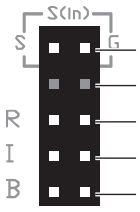
Using the Mini SSC II (Serial Servo Controller)

The Mini SSC II is an electronic module that controls eight pulse-proportional ("hobby") servos according to instructions received serially at 2400 or 9600 baud. The Mini SSC II is addressable, allowing two units to share the same serial line to control a total of 16 servos. This addressability is expandable—you can order units programmed with other ranges of servo addresses so that a total of 32 Mini SSC IIs (controlling up to 255 servos) can share a single serial line.

Configuring the Mini SSC II

You can customize the operation of the Mini SSC II by installing or removing jumpers (shorting blocks) on the configuration header at the upper-right corner of the circuit board as shown in the figure:

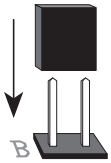


	Jumper		
Function	**off**	**on**	**Description**
Serial in	always	never	*Do not jumper!* See "Connecting the Mini SSC II"
none	—	—	not used
range	90°	180°	Sets positioning precision and range of motion
identification	0–7	8–15	Sets range of servo addresses
baud rate	2400	9600	Sets baud rate (no parity, 8 data bits, 1 stop bit)

The Mini SSC II's default configuration (all jumpers removed) is:

2400 baud • servos 0 through 7 • range of motion = 90°

To change any of these settings, install a jumper block across the appropriate pair of header posts as shown at right. Changes take effect the next time the Mini SSC II is powered up, so you should install or remove jumpers when the Mini SSC II is turned off.



Here are more details on the configuration choices:

(R)ange: With no jumper at R, the Mini SSC II controls servos over a 90-degree range of motion. Servos' positions are expressed in units from 0 to 254, so each unit corresponds to a 0.36-degree change in the servo's position. In some applications, you may want to trade precision for a wider range of motion. With a jumper installed at R, the Mini SSC II controls servos over as much as 180 degrees, with each unit corresponding to a 0.72-degree change in position.

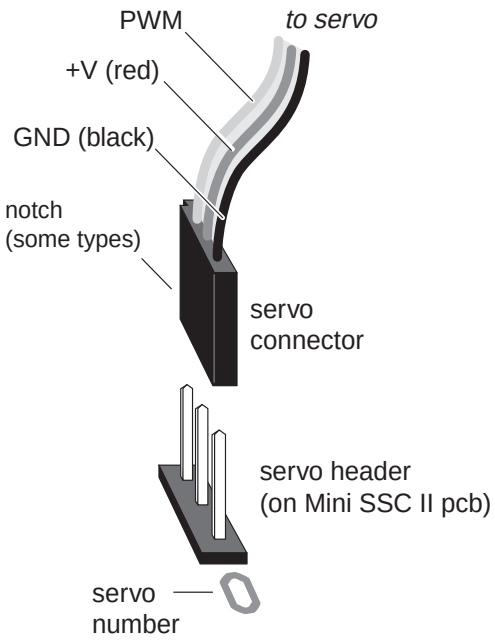
> **Note:** Some servos cannot move through a full 180-degree range. They may stall when position values less than 50 or greater than 200 are used with the Mini SSC II in 180-degree mode. Always check out unfamiliar servos before using them with a Mini SSC II configured for 180-degree motion. Move the servo a few units at a time towards the ends of travel and see whether it stalls. If it does, either use the servo only in 90-degree mode, or restrict the range of position values to the safe range.

(I)dentification: With no jumper at I, servo addresses are the same as the numbers printed below the servo headers, 0 through 7. With a jumper at I, the Mini SSC II adds 8 to these addresses, so that servo 0's address is 8, servo 1 is 9...and servo 7 is 15. This allows you to connect two Mini SSCs to the same serial port and address each of 16 servos individually.

(B)aud: With no jumper at B, the Mini SSC II receives serial data at 2400 baud; with a jumper installed at B, the baud rate is 9600. In both cases, the data should be sent as 8 data bits, no parity, 1 stop bit; abbreviated N81. In addition, the data should be inverted, just the way it comes out of a standard serial port.

