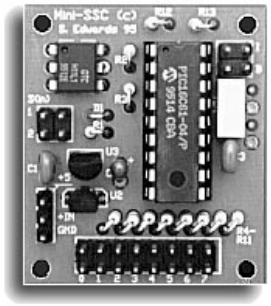
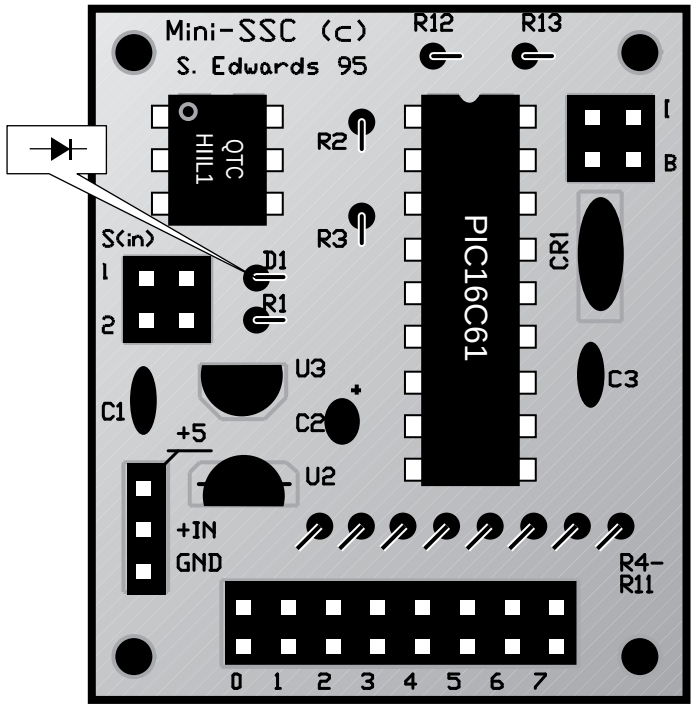
Building and Using the

Mini SSC
(Serial Servo Controller)



Scott Edwards Electronics
PO Box 160
Sierra Vista, AZ 85636-0160 USA
ph: 520-459-4802 • fax: 520-459-0623
e-mail: 72037.2612@compuserve.com

Assembling the Mini SSC



U1—QTC H11L1 Optoisolator (dot at upper left)
U2—LP2950 voltage regulator (flat side as shown)
U3—MN1381 or TC44VC4303 reset circuit (flat side as shown)
U4—PIC16C61 microcontroller (in 18-pin socket; notch at top)
R1—1.8k (brown-gray-red) 1/8W resistor
R2—10k (brown-black-orange) 1/8W resistor
R3–R11—220Ω (red-red-brown) 1/8W resistor
R12, R13—100k (brown-black-yellow) 1/8W resistor
C1,C3—0.1µF monolithic or ceramic capacitor
C2—1µF tantalum capacitor (longer + lead in hole marked +)
D1—1N4148 diode (striped end upward, body on left)
CR1—3-terminal 8MHz ceramic resonator (marked 8.000)
Header stakes—mount the 2x2, 2x8, and 1x3 header strips as shown
9V clip—socket connects to 1x3 header; red to +IN and black to GND
Shunts—shorting jumpers provided for baud rate (B) and address/ID (I); see “Using the Mini SSC” for configuration information.

Using the Mini SSC (Serial Servo Controller)

The Mini SSC is a module that controls eight servos according to instructions received over a 2400- or 9600-baud serial connection. The Mini SSC is serially addressable, so up to 32 units can share one serial line to control 256 servos.

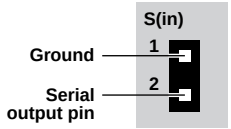
Although the examples below concentrate on the BASIC Stamp, the Mini SSC may be directly connected to any RS-232 data source at 2400 or 9600 baud (no parity, 8 data bits, 1 stop bit). PC-specific hookup instructions appear later in this booklet, but PC users should at least skim the Stamp writeup below in order to get familiar with the Mini SSC.

Configuring the Mini SSC

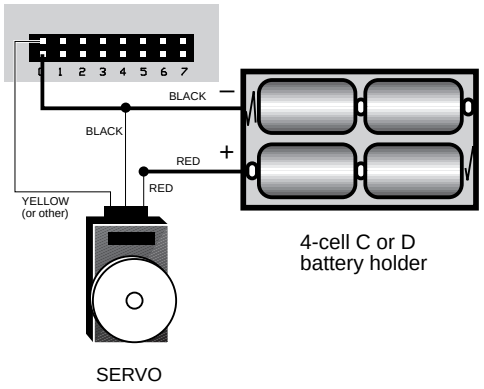
The default configuration of the Mini SSC is 2400-baud communication and a serial address (ID) of 0 (zero). The configuration header is at the upper-right corner of the Mini SSC circuit board. Installing a jumper on the posts marked B sets the unit for 9600-baud communication; a jumper at I changes the ID to 1 (one). The ID number sets the range of servo addresses according to the formula (ID x 8) + servo. For instance, if the ID is 1 (jumper installed), and you want to address the servo connected to output 3, you would send 11 [(1x8) + 3] as the servo number. If your system expands beyond two Mini SSCs, you may order additional units with different IDs.

