Serial 8-Servo Controller

User's Guide

Contents:

http://www.pololu.com/

Important Safety Warning

This kit is not intended for young children! Assembly of this kit requires high-temperature soldering and the use of sharp cutting tools. Some included components may become hot, leak, or explode if used improperly. Pololu strongly recommends that you wear safety glasses when building or working with ***any*** electronic equipment. Children should use this kit only under adult supervision. **By using this product, you agree not to hold Pololu liable for any injury or damage related to the use or to the performance of this product. This product is not designed for, and should not be used in, applications where the malfunction of the product could cause injury or damage.**

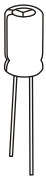
Contacting Pololu

You can check the Pololu web site at `http://www.pololu.com/` for latest information about the servo controller, including color pictures, application examples, and troubleshooting tips.

We would be delighted to hear from you about your project and about your experience with our servo controller. You can contact us through our online feedback form or by email at `support@pololu.com`. Tell us what we did well, what we could improve, what you would like to see in the future, or anything else you would like to say!

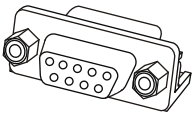
Parts List

The following components are the servo controller kit parts. Make sure to verify that all components are included, and that you know which component is which. For each component, the reference number, description, and quantity are indicated. There are 24 parts in the kit, including the PCB. (The DB9 connector comes snapped into the PCB; remove it by pushing the PCB down on a flat surface before beginning assembly.)



C1
Electrolytic
Capacitor
x 1

C2
Ceramic
Capacitor
x 1

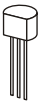


CON3
Female DB9
Connector
x 1

D1
Yellow
LED
x 1

D2
Green
LED
x 1

D3
Red
LED
x 1

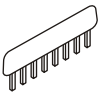


Q1
2N3904
Transistor
x 1

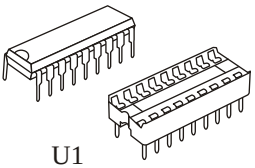
R1, R3
10 kOhm
Resistor
x 2

R2
4.7 kOhm
Resistor
x 1

R4-R7
1 kOhm
Resistor
x 4



RP1,RP2
8-pin
Resistor Pack
x 2

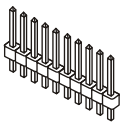


U1
18-pin IC
and Socket
x 1

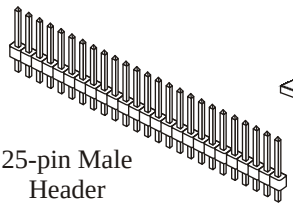
U2
LM2931 5-volt
Regulator
x 1

Y1
8 Mhz
Resonator
x 1

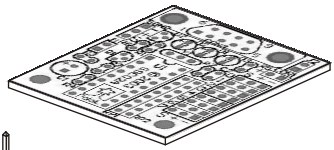
Shorting
Block
x 1



25-pin Male
Header
x 1



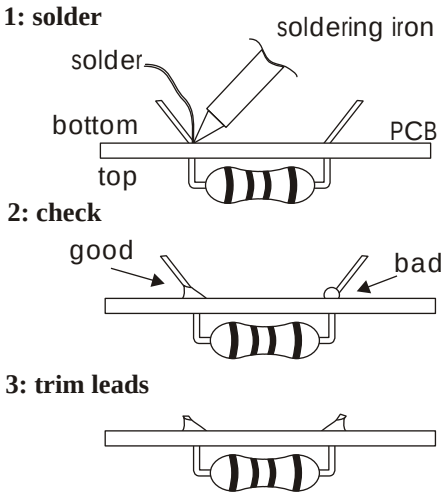
25-pin Male
Header
x 1



Printed Circuit
Board (PCB)
x 1

How to Solder

You need a soldering iron and diagonal cutters to assemble the servo controller. The green printed circuit board (PCB) is the base that holds the components together and establishes the necessary electrical connections. The PCB has two sides: a top side, or *component side*, which has white silkscreen markings, and a bottom side, or *solder side*. Insert the components from the top side and solder them on the solder side. In general, you should insert and solder the components so that they are as close as possible to the PCB. After soldering, trim excess leads with diagonal cutters.



