SSC-12 Ver 2.0

12 Channel Serial Servo Controller
with Independent Variable Speed

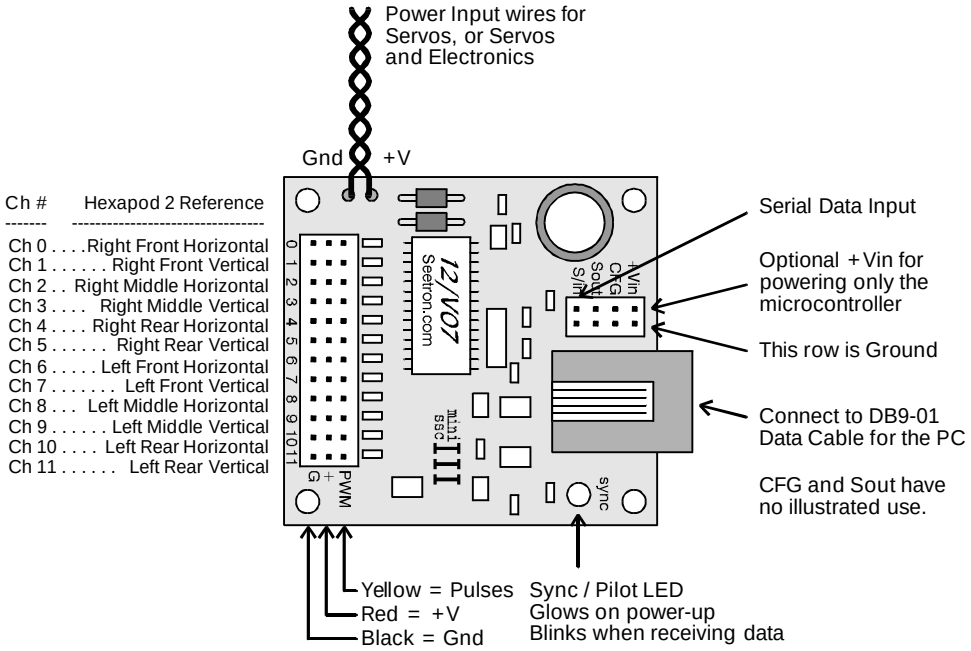
Lynxmotion, Inc.
PO Box 818
Pekin, IL 61555-0818
Tel: 309-382-1816 (Sales)
Tel: 309-382-2760 (Support)
Fax: 309-382-1254
E-m: sales@lynxmotion.com
E-m: tech@lynxmotion.com
Web: http://www.lynxmotion.com

Assembly Manual SSC-12 Ver 2.0

SSC-12 Version 2.0

Caution:

Read the manual completely before wiring and applying power to the board! Errors in wiring can damage the controller board and any attached servos!



Do not reverse polarity, even momentarily. Reversed power will destroy the electronics.
Do not exceed 9.6vdc into the SSC-12. Over voltage may damage the unit or the servos.
The "phone" jack is for serial data. Connecting it to a telephone system will damage the unit.

Note:

The twisted pair wires for power can break off if the wires are bent back and forth. Mount the board and terminate the wires as soon as possible.

Using the SSC-12

Powering Options

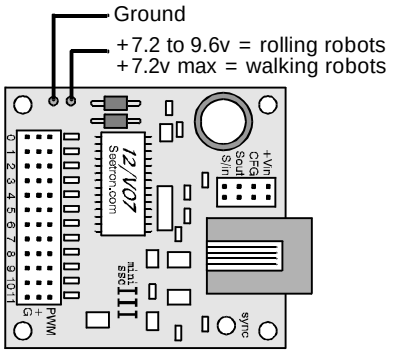
There are two options for powering your servos and servo controllers electronics. The easiest method is to power the servos and the electronics from a common source of 7.2 to 9.6vdc. For small robots this could be a 6 cell pack consisting of "AAA" through "D" alkaline cells, or a 6 to 8 cell pack of NiCad or NiMh cells. This design has proven to be very reliable for robots using standard size servos in pretty demanding applications. Always use fresh batteries. Old and tired batteries will cause problems with the operation of your robot.

In some instances it may be necessary to power the servo motors separately from the servo controllers power supplies. Some micro size servos can't be powered from voltages over 4.8vdc. Also, large servos can be power hogs and may require a separate supply to prevent power dips from effecting the microcontroller. For these special cases you need to wire the +Vin terminal posts to a 7.2 to 9.6vdc supply, such as a standard 9v battery, and add the appropriate size battery for the servos.