Connecting the Mini SSC

The BASIC Stamp or other serial device connects to the posts marked S(in) 1 and 2 located on the left edge of the circuit board as shown at right. This connection assumes that the Stamp is set for "inverted" serial data as described in the programming section below, and will also work with other inverted data sources, such as a PC serial port.



The diagram below shows how to connect servos to the Mini SSC. Most servos require 4.8 to 6 Vdc from a source capable of delivering peak currents of several amperes per servo. Four-packs of Nicad or alkaline C or D cells are perfect. The Mini SSC can be powered by its own 9V battery, the Stamp's battery, or any supply that can consistently deliver 6 to 15 volts DC. If you use the Stamp's battery, make sure to connect ground to both the power ground (marked gnd) and pin 1 of the S(in) header.



Note that it's best not to use the same power supply for the servos and Mini SSC. Under load, the servos may draw down the supply voltage. This would cause the Mini SSC to reset, losing servo position data.

Servos come with several different connector styles. The simplest way to make them compatible with the Mini SSC is to remove the existing connector and replace it with header sockets that fit the Mini SSC's 0.025" square posts. See the Hints section for more information.

Programming the Mini SSC

To command a servo to a new position requires sending three bytes at the appropriate serial rate (2400 or 9600 baud, N81). These bytes consist of:

byte 1: `255` **byte 2:** `servo (0-254)` **byte 3:** `position (0-254)`

Note that these must be sent as individual byte values, not as text representations of numbers as you might type at a terminal. Sending numbers as byte values in PBASIC simply requires that you 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; see the PC-specific section. Here's a line of BS1 PBASIC that sends servo 4 to position 50:

```
Serout 0,N2400,(255,4,50)     ' Servo 4 to position 50.
```

The line above assumes that Stamp pin 0 is connected to S(in) 2 as shown on the previous page. Note that the Mini SSC's serial protocol requires the byte 255 to be sent as a marker indicating the start of a servo instruction. Next comes the servo number, followed by the position value (0 through 254).

The servo number and position value may, of course, be determined by variables:

```
Serout 0,N2400,(255,b2,b3)
```

That line will send the servo whose number is stored in the byte variable b2 to the position stored in byte variable b3. Using variables, you can use the Stamp's math, logic, table, and looping operators to determine which servos to send to what positions. For example, here's a pair of For/Next loops that cycle all eight servos back and forth:

```
Again:
  for b3 = 0 to 254              ' Positions 0 to 254.
    for b2 = 0 to 7              ' All 8 servos.
    Serout 0,N2400,(255,b2,b3)   ' Send servo b2 to position b3.
    next b2
  next b3
Goto Again                       ' Endless loop.
```

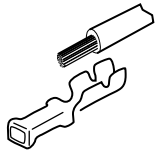
If you run this example, try changing the first for statement by adding different step values, like so:

```
  for b3 = 0 to 254 step 10      ' Positions 0 to 254.
```

The servos will move through the position range much more rapidly than before, in choppy steps. Keep this in mind as you write your own programs for the Mini SSC—you can trade off smoothness and speed to achieve the exact results you need.

Hints on using the Mini SSC

• You can conveniently connect servos and other devices to the Mini SSC's header posts (and the posts on the Stamp and lots of other electronic devices) using crimp-on sockets, as shown at right. The sockets are available from Jameco (phone: 1-800-831-4242 or 1-415-592-8097; fax: 1-415-592-2503) as part number 100765. A tool, part number 99442, makes the crimping job easy. Jameco also carries plastic housings that align the sockets at standard 0.1" intervals, such as part number 100811. An excellent alternative is to cover the individual header sockets with pieces of heat-shrink tubing.



• Futaba J connectors used on many servos already have sockets installed, but they must be removed from their plastic housing. To do this, just pry up the fingers on the side of the housing and slide the sockets out.