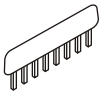
To solder, heat a component lead and the PCB pad and then apply solder until the solder flows onto both the lead and the pad. If the solder beads up on the lead or on the pad, the connection is bad, so you should apply more heat. However, be careful not to damage any components through overheating. A good connection should be nice and shiny.

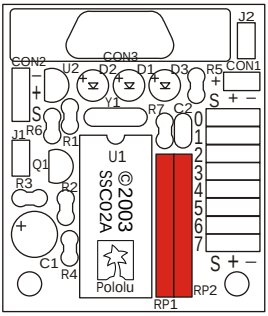
Assembly Instructions

Caution: The components U1 and U2 can be damaged by static electricity. Ground yourself (touch a water pipe or the metal frame of a piece of electrical equipment) before handling these components, and avoid touching their leads. Once assembled, the device can be stored safely in the pink conductive bag in which the kit is packaged.

If you have not already done so, remove the DB9 connector that comes snapped into the PCB. You can easily remove the connector by placing the PCB on a flat surface and pushing down to pop the connector out.

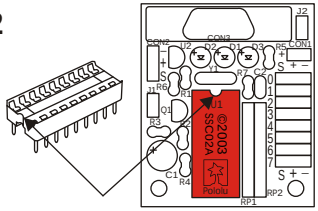
1





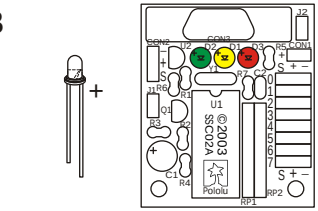
Solder the two 8-pin resistor packs into the locations indicated on the drawing to the left. The resistor packs are symmetrical, so it doesn't matter which way you insert them. Each resistor pack contains four individual resistors.

2



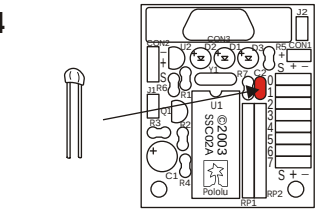
Next, add the 18-pin socket in the area marked U1. **The socket has a notch on one side, which you should align with the notch on the PCB drawing.** The socket protects the PIC microcontroller from being damaged during soldering and allows the PIC to be removed.

3



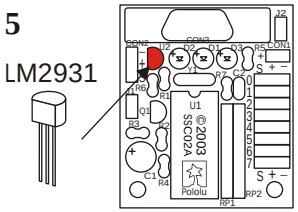
Solder in indicator LEDs D1-D3. **LEDs are polarized, so you must insert them properly.** The longer lead must be on the side indicated by a '+' on the PCB silkscreen. When the PCB is oriented as in the diagram to the left, the left LED is green, the middle is yellow, and the right one is red.

4



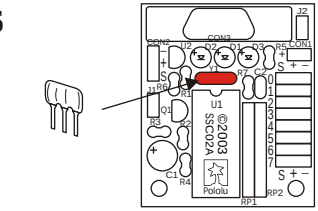
Solder in capacitor C2. It is not polarized, so the orientation in which you insert it does not matter. You may need to reshape the leads to fit the PCB hole spacing. The capacitor helps keep the 5V power supply for the board clean and stable.

5



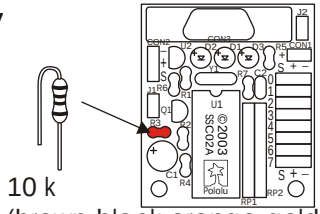
Add U2, the 5-volt voltage regulator. **The regulator looks just like transistor Q1, so make sure you verify the 'LM2931' label on the package.** Bend the middle lead back, and **make sure the device is oriented as indicated on the silkscreen drawing, with the flat side facing out from the PCB.** The voltage regulator takes the servo controller input voltage and provides the microcontroller with a necessary 5V supply.

6



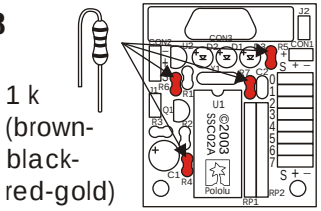
Now, mount the 8 MHz resonator, Y1. The resonator is a symmetrical device that can be soldered in two orientations. It is generally good practice to mount components so that their markings are as visible as possible.