Connecting the Mini SSC II

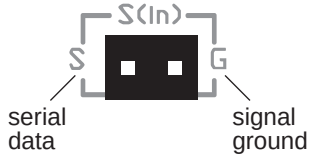
Servos: Servos with standard three-conductor socket connectors (e.g., Futaba-J connector) can be plugged directly onto the three-pin header at the bottom edge of the Mini SSC II board. The drawing shows how the connector should be aligned. If the connector is notched, the notch should line up as shown in the drawing. If the connector isn't notched, use the color code to determine which end is which. Generally the power wires will be red and black—the black wire lines up with the numbered edge of the servo connector on the Mini SSC II board.



Servo Power: Connect power (4.8 to 6Vdc) for the servos to the red (+) and black (–) wires at SVO on the Mini SSC II board. A four pack of alkaline or NiCd C or D cells works fine. For an ac power supply use a regulated linear (not switching) 5-volt supply rated for at least 1 ampere.

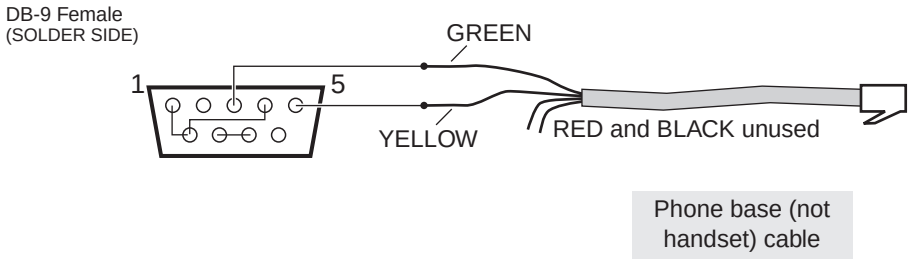
Mini SSC II Power: Connect a 9V battery to the battery snap. If you want to use another power source, cut off the snap and connect 7 to 15Vdc to the wires; + to red, ground (–) to black. *Do not* use the servo power supply for this purpose; the voltage is too low and the varying current demands of the servos make it unreliable for powering the electronics.

Serial Input: The Mini SSC II requires only two connections to a computer—serial data and signal ground. There are two places to make these connections, a modular phone jack and a pair of pins marked S(in) on the configuration header. It doesn't matter which of these you use; pick the one that is most convenient. With single-board computers like the BASIC Stamps (™ Parallax Inc.) you'll



probably use the header pins. Connect the Stamp pin used for serial output to S. Connect Stamp ground (Vss) to G. In the case of the BS2, *do not* use the pin marked Sout; that's used for downloading programs.

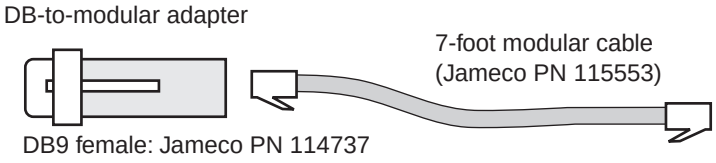
With PCs, you'll probably use the modular jack. Here's how to make a cable to adapt a PC serial port to the modular connector:



Phone base (not handset) cable

The connections between pins (1-9-4 and 7-8) are needed only if the programming language you're using can't turn off serial port handshaking. QBASIC, the language used in the PC example program, doesn't require these connections.

You may also purchase a ready-made cable from the manufacturer, Scott Edwards Electronics. Here's how to make your own with parts from Jameco (1-800-831-4242):



Adapter Setup /DB9

YELLOW:	pin 5
GREEN:	pin 3
RED:	cut off
BLACK:	cut off

The DB9-to-modular adapter comes disassembled. To assemble it, push the pin sockets into the numbered holes in the housing as indicated in the drawing above. Snap the adapter housing together and plug in the modular cable. If you have a different modular cable than the one specified, don't worry. The Mini SSC II's input is wired to accept straight and flipped modular cables. However, avoid using the modular cables that come with inexpensive telephones; these use only the red and green wires, and won't work.

NEVER connect the Mini SSC II to the phone line!

25-pin Serial Ports: If your PC has a 25-pin serial connector, you may use a commercial DB25-to-DB9 adapter to connect to either of the cables shown above.