• When it is first powered up, the Mini SSC positions all servos to the center of their travel, the equivalent of sending each servo a position value of 127. If the starting positions of the servos are critical in your application, make sure that your program initializes all servos to their correct starting positions as soon as possible after startup. In some cases the starting positions of the servos are really critical (e.g., to prevent damage to a mechanism). In such cases, you should configure the mechanical linkages so that 127 is a safe starting position. If that's not possible, you may want to use a switch or relay to control power to the servos. Turn on servo power after your program initializes the Mini SSC to appropriate starting positions.

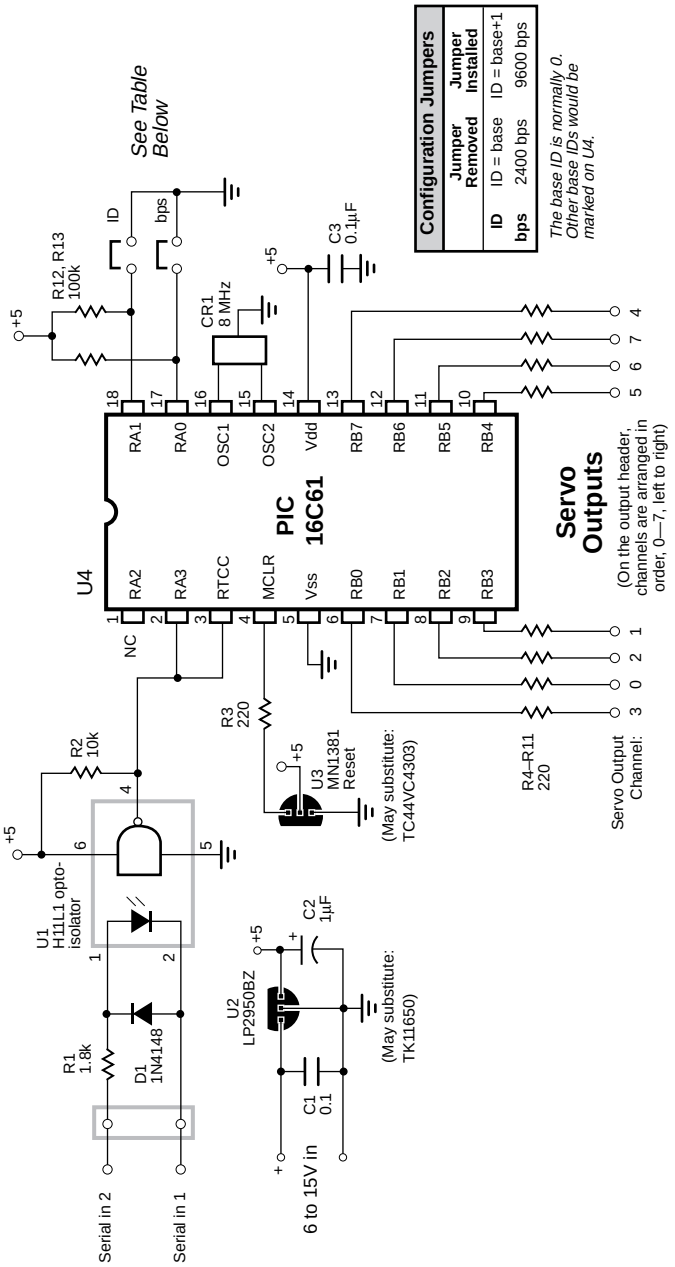
• The Mini SSC reads the baud-rate (B) and ID (I) configuration headers only when it is first powered up. If you change the jumper settings while the Mini SSC is on, the changes won't take effect until the power is turned off and back on.

• Up to nine Mini SSCs may be connected in parallel to a single Stamp pin. If your application requires driving more controllers, either use a separate Stamp I/O pin for each group of nine controllers, or add a buffer capable of driving at least 2 mA per connected Mini SSC.

• With multiple Mini SSCs, IDs can be set so that the servo numbers are sequential. For example, the servos of a Mini SSC with ID 0 are numbered 0 through 7; the servos of ID 1 are numbered 8 through 15. Mini SSC PICs with base IDs of 2 through 30 may be obtained from the manufacturer (address on cover) or by special order through the dealer from whom you purchased this unit.

• The Mini SSC printed circuit board features a top-layer ground plane and plated-through hole construction. As a result, the mounting holes at the four corners of the board are grounded. Any wiring that should not short to ground shouldn't contact metal screws used in these mounting holes either.

• Terminal software generally cannot send instructions to the Mini SSC. The reason is that terminals send numeric values as strings of text. For example, when you type "255" on the screen, it is sent as three bytes; the ASCII character codes for the symbols "2", "5", and "5". The Mini SSC expects a single byte with all its bits set to 1; a byte value of 255. Some terminal programs can send these values, but only through a combination of keys or a special menu.