7



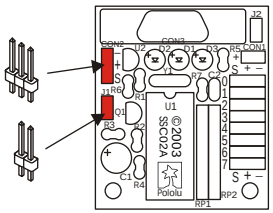
Insert resistor R3 by bending it into the shape shown to the left. The colors of the 10 kOhm resistor are **brown-black-orange**, with a gold band on the end. The resistor is used with the shorting block to put the servo controller in one of two operating modes (see the "Using the Servo Controller" section).

8



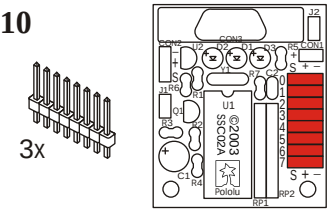
Insert resistors R4-R7 by bending them into the shape shown to the left. The colors of the 1 kOhm resistors are **brown-black-red**, with a gold band on the end. Resistors R5-R7 set the LED brightnesses, and R4 is used in the microcontroller (U1) reset circuit.

9



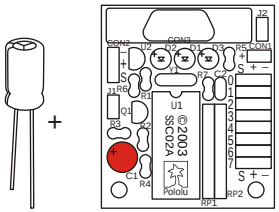
Using diagonal cutters or two sets of pliers, snap off 2-pin and 3-pin sections of male header **from the 10-pin strip**, and solder them into locations J1 and CON2. You can also solder in wires directly or use your own connector for CON2, which supplies logic power and a TTL-level serial input to the servo controller.

10



Snap off three 8-pin header sections **from the 25-pin strip** to use as servo connectors. It is crucial to get these pins parallel and perpendicular to the PCB so that your servo connectors can plug in. If you have a servo or two available, you should plug them in while you solder the header pins to guarantee proper alignment of the pins.

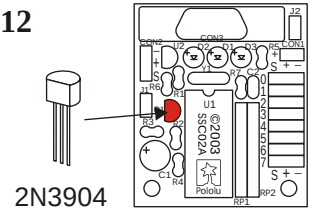
11



Solder in capacitor C1, which is a 100 microfarad electrolytic capacitor. **The capacitor is polarized: the longer lead is positive, and the cylinder is marked with a stripe on the negative side.** The hole closer to connector CON3 is the positive hole.

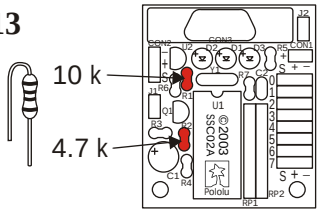
Instructions 12-14 are for components necessary for connecting the servo controller to a PC serial port. If you are not going to use this feature and are instead going to use 5V serial signals from a robot controller board or other source, you can skip to step 15. You can come back and add the components at a later time if you change your mind.

12



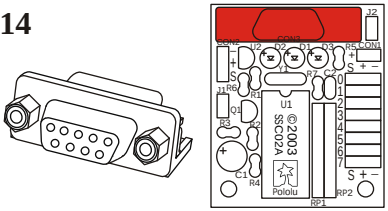
Add NPN transistor Q1, which is labeled '2N3904'. It looks just like the voltage regulator, U2, but if you got the voltage regulator right, the only remaining 3-pin part is Q1. Bend the middle pin back a bit, and make sure the flat side faces out, as indicated on the PCB silkscreen drawing.

13



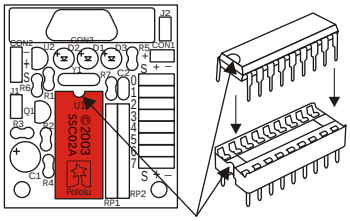
Insert resistors R1 and R2 after bending them into the shape shown to the left. The colors for the 10 kOhm R1 are **brown-black-orange**, with a gold band on the end. The 4.7 kOhm R2 has colors **yellow-purple-red**, again with gold at the end.

14



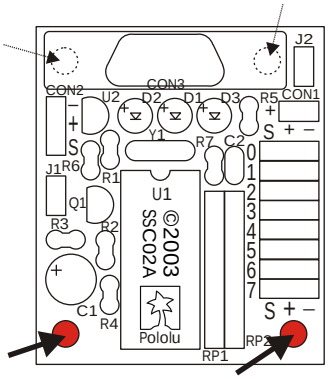
Insert the DB9 connector into the space marked CON3. Make sure to solder all pins, including the large pins on the sides of the connector, to prevent the connector from breaking loose when a serial cable is attached.