Initial Checkout

When the Mini SSC II is first powered up, it will turn on the sync LED and move all servos to the centered position (see Theory of Operation). Once you have connected servos, servo power, and Mini SSC II power, the servos will immediately swing to center. If they're already centered, the servos may not move much, so you can confirm correct operation by trying to move a servo with your fingers. It should resist your efforts to move it. If it does not, recheck servo and power connections.

To verify proper serial hookup, run one of the demonstration programs. When correctly formattted serial data is received, the *sync* LED will blink and the received servo-position instruction will be carried out. If servos do not respond, make sure that the configuration is set exactly as specified in the program, and that the computer (BASIC Stamp or PC) is correctly connected.

Programming for the Mini SSC II

To command a servo to a new position requires sending three bytes at the appropriate serial rate (2400 or 9600 baud, depending on the setting of the B jumper; see Configuring the Mini SSC II). These bytes consist of:

Byte 1	Byte 2	Byte 3
<sync marker (255)>	<servo # (0–254)>	<position (0–254)>

These must be sent as individual byte values, *not* as text representations of numbers as you might type at a terminal. The example programs at the back of this manual show how to convert numbers to byte values. In PBASIC, you just omit any text-formatting functions (# for the BS1; DEC, HEX, ?, etc. for the BS2). In other BASICs, use the CHR$ function to convert numbers to bytes.

The sync LED on the Mini SSC II board can help you debug your serial routines. It lights steadily when power is first applied to the board and stays on until the first complete three-byte instruction is received. Thereafter, the LED lights only after a valid sync marker and servo address are received. It stays on until a position byte is received, then turns off. If your program is sending lots of data to the Mini SSC II, the LED will appear to light steadily, but will actually be blinking very rapidly.

Multiple Mini SSCs

To control two Mini SSC IIs, connect them in parallel to the same serial line. Set both units for the same baud rate. Install a jumper at the I configuration header of one; leave this header open on the other. Address the servos as follows:

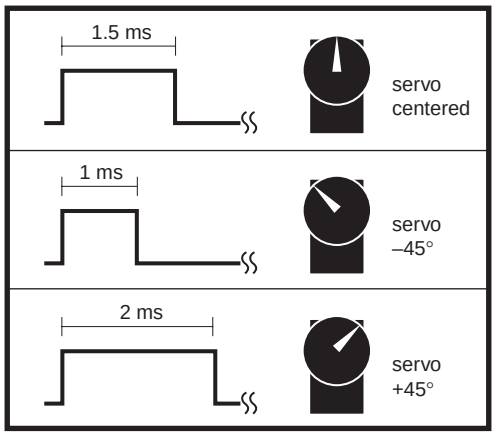
Jumper at I?	Servo Numbers
no	0—7
yes	8—15

With a jumper installed at I, the Mini SSC II adds 8 to the servo addresses, so that servo 0's address is 8, servo 1 is 9...and servo 7 is 15. If you need to control more than 16 servos, you may special-order Mini SSC IIs with higher address ranges from the manufacturer. For example, your third and fourth Mini SSC IIs would have a base address (the address corresponding to output 0 with no I jumper) of 16.

Theory of Operation

Pulse-proportional servos are designed for use in radio-controlled (R/C) cars, boats and planes. They provide precise control for steering, throttle, rudders, etc. using a signal that is easy to transmit and receive. The signal consists of pulses ranging from 1 to 2 milliseconds long, repeated 60 times a second. The servo positions its output shaft in proportion to the width of the pulse, as shown in the drawing at right.

In radio-control applications, a servo generally needs no more than a 90-degree range of motion, since it is usually driving a bellcrank mechanism that can't move more than 90 degrees. So when you send pulses within the manufacturer-specified range of 1 to 2 ms, you get a 90-degree range of motion. With no jumper at the (R)ange header, a position value of 0 corresponds to 1-ms pulses; 254 to 2.016 ms. A 1-unit change in position value produces a 4-microsecond change in pulse width. Positioning resolution is 0.36 degrees/unit (90 degrees/250).



Most servos have more than 90 degrees of mechanical range, though, in order to allow adjustment for component variations, mounting position, etc. The Mini SSC II lets you use this extra range. With a jumper at (R)ange, a position value of 0 corresponds to 0.5-ms pulse; 254 to 2.53 ms. A 1-unit change in position value produces a 8-microsecond change in pulse width. Positioning resolution is 0.72 degrees/unit (180 degrees/250).