Listing 1a. BASIC Stamp I (BS1) Demonstration Program

```
' Program: LEGS.BAS
' This program demonstrates servo control using the Mini SSC.
' It commands six servos through a simplified, insectile walking
' sequence. The servo identifiers and positions are stored in a
' pair of tables in the routine Servo_value.

SYMBOL  steps = b0     ' Current place in step sequence.
SYMBOL  servo = b3     ' Servo being addressed.
SYMBOL  position = b4  ' Servo position value.
SYMBOL  temp = b5      ' Temporary variable used in table lookup.
SYMBOL  delay = w5     ' Delay between leg movements; controls speed.

let delay = 1000       ' 1 second between steps.

again:
  for steps = 0 to 13
    gosub servo_value  ' Get the servo no. and position value.
    serout 0,n2400,(255,servo,position)      ' Send to the SSC.
    if bit0 = 0 then skip_delay      ' Delay after every 2nd xmission.
    pause delay
skip_delay:
  next steps
goto again             ' Endless loop.

' Look up the servo number and position value. Since the same data is
' sent to two different servos, the position table is half the length
' of the servo table. Looking up the correct position entry requires
' dividing the step number by 2.
Servo_value:
  lookup steps,(1,4,0,2,1,4,0,2,3,5,1,4,3,5),servo
  let temp = steps/2
  lookup temp,(80,200,170,50,50,80,200),position
return
```

Listing 1b. BASIC Stamp II (BS2) Demonstration Program

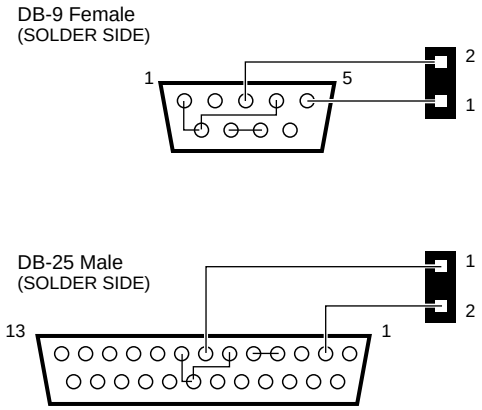
```
' Program: LEGS.BS2
' This program demonstrates servo control using the mini SSC.
' It commands six servos through a simplified, insectile walking
' sequence. The servo identifiers and positions are stored in a
' pair of tables in the routine servo_value.

steps    var   byte    ' Current place in step sequence.
oddStep  var   steps.lowbit   ' Least-significant bit of step no.
servo    var   byte    ' Servo being addressed.
position var   byte    ' Servo position value.
temp     var   byte    ' Temporary variable used in table lookup.
delay    var   word    ' Delay between leg movements; controls speed.
n96n     con   $4054   ' Baudmode for 9600 bps.
n24n     con   $418d   ' Baudmode for 2400 bps.
delay = 1000           ' 1 second between steps.

again:
  for steps = 0 to 13
    gosub servo_value  ' Get servo no. and position value.
    serout 0,n24n,[255,servo,position]       ' Send to SSC at 2400 baud.
    if oddStep = 0 then skip_delay    ' Delay every second xmission.
    pause delay
skip_delay:
  next
goto again             ' Endless loop.

' Look up the servo number and position value. Since the same data is
' sent to two different servos, the position table is half the length
' of the servo table. Looking up the correct position entry requires
' dividing the step number by 2.
Servo_value:
  lookup steps,[1,4,0,2,1,4,0,2,3,5,1,4,3,5],servo
  temp = steps/2
  lookup temp,[80,200,170,50,50,80,200],position
return
```

PC-Specific Instructions

Two Mini SSCs may be connected in parallel to the output of a PC serial (com) port. A simple program (listing 2) is all that's required to command the servos. If your application requires driving more controllers, either use a separate port for each pair of controllers, or add a buffer capable of driving at least 2 mA per connected Mini SSC. The Mini SSC is also compatible with the RS-422 signaling protocol. Connect TD+ to S(in) 1 and TD– to S(in) 2.

Connecting the Mini SSC to a PC

The diagram below shows how to connect the serial output of PC com ports to the Mini SSC. The diagrams include additional connections that loop back the port's handshaking lines. The Mini SSC doesn't use these connections, and some software requires handshaking in order to work properly. (The example QBASIC program in listing 2 disables handshaking when it opens the com port, so the loopback connections aren't required.)



Note: The additional connections shown above are loopbacks for the handshaking lines. These deactivate hardware handshaking required by some software.

Listing 2. QBASIC (PC) Demonstration Program

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

' The line below assumes that the B jumper is installed for
' 9600-baud operation, and that the Mini SSC is connected
' to COM1.


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
```