15



Finally, insert the PIC microcontroller into its socket. **The package has a notch that you must match up with the notch on the PCB silkscreen.**

Mounting the Servo Controller

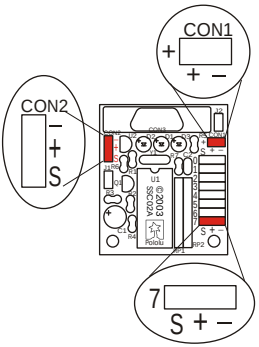


The servo controller has two 1/8" mounting holes that are indicated to the right. You can bolt or screw your servo controller to your target platform, or you can stick on rubber feet to the back of the PCB in the unpopulated areas around the holes. If you are not using the DB9 serial port connector (CON3), you can also use the two holes on the sides of the connector footprint (hole pattern).

To prevent any accidents, we strongly recommend that you mount your servo controller securely. The servo connectors are connected directly to the servo power source, and shorting the adjacent '+' and '-' pins can destroy your servo controller and power supply. Avoid having any of the leads on the bottom side of the PCB touching the surface to which you are mounting the servo controller, or make sure the surface is not electrically conductive.

Connecting the Servo Controller

Connect the power supply for your servos to **CON1**, making sure to observe the '+' and '-' markings on the silkscreen. The servo supply should typically be 4.8-6.0 volts, and it should be capable of providing several amps of current. If many servos will be straining simultaneously, the total current can be close to 10 amps. In such applications, you should use the servo controller PCB just for signals and separately wire up the power and ground pins of the servos individually.

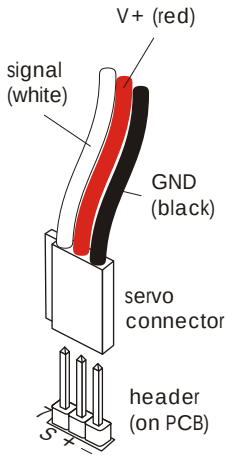


CON2 is the connection point for 5.6-25 volts for running the servo controller electronics. Make sure to observe the correct polarity. The pin labeled 'S' is the TTL-level serial input.

CON3 can be used to connect the servo controller directly to a computer's serial port using a straight-through (not null-modem) cable. **Do not attempt to send serial commands through both CON2 and CON3 at the same time**.

It is possible to power both the servos and the servo controller off of one power supply, using **CON1**. Jumper **J2** connects the servo power supply to the regulated supply of the PIC microcontroller. If your servo supply is in the 4.0-5.5 volt range (which would be the case for a 4-cell, 4.8-volt NiCd or NiMH battery pack), you can solder a small piece of wire between the two pins of J2. **Do not apply power to CON2 if J2 is installed.**

If you are connecting multiple devices to the same serial line, you can connect all the 'S' inputs together. If you want to use CON3 to connect to the servo controllers from a PC, use one serial cable and connect to one of your servo controllers' CON3 connectors. The 'S' pin on that servo controller will then output the TTL-level signal to the 'S' pins on the other controller boards.

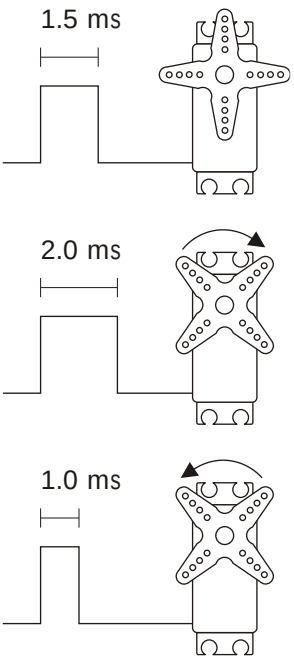


When connecting servos, be careful because it is possible to plug in a servo backwards. **Make sure to connect your servos correctly, or they may be destroyed**. The signal (usually white or yellow) wire should be closest to the PIC microcontroller, and the black wire should be closest to the edge of the board. Some servo connectors have a polarizing nub that indicates the signal lead; that notch should be above the servo number silkscreen on the PCB.

How Servos and the Servo Controller Work

Radio Control (RC) hobby servos are small actuators designed for remotely operating model vehicles such as cars, airplanes, and boats. A servo might typically move the control surface of an airplane or the steering mechanism in a car. A servo contains a small motor and gearbox to do the work, a potentiometer to measure the position of the output gear, and an electronic circuit that controls the motor to make the output gear move to the desired position. Because all of these components are packaged into a compact, low-cost unit, servos are great actuators for robots.

