commands that are sent to it via RS-232 serial protocol. The voltage swing of this serial line is 0-5 VDC (TTL level). Each serial command must be preceded with an exclamation point, "!", and the pair of letters, "SC".

When your PSC starts up, the default baud rate is 2400 N 8 2, (no parity, eight data bits, two stop bits) true data, open-drain (driven low, pulled high). The exclamation point is used in some AppMods to determine the incoming baud rate, thereby supporting a feature called Auto-Baud. The PSC does not support Auto-Baud. The "SC" portion is an identifier that pertains to the PSC. Together, the "!" and the "SC" form a preamble, "!SC". The preamble serves to distinguish commands for the PSC from other messages on the serial I/O line, and allow different types of AppMods to use the same serial line.

After the preamble is sent, the command and associated parameters are sent. The eighth and final character sent is a $0D, (CR), used to terminate the string. If the command causes the PSC to reply, a three-byte reply is sent after a 1.5 ms delay.

VER? Command – Identify Firmware Version Number

Syntax: "!SCVER?" $0D
Reply: "1.3"

The VER? command causes the PSC to reply with its firmware version number. A string terminator, $0D, must follow each command; note that you may use the constant CR instead. The version number divulged pertains to the version of firmware contained within the IC. The following code snippet can be used to find and identify your PSC.

```
'{$PBASIC 2.5}
Sdat            PIN     15                    ' Serial Data I/O pin
Baud            CON     396                   ' Constant for 2400 baud
buff            VAR     Byte(3)               ' temporary variable

FindPSC:                                      ' Find and get the version
  DEBUG  "Finding PSC", CR                    ' number of the PSC.
  SEROUT Sdat, Baud+$8000, ["!SCVER?",CR]
  SERIN  Sdat, Baud, 500, FindPSC, [STR buff\3]
  DEBUG  "PSC ver: ", buff(0), buff(1), buff(2), CR
  STOP
```

Within the Debug Terminal, you will see a series of "Finding PSC" messages. The BASIC Stamp should find the unit immediately. This is evidenced by the PSC replying with a number like "1.3", which is it's firmware version number. If after 3 or 4 messages the PSC has not been found, you should check that all the connections are proper and that power is applied. If the PSC fails to respond, you may momentarily push the Reset button on the PSC. If the PSC fails to respond, contact Parallax Technical Support.

SBR Command – Set the Baud Rate (to either 2400 or 38K4 Baud)

Syntax: "!SCSBR" x $0D
Reply: "BR" x (where x is either 0 for 2400, or 1 for 38K4)

After establishing communications with the PSC you may wish to elevate the baud rate to 38K4 baud. To do this, you may follow the example in the following code snippet.

```
'{$PBASIC 2.5}
Sdat            PIN     15                    ' Serial Data I/O pin
Baud            CON     396                   ' Constant for 2400 baud
buff            VAR     Byte(3)               ' temporary variable

SetBaud:
  DEBUG "Setting Baudrate", CR
  SEROUT Sdat, Baud+$8000, ["!SCSBR",1,CR]
  SERIN  Sdat, 6,500, SetBaud, [STR buff\3]
  DEBUG "Baud reply:  ", buff(0), buff(1), DEC1 buff(2), CR
  STOP
```

Note that the SERIN command's baud mode is set to 6 (38K4). That's because the PSC will reply to the command with the new baud rate to confirm the command has been executed. If you wished to set the baud rate back to 2400, you must either reset the controller or send it a SBR, $0 command at 38K4 baud

rate. Once the baud rate has been changed, it would be wise to "ping" the servo controller from time to time. If either the BASIC Stamp or the PSC reset without the other resetting, they could be left in the state of different baud rates. When "pinging" the PSC with an VER? command, the PBASIC SERIN function should employ its timeout feature. At this timeout label, you can them attempt to VER? the PSC at the other baud rate. Once identified, the baud rate can again be set, and the program can resume.