Caution

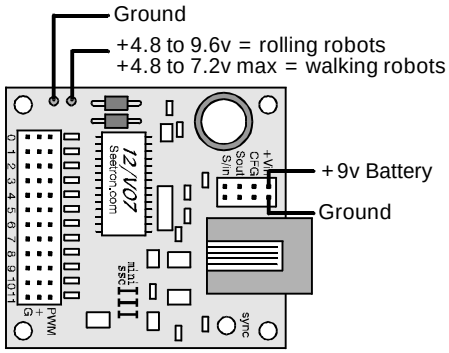
Be aware that some wiring plans apply the full 7.2 to 9.6vdc to the servos, which are rated for 4.8 to 6vdc. The servo is an analog device, which means there is a certain amount of tolerance inherent to the design. Lynxmotion routinely runs servos for rolling platforms at 7.2 to 9.6vdc and servos for walking robots at 7.2vdc, and we have never had a problem. However, run this mode at your own risk. Lynxmotion will not replace servos that were damaged, or appear to have been damaged, from over voltage. Remember that connecting the power backwards will result in damage to the circuitry or the servos. Check it twice, check it again, then apply the power.

Normal Mode



This method of powering is a good general purpose scheme.

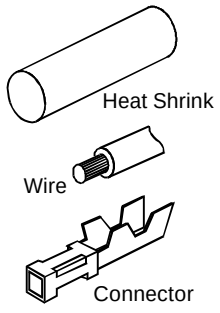
Dual Supply Mode



This method of powering provides noise isolation by powering the micro separately from the servos.

Making the Connections

Do not directly solder wires into place, this does not facilitate easy changes and modifications which are common in robot experimentation, and will ruin the header posts. The best way to make the physical connections are to use the header posts on the Next Step micro and other devices, using small removable jumpers. You can make your own jumpers using crimp on sockets, as shown. Lynxmotion also sells a Connector Kit part number CK-01. This kit includes a crimping tool, 50 female crimp pins, 6 assorted colors of 24 gauge wire and 3/32" heat shrink. You can make up to 25 jumpers with this accessory kit. That's more than enough for a small robot project. These jumpers are easy to plug in and remove to facilitate quick changes and ease troubleshooting.



Using the SSC-12

Configuring the SSC-12

Users of the SSC-01 are used to configuring several parameters of the device with shorting jumpers. The serial data rate, amount of servo output shaft throw and the addressing of the servo channels are adjustable on the SSC-01. Because the SSC-12 was designed specifically for the Hexapod 2 these parameters are not adjustable, nor do they need to be. The serial data rate is fixed at 9600 baud. The servo output shaft throw is fixed at 180° and due to the method of communicating the servos speed parameter, addressing is no longer available.

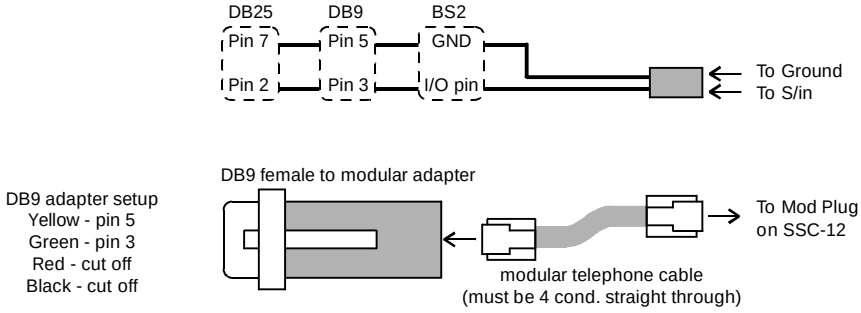
Note Regarding Range:

Servos are designed to allow 90° of rotation, however due to mechanical and electrical tolerances, more range is available. Be aware that some servos cannot move through a 180° range, therefore care should be taken to avoid stalling and potentially damaging a servo.

Connecting the SSC-12

The SSC-12 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 header pins marked S/in on the board. For use with a PC it is far easier to use our DB9-01 cable. For use with our Next Step and a BASIC Stamp 2 use the header pins. Connect the I/O pin that your program will use for serial output to S/in and connect ground to the Ground row on the SSC-12 board. Lynxmotion carries a connector kit, part number CK-01, that makes it easy to route these signals.

The figure below shows how to wire an SSC-12 cable for use with a DB9 connector found on PCs and other computers. If you prefer not to make your own cable, a ready made cable is available from Lynxmotion Inc. as part number DB9-01.