An RC servo has three leads: two for power and ground, and a third for a control signal input. The control signal is a continuous stream of pulses that are 1 to 2 milliseconds long, repeated approximately fifty times per second, as shown below. The width of the pulses determines the position to which the servo moves. The servo moves to its neutral, or middle, position when the signal pulse width is 1.5 ms. As the pulse gets wider, the servo turns one way; if the pulse gets shorter, the servo moves the other way. Typically, a servo will move approximately 90 degrees for a 1 ms change in pulse width, but the exact correspondence between pulse width and servo position varies from one servo manufacturer to another.



The Pololu servo controller performs the processor-intensive task of simultaneously generating 8 independent servo control signals. The servo controller can generate pulses from 0.25 ms to 2.75 ms, which is greater than the range of most servos, and which allows for a servo operating range of over 180 degrees.

Internally, the servo controller maintains a servo position value that is two times the pulse width, measured in microseconds. Thus, the 1.5 ms neutral position, which is 1500 microseconds long, is represented internally as 3000. The internal values thus range from 500 to 5500. Various interface modes allow the user to set the position value for each servo in multiple ways, which are described in depth in the "Using the Servo Controller" section.

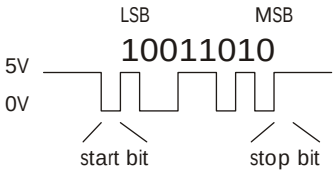
Using the Servo Controller

Initial Test

The first time you turn on the servo controller, do so without any servos attached, without the servo power supply connected or turned on, and without a serial input. Before connecting power, temporarily connect a wire from the serial input to ground (if you can't conveniently access the negative terminal of your power source, you can connect to one of the servo ground pins). Now, power up the servo controller. All four of the indicator LEDs should turn on. If they do, you can disconnect your serial line from ground and connect the real serial input. If one or more of the LEDs do not turn on, something is wrong with your servo controller. Check over your servo controller and the assembly instructions to make sure you haven't made any mistakes, and consult the "Troubleshooting Tips" section of this guide. Without the LEDs working, it will be very difficult to diagnose any problems you may encounter later.

Serial Input

The serial commands sent to the servo controller must be sent eight bits at a time, with no parity and one stop bit (sometimes abbreviated 8N1). The serial input on CON2 must be *non-inverted*, meaning that a zero is sent as a low voltage, and a one is sent as a high voltage, as shown in the diagram to the right. (The circuit supporting the DB9 connector (CON3) inverts the signal from a PC serial port, as required.) *Commands sent to the serial input **must** conform to the above format and use the appropriate protocol (described in detail later) or else the servo controller and other devices connected to the serial line may behave unexpectedly.*



When you turn on power with your serial input connected, you should either see all of the LEDs or just the yellow LED turn on. After the servo controller turns on and determines the communication mode (see below), it waits for a serial input to determine the baud rate. If the input line is low, it turns all LEDs on and waits for the line to go high, which is the idle state for the serial line. Once the line is detected to be high, only the yellow LED is turned on, and the servo controller waits for a serial input. If the detected baud rate is too high, the red LED will turn on and the green LED will flash quickly. If the serial rate is too slow, the red LED will turn on and the yellow LED will flash. From this point on, the servo controller behavior depends on the communication mode. **Once you choose a baud rate, all subsequent transmissions must be at that same baud rate.**

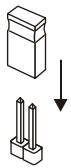
Indicator LEDs

The green LED indicates serial activity: it should flicker whenever the PIC receives data. The yellow LED indicates a warning regarding position: either the absolute or neutral position you have requested is out of range, or a combination of neutral, range, and 7-bit or 8-bit position caused the internal position variable to go out of range. The position will just be limited to the max or min, and the yellow LED will go out when all requested positions are in range. The red LED indicates a fatal error that prevents further operation (for example, a fatal error could be caused by invalid serial input).

Interface Options
You can communicate with the servo controller using one of two communication protocols. One of the two interface modes is chosen based on the state of jumper J1 when the servo controller is powered up; you cannot change modes without resetting the servo controller by turning it off and then on.