Technical Support for the Mini SSC

Contact the manufacturer:

Scott Edwards Electronics, Inc.

PO Box 160

Sierra Vista, AZ 85636-0160

phone: 520-459-4802; fax: 520-459-0623

www.seetron.com

Listing 1. BASIC Stamp I Program Demonstrating the Mini SSC

```
' Program: SCAN.BAS (BS1 servo control demo)
' This program demonstrates servo control using the MiniSSC.
' It commands servo 0 slowly and smoothly through its full
' range of travel. To run this program, leave all configuration
' jumpers off the Mini SSC board. To run the demo, connect:
'            BS1       Mini SSC      Purpose
'           -----     ---------     -------------
'           pin0     S(in) pin S    Serial signal
'           Vss      S(in) pin G    Ground
' Plug a servo into Mini SSC output 0 and connect power as described
' in the manual. Run this program. The servo will slowly scan back
' and forth. Try changing the step values in the for/next loops
' to see the effect on servo movement.

SYMBOL svo = 0          ' Use servo 0.
SYMBOL sync = 255       ' Mini SSC sync byte.
SYMBOL pos = b2         ' Byte variable b2 holds position value.

again:
 for pos = 0 to 254 step 1      ' Rotate clockwise in 1-unit steps.
  serout 0,n2400,(sync,svo,pos) ' Command the mini SSC.
 next pos                       ' Next position.

 for pos = 254 to 0 step -1     ' Rotate counter-clock, 1-unit steps.
  serout 0,n2400,(sync,svo,pos) ' Command the mini SSC.
 next pos                       ' Next position.
goto again                      ' Do it again.
```

Listing 2. BASIC Stamp II Program Demonstrating the Mini SSC

```
' Program: SCAN.BS2 (BS2 servo control demo)
' This program demonstrates servo control using the MiniSSC.
' It commands servo 0 slowly and smoothly through its full
' range of travel. To run this program, install a jumper at
' B on the Mini SSC board; leave all other jumpers off.
' To run the demo, connect:

'            BS2       Mini SSC      Purpose
'           -----     ---------     -------------
'           P0       S(in) pin S    Serial signal
'           Vss      S(in) pin G    Ground

' Plug a servo into Mini SSC output 0 and connect power as described
' in the manual. Run this program. The servo will slowly scan back
' and forth. Try changing the step values in the for/next loops
' to see the effect on servo movement.

svo     con    0       ' Use servo 0.
sync    con    255     ' Mini SSC sync byte.
pos     var    byte    ' Byte variable holds position value.
n96n    con    $4054   ' Baudmode constant for 9600-baud serial.
n24n    con    $418D   ' Baudmode constant for 2400-baud serial

again:
 for pos = 0 to 254 step 1      ' Rotate clockwise in 1-unit steps.
  serout 0,n96n,[sync,svo,pos]  ' Command the mini SSC.
 next                           ' Next position.

 for pos = 254 to 0 step 1      ' Rotate counter-clock, 1-unit steps.
  serout 0,n96n,[sync,svo,pos]  ' Command the mini SSC.
 next                           ' Next position.
goto again                      ' Do it again.
```

Listing 3. QBASIC Program for PCs Demonstrating the Mini SSC

```
DEFINT A-Z
Sync.byte = 255

' The line below assumes that the B jumper is installed for
' 9600-baud operation.
OPEN "com1:9600,N,8,1,CD0,CS0,DS0,OP0" FOR OUTPUT AS #1
CLS
PRINT "                    MINI SERIAL SERVO CONTROLLER"
PRINT : PRINT
PRINT "At the prompt, type the servo number (0 to 7), a comma,
PRINT "and a position value (0 to 254)."
PRINT "Press <CNTL> - <Break> to end."
Again:
  LOCATE 8, 1
  PRINT "                                                "
  LOCATE 8, 1
  INPUT "Servo,position>", Servo, Position
' Perform some basic error trapping
  IF Servo > 7 THEN Servo = 7
  IF Servo < 0 THEN Servo = 0
  IF Position > 254 THEN Position = 254
  IF Position < 0 THEN Position = 0
  PRINT #1, CHR$(Sync.byte); CHR$(Servo); CHR$(Position);
GOTO Again
```