Initial Checkout

When the SSC-12 is first powered up, it will turn on the Sync LED and move all of the servos to the centered position (see Theory of Operation). Once you have connected servos and power, the servos will immediately swing to center. If they are already centered, they may not move much, but you can confirm proper operation by trying to move the servos output shaft by hand (gently). The servo will resist your efforts to move it.

At powerup, the green Sync LED indicates not only that the SSC-12 is receiving power, but also that its microcontroller is working properly (passed startup initialization). If the green light comes on, but the servos don't appear to be responding correctly, the problem is most likely with the servo connections, or serial hookup, *not* with the SSC-12.

To verify proper serial hookup, run the demonstration program listed below. Make sure the programs baud rate is set for 9600, and that the I/O pin or comm port selected for serial output is the one the SSC-12 is connected to.

Programming for the SSC-12

To command a servo to a new position requires sending three serial bytes. These bytes consist of:

Byte 1	**Byte 2**	**Byte 3**
[sync marker (255)]	[servo # (0-11)]	[position (1-254)]

These must be sent as individual byte values, not as text representations of numbers as you might type at a keyboard. In PBASIC, you just omit any text formatting functions (DEC, HEX, ?, etc. for the BS2). In other BASICs, use the CHR$ function to convert numbers to bytes.

The Sync LED on the SSC-12 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 instruction is received. If your program is sending lots of data to the SSC-12, the LED will appear to light steadily, but will actually be blinking very rapidly.

A 32-bit Windows DLL is available free from Scott Edwards Electronics web site www.seetron.com/SSC_an1.htm. Documentation includes a complete description of the DLL, plus program examples for Microsoft Visual BASIC 6 and Borland Delphi 5. The DLL may be used with any Windows programming environment that supports DLLs. This was written for the original SSC-01, but the SSC-12 uses the same serial configuration so this should work well.

The SSC-12 uses bitwise math to receive the servo speed value. The four-bit speed factor is tucked into the upper four bits of the servo number. Speed is expressed in 1/2 unit per frame with a frame rate of 50 frames per second, see figure 3. A speed position of "0" is a special case. It means move immediately to the commanded position without delay. This feature allows the SSC-12 to accept code written for the original SSC-01.Depending on how comfortable you are with bitwise math, it may be easier to think of the speed factor in terms of a decimal number that you add to the servo number. Just multiply the desired speed value by 32 and add it to the servo number. For example, to run servo 3 at a speed of 4 units per frame to position 100, your instruction would be as follows. [255, (32 x 4 + 3), 100]

Naturally you don't want to do this math in your program, better to define constants with the speed factors you want to use. The figure below shows the speed value and its associated unit/frame value and approximate time to move a servo from stop to stop, or full range.

0 = as fast as possible (depends on servo)	128 = 4.0 unit/frame (1.27 sec full range)
16 = 0.5 unit/frame (10.16 sec full range)	144 = 4.5 unit/frame (1.13 sec full range)
32 = 1.0 unit/frame (5.08 sec full range)	160 = 5.0 unit/frame (1.02 sec full range)
48 = 1.5 unit/frame (3.39 sec full range)	176 = 5.5 unit/frame (0.92 sec full range)
64 = 2.0 unit/frame (2.55 sec full range)	192 = 6.0 unit/frame (0.85 sec full range)
80 = 2.5 unit/frame (2.03 sec full range)	208 = 6.5 unit/frame (0.78 sec full range)
96 = 3.0 unit/frame (1.69 sec full range)	224 = 7.0 unit/frame (0.73 sec full range)
112 = 3.5 unit/frame (1.45 sec full range)	240 = 7.5 unit/frame (0.68 sec full range)

To further simplify your instruction would look like the following. [255, (128 + 3), 100]

What is exciting is the servos can move at different speeds! In order to move servo 3 quickly to position 100, while at the same time move servo 5 slowly to position 50 you simply send two serial instructions. [255, 3, 100] [255, (16 + 5), 50] Because these instructions are sent so quickly at 9600 baud, the motion is virtually instantaneous. Even updating the position of 12 servos has the illusion of synchronized motion.

The only exception to the SSC-12 being able to run SSC-01 programs is that the SSC-12 uses a servo position value of 0 as a method of removing the servo pulses on that channel. So sending the following instruction causes servo 3 to no longer receive pulses. [255, 3, 0] Sending another instruction with a non zero value to that channel will restore the pulses.