Pololu Mode: The default mode, when jumper J1 is open (no shorting block), is a Pololu protocol used for controlling multiple serial devices. In this mode, the servo controller can be on the same serial line as other devices such as our Dual Serial Motor Controller. This mode also allows access to all of the special features of the servo controller, such as setting speeds, ranges, and neutral settings.

Mini SSC II Mode: This mode is set by placing the shorting block over the two J1 pins. This setting allows the servo controller to respond to the protocol used by the Mini SSC II servo controller made by Scott Edwards Electronics. This protocol is more simple, but it only allows the user to specify the desired servo positions in only one way. In this mode, the servo controller is not compatible with other Pololu serial peripheral products.



Mini SSC II Mode
Baud Rate. The available baud rate range in this mode is approximately 500-10k baud, but the Mini SSC II only works at 2400 or 9600 baud. If you want to put a Mini SSC II servo controller on the same serial line as your Pololu 8-servo controller, you must use one of the two baud rates that the Mini SSC II can support.

Protocol. To set the servo position, send a sequence of three bytes. The first byte is a synchronization value that must always be 255. Byte 2 is the servo number, and it can be 0-254. Byte 3 is the position to which you want the servo to move, also 0-254.

start byte = 0xFF	servo number, 0x00-0xFE	Servo position, 0x00-0xFE

Two motion ranges are available in this mode. Each Pololu servo controller responds to 16 servo numbers. Addressing the lower 8 will move them within an approximately 90 degree range, while addressing the upper 8 servo numbers will give twice the range. You can set the group of servo numbers to which your servo controller responds (see "Setting and Checking the Servo Numbers"). For example, if your servo controller is set to servo numbers 0, it responds to servo numbers 0-15, and sending the command sequence [255, 10, 254] will move servo 2 all the way to one extreme of its range in 180 degree mode. If you send servo numbers that are not recognized, the servo controller will ignore the command. Up to sixteen servo controllers can be connected on one serial line to control up to 128 servos independently.

Pololu Mode

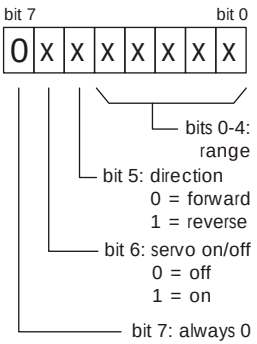
In this mode, there are several options for controlling your servos. As mentioned in the "How Servos and the Servo Controller Work" section, the servo controller holds an internal variable for each servo, the value of which ranges from 500 to 5500, where the number corresponds to the pulse length in increments of half of a microsecond. The various commands deal with setting these internal values. With *absolute* commands, you simply set the value for each servo. In 7- and 8-bit modes, you set *neutral*, *range*, and *direction* parameters for each servo; then, when you send a 7- or 8-bit position command, the servo controller combines all of the parameters to obtain the actual servo position. Whether you use absolute commands or not, you can individually control the speed of each servo and whether the servo is on or off (most servos shut off when they receive no pulses). This section describes the interface details.

Baud Rate. The available baud rate range in this mode is approximately 2,000-40,000 baud. It is possible for the servo controller to operate at rates as high as 57600 baud, but the success of the attempt will depend on the exact speed of your serial control source. The servo controller is not committed to standard baud rates, so if you have a device that can transmit at a maximum rate of 45,000 baud, try it, and it might work.

Protocol. To communicate with the servo controller, send sequences of five or six bytes. The first byte is a synchronization value that must *always* be 0x80 (128). Byte 2 is the Pololu device type number, which is 0x01 for the 8-servo controller. Byte 3 is one of six values for different commands to the controller; the commands are discussed below. Byte 4 is the servo to which the command should apply. Bytes 5 and possibly 6 are the data values for the given command. **In every byte except the start byte, bit seven must be clear. Thus, the range of values for bytes 2-6 is 0-0x7F (0-127).**

start byte = 0x80	device ID = 0x01	command	servo num	data 1	data 2

Command 0: Set Parameters (1 data byte)



- Bit 6 specifies whether a servo is on or not; a 1 turns the servo on, and a 0 (default) turns it off.
- Bit 5 sets the direction the servo moves, which only applies to 7- and 8-bit position commands. If the bit is 0 (default), a larger position number causes the output pulse to get bigger; if the bit is 1, a larger position number will make the output pulse shorter.
- Bits 0-4 set the range through which the servo moves in 7- and 8-bit commands. A larger value will give a larger range, and setting the range to 0 will make the servo always stay at neutral. Given the same range setting, an 8-bit position command will move the servo through twice the range of a 7-bit position command. The default range setting is 15, which will give approximately 180 degrees in 8-bit commands and 90 degrees in 7-bit commands.

