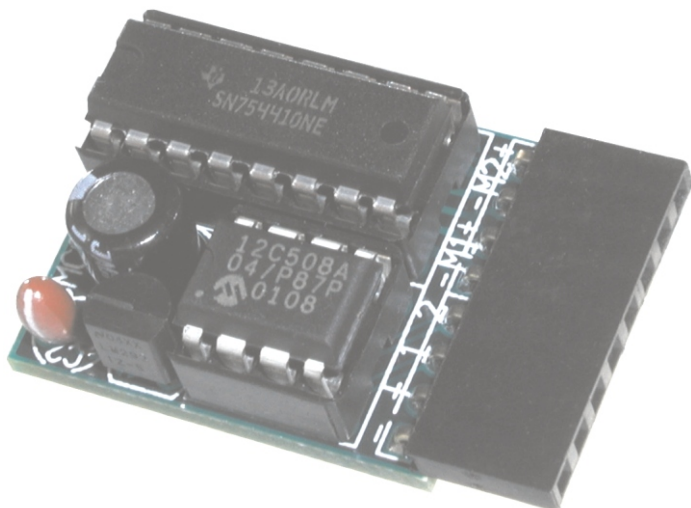


# Dual Serial Motor Controller

*User's Guide*



## Contents:

- Safety Warning
- Parts List
- Contacting Pololu
- How to Solder
- Assembly Instructions
- Connecting the Motor Controller
- Basics of the Serial Interface
- Configuring the Motor Controller
- Using the Motor Controller
- Troubleshooting Tips
- Example BASIC Stamp II Program
- How the Motor Controller Works
- Description and Specifications



© 2004

<http://www.pololu.com/>

SMC01B

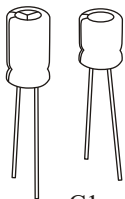


## Important Safety Warning

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.**

## Parts List

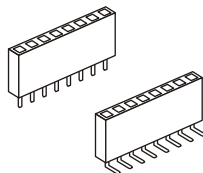
The following components are the motor controller parts. Make sure to verify that all components are included, and that you know which component is which. Each component is labeled with its reference number and description. There are 11 parts in the kit, including the PCB.



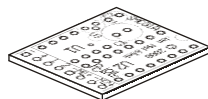
C1  
Electrolytic  
Capacitor  
(2 options)



C2  
Tantalum  
Capacitor



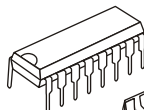
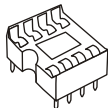
CONNECTOR  
8-pin female  
header  
(2 options)



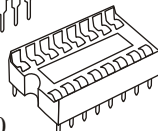
PCB  
Printed Circuit  
Board



U1  
PIC12F629  
Microcontroller  
and Socket



U2  
SN754410  
Dual H-Bridge  
and Socket



U3  
LM2931  
5-volt Regulator



## Contacting Pololu

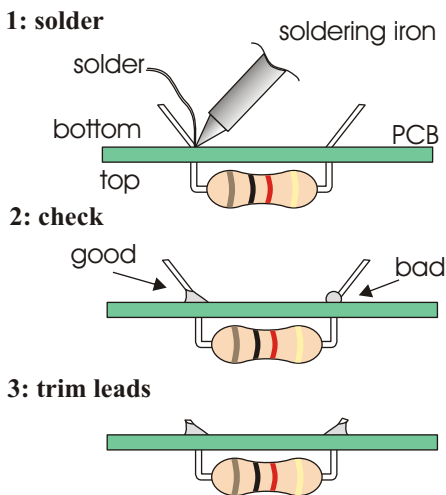
You can check the Pololu web site at <http://www.pololu.com/> for the latest information about the motor controller, including color pictures, application examples, and troubleshooting tips.

We would be delighted to hear from you about your project and about your experience with our motor controller. You can contact us through our online feedback form or by email at [support@pololu.com](mailto: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!

## How to Solder

You need a soldering iron and diagonal cutters to assemble the motor 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. All components in this kit, except for voltage regulator U3, should be flush with 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.



## Assembly Options

You can assemble the motor controller in several ways, so before you begin, there are three choices you must make. The options concern the size of the assembled motor controller. You can insert the various components into the PCB without soldering to help determine which options are best for you.

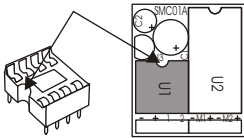
**Should I use IC sockets for U1 and U2?** The sockets are not necessary, and not using them will make your motor controller slightly smaller. Not using them, however, will make it much more difficult to replace the ICs if they are damaged, and also make it possible for you to damage the devices while soldering. **We strongly recommend using the sockets.**

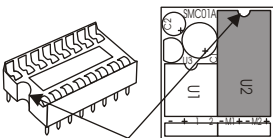
**Which C1 option should I use?** Two electrolytic capacitors are provided in the kit. The larger capacitor is rated for 25 volts, whereas the smaller one is rated for 16 volts. These ratings limit the voltage you can apply to the motor controller and thus limit the maximum voltage with which you can drive the motors. **If you want to drive the motor with voltages higher than 16 volts, or if you don't care about size, use the larger capacitor.**

**Which connector should I use?** A straight connector and a right-angle connector are provided in the kit. It is up to you which you choose to use; you can also solder motor leads directly into the PCB holes and avoid using a connector altogether.

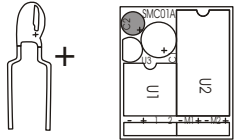
## Assembly Instructions

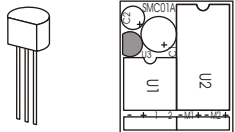
**Caution:** The components **U1-U3 can be damaged by static electricity.** Ground yourself (touch a water pipe or the metal frame of a piece of electrical equipment) before handling these components, and avoid touching their leads. Once assembled, the motor controller can be stored safely in the conductive bag in which the kit is packaged.

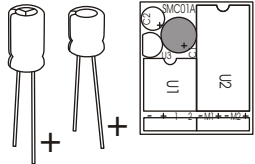
- 

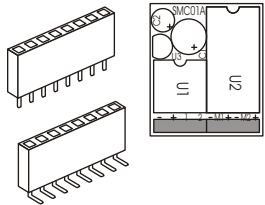
1 Insert the 8-pin socket from the top side of the PCB in the area indicated U1, and solder. On one side of the socket is a notch that should be aligned with the notch on the PCB drawing. The socket protects the PIC microcontroller