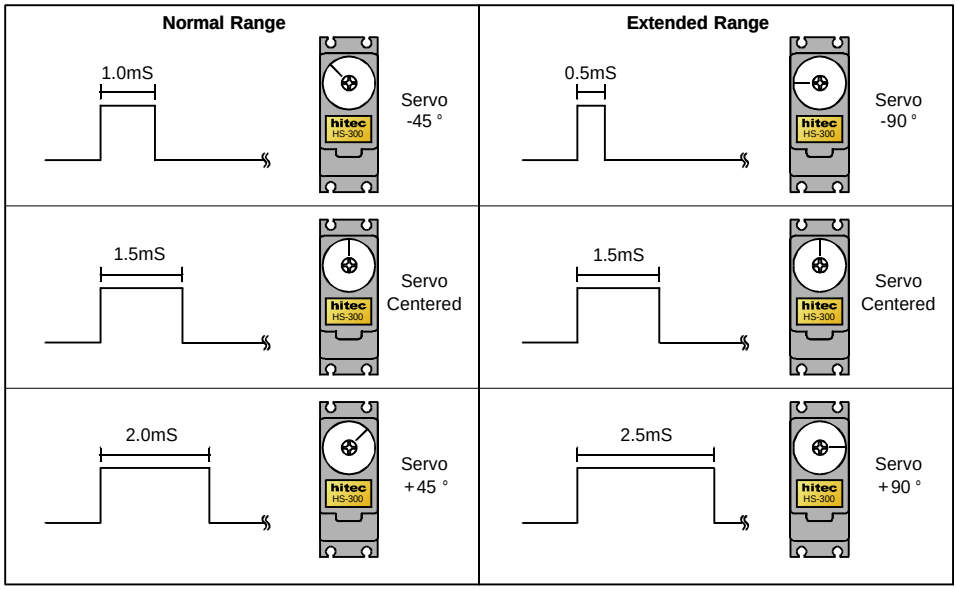
Theory of Operation

Powering Options

Pulse-proportional servos are designed for use in radio-controlled (R/C) cars, boats and planes. They provide precise control for steering, throttle, rudder, 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 below.

In radio-control applications, a servo needs no more than a 90° range of motion, since it is usually driving a crank mechanism that can't move more than 90°. So when you send pulses within the manufacturer-specified range of 1 to 2 mS, you get a 90° range of motion.

Most servos have more than 90° of mechanical range, though, in order to allow adjustment for component variations, mounting positions, etc. The SSC-12 lets you use this extra range. A position value of 0 corresponds to 0.5mS pulse, and a position value of 254 corresponds to a 2.5mS pulse. A one unit change in position value produces a 8-microsecond change in pulse width. Positioning resolution is 0.72°/unit (180°/250).



Remember that some servos may not be able to move the entire 180° range. Use care when testing servos. Move to the extreme left or right slowly, looking for a point when additional positioning values no longer result in additional servo output shaft movement. When this value is found, put it as a limit in your program to prevent damaging the servo.

Programming Examples for the SSC-12

Note: The SSC-12 operates at 9600 baud. Therefore the Basic Stamp 1 is not supported.

```
' Listing 1. BASIC Stamp II Program Demonstrating the SSC-12
' Program SCAN.BS2 (BS2 Servo control demo)
' This program demonstrates servo control using the SSC-12.
' It commands servo 0 slowly and smoothly through its full
' range of travel, then moves it quickly through 90° of travel.
' Connect:
'           BS2  SSC-12  Purpose
'           ---  ------  -------------
'           P8   S pin   Serial signal
'           GND  G pin   Ground
' Plug a servo into SSC-12 output 0 and connect power as
' described in the manual. Run this program. The servo will
' slowly scan back and forth, then move quickly through 90° of
' travel.
svo con 0        ' Use servo 0.
sync con 255     ' Sync byte.
pos1 con 63      ' Holds -45° position value.
pos2 con 191     ' Holds +45° position value.
speed con 32     ' Holds the speed value.
n96n con $4054   ' Baudmode: 9600 baud (BS2-SX, change to $40F0).
again:
  serout 8,n96n, [sync, svo + speed, pos1]
  pause 3000     ' Wait 3 seconds.
  serout 8,n96n, [sync, svo + speed, pos2]
  pause 3000
  serout 8,n96n, [sync, svo, pos1]
  pause 500
  serout 8,n96n, [sync, svo, pos2]
  pause 500
goto again
```