Command 1: Set Speed (1 data byte)
This command allows you to set the speed at which the servo moves. If the speed is set to 0 (default), the output pulse will instantly change to the set position. If the speed value is nonzero, the pulse changes gradually from the old position to the new position. With a speed of 1, the pulse width changes at 50 microseconds per second; the maximum speed of 6.35 ms per second is achieved with a speed setting of 127.

Command 2: Set Position, 7-bit (1 data byte)
When this command is sent, the data value is multiplied by the range setting for the corresponding servo and adjusted for the neutral setting. This command can be useful in speeding up communications since only 5 total bytes are sent to set a position. Setting a servo position will automatically turn it on.

Command 3: Set Position, 8-bit (2 data bytes)
This command is just like the 7-bit version, except that two data bytes must be sent to transfer 8 bits. Bit 0 of data 1 contains the most significant bit (bit 7), and the lower bits of data 2 contain the lower seven bits. (Bit 7 in data bytes must always be 0.) Setting a servo position will automatically turn it on.

Command 4: Set Position, Absolute (2 data bytes)
This command allows direct control of the internal servo position variable. Neutral, range, and direction settings do not affect this command. Data 2 contains the lower 7 bits, and Data 1 contains the upper bits. The range of valid values is 500 through 5500. Setting a servo position will automatically turn it on.

Command 5: Set Neutral (2 data bytes)
Setting neutral only applies to 7- and 8-bit commands. The neutral value sets the middle of a range, and corresponds to a 7-bit position value of 63.5 or an 8-bit position value of 127.5. The neutral position is an absolute position just like in command 4, and setting the neutral position will move the servo to that position. The default value is 3000. It may be useful to change neutral if you change servos and need to calibrate your system, or if you cannot get your mechanical linkages to just the right lengths.

Tip: Setting neutral and servo direction can be useful if you have a device, such as a walking robot, that has multiple symmetrical structures on two sides of a chassis. Instead of determining a sequence of positions for each leg individually, you can design a single leg, and then use the same position values for other legs, changing only the neutral position and direction as necessary.

Setting and Checking the Servo Numbers

The Pololu 8-servo controller has the unique feature of allowing the user to set the servo numbers to which the controller responds. By default, the servo controller responds to servo numbers 0-7 (in Pololu mode), but you can set it to respond to numbers 8-15, 16-23, all the way to 120-127. (In Mini SSC II mode, the servo controller would respond to numbers 0-15, 16-31, all the way through 240-254.) This feature is useful if you want to use more than one servo controller at a time to control up to 128 independent servos.

To set the servo numbers, put the servo controller in Pololu mode (shorting block removed from J1) and send the serial sequence [128, 2, <*servonums*>], where <*servonums*> is a number from 0 through 15. A setting of 0 will make the servo controller respond to servo numbers 0-7, a setting of 1 will make it respond to servo numbers 8-15, and so on.

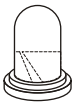
start byte = 0x80	change servo numbers = 0x02	new setting, 0x00-0x10

Upon receiving the command, the servo controller will turn on the red and yellow LEDs and quickly flash the green LED <*servonums*> + 1 times. The green LED will thus quickly flash 1-16 times. The green LED will then pause for approximately 1 second before flashing again. The 8-servo controller must be reset (power turned off and back on) before it can be used.

If you want to just see the servo numbers setting without changing it, use the above command, but use the value 16 for <*servonums*>. The servo number settings will remain unchanged, but the green LED will flash to indicate the servo numbers, as described above.