Position Command – Set the Position of a Servo Channel

Syntax: "!SC" C R pw.LOWBYTE, pw.HIGHBYTE, $0D
Reply: none

To control a servo, you must write a position command to the PSC. Each position command is comprised of a header, three parameters: C, R, and PW, and a command terminator.
The Header: "!SC" is the header. The header signifies to all devices on the same wire that this is a command for a Servo Controller.

The C parameter is a binary number 0-31 corresponding to the servo channel number. The servo channel number should be 0-15 with no jumper present on the PSC, or 16-31 with the jumper present. With a jumper present on the PSC, servo channel 0 become channel 16, channel 1 becomes 17, etc.

The R parameter is a binary number 0 – 63 that controls the ramp function for each channel. If the ramp parameter is set to 0, ramping is disabled and the pulse width will be set to the P parameter sent immediately. Ramp values of 1-63 correspond to speeds from ¾ of a second up to 60 seconds for a full 500 µs to 2.50 ms excursion for standard servos. This correlation is rather linear though no equation presently exists.

The P parameter is a 16-bit Word that corresponds to the desired servo position. The range, (250-1250), corresponds to 0 to 180 degrees of servo rotation with each step equaling 2 µs.
The command terminator, $0D, (CR), must not be omitted.

The following code snippet demonstrates the Position Command by ramping a servo channel 11 from 0 to 255 at ramp rate 7.

```
'{$PBASIC 2.5}
ch    VAR Byte
pw    VAR Word
ra    VAR Byte
Sdat  CON 15
baud  CON 396

  ra = 7
  ch = 15
DO
  pw = 1250
  SEROUT Sdat, Baud+$8000,["!SC", ch, ra, pw.LOWBYTE, pw.HIGHBYTE, CR]
  PAUSE 1000
  pw = 250
  SEROUT Sdat, Baud+$8000,["!SC", ch, ra, pw.LOWBYTE, pw.HIGHBYTE, CR]
  PAUSE 1000
LOOP
```

Note: Not all servos are exactly alike. If your servos appear to strain when commanded to position 250, you should increase the 250 up to 260 (or so) to prevent the servo from straining itself. Similarly, if your servo strains when commanded to go to position 1250, you should decrease the 1250 to 1240 or so to prevent the servo from straining itself.

RSP Command – Report the Position of a Servo Channel

Syntax: "!SCRSP" x $0D
Reply: x y z(where x is the channel number, and z:y is the value reported)

Sometimes it is necessary to know the current servo position. The RSP command returns the value of the specified servo channel. If the servo is still moving as a result of a ramp command, the position may still be read. The following code snippet demonstrates how to use this command.

```
'{$PBASIC 2.5}
ch    VAR Byte
pw    VAR Word
ra    VAR Byte
x     VAR Byte
Buff  VAR Byte(3)
Sdat  CON 15
baud  CON 396

Init:
  ra =   15: ch = 12
DO
  pw = 1240: GOSUB WRservo
  pw =  240: GOSUB WRservo
LOOP

WRservo:
  SEROUT Sdat, Baud+$8000,["!SC", ch, ra, pw.LOWBYTE, pw.HIGHBYTE, CR]
  FOR x = 0 TO 4
    PAUSE 1000
    SEROUT Sdat, Baud+$8000, ["!SCRSP", ch, CR]
    SERIN  Sdat, Baud, 1000, Init,[STR Buff\3]
    DEBUG "Servo ", DEC buff(0), "  ", HEX2 buff(1), " :", HEX2 buff(2), CR
  NEXT
  RETURN
```

Within the DO-LOOP, this program sets the pulse width (pw) to one extreme or the other and writes this value to the PSC. The ramp value (ra) was set to give the program time to poll the servo position several times. Within the WRservo subroutine, the new pw is sent, and the position is polled five times, once each second. A DEBUG command is used to format the reply and post it to a window for your review.

Liability Disclaimer