Troubleshooting Tips

None of the LEDs ever turn on



There is apparently a significant problem with your servo controller, and there could be several reasons for none of the LEDs turning on. First, make sure you are applying a correct voltage and polarity to the power pins. Next, verify that all of your LEDs are soldered in correctly. When you look at the LEDs from the side, the metal inside the plastic dome of the LED appears as indicated in the drawing to the right. The smaller side should be on the side of the serial connector (CON3). Also, check that the notch on U1 matches the notch on the PCB silkscreen. Next, make sure that the voltage regulator, U2, and the capacitors are installed correctly. A last thing to verify visually is the solder connections: make sure there are no shorts between adjacent pins and all solder joints are shiny.

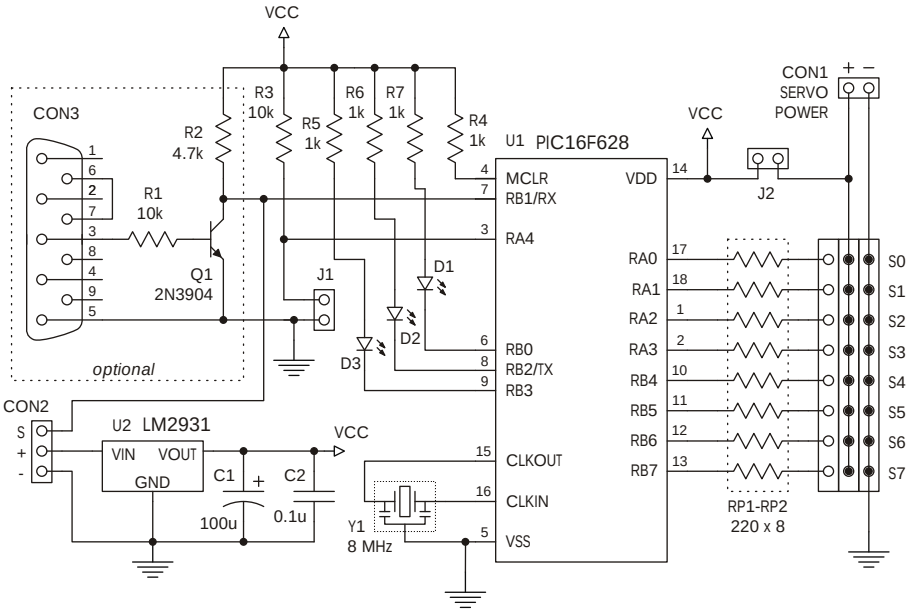
Troubleshooting Tips (continued)

If everything looks correct and the LEDs still don't come on, you will need a multimeter to do more troubleshooting. Begin by disconnecting power and checking continuity between the power inputs to check for a short circuit on the board. Also, check for a short between regulated power and ground (this can easily be done across C1 or C2). If there is a short, use your meter and the schematic diagram on the next page to find it and fix it. Next, using the schematic diagram, verify that all of the proper connections are made. If everything looks correct but your servo controller still doesn't work, connect power to the board and make sure that you have 5 volts across C2. If not, the voltage regulator may have to be replaced. Otherwise, remove U1 (disconnect power first) and connect a wire between the ground and any of the U1 pins that connect to the LEDs. The corresponding LED should turn on; if it doesn't, the LED may need to be replaced; if the LEDs turn on, then U1or Y1 may need to be replaced.

Some of the LEDs never turn on

Remove U1 from its socket, apply power, and connect a wire between ground and the U1 output pin corresponding to the LED that does not turn on (use the schematic on page 15). If the LED does not turn on, it might be installed incorrectly or damaged. If the LED does light up, U1 may be damaged and need replacement. Contact Pololu for more assistance.

Servo Controller Schematic Diagram



The Pololu Serial 8-Servo Controller

The Pololu servo controller allows you to connect up to eight RC servos to almost any robot controller. The interface to the servo controller is a TTL-level serial line or a standard RS-232 serial port, using any baud rate between 1200 and 38400 baud. The servo controller has advanced features that allow you to individually control each servo's speed and motion range. Multiple servo controllers can be connected to a single serial line, and they are compatible with the Pololu Motor Controller, so that an almost unlimited number of servos and motors can be controlled.

Specifications

PCB size	1.45" x 1.70"
Number of servo ports	8
Pulse width range	0.25-2.75 ms
Resolution	1 microsecond (about 0.1 degree)
Supply voltage	5.6-25 V
I/O voltage	0 and 5 V
Baud rate	1200-38400 (auto detect)
Components in kit	24 (including PCB)
Current consumption	15 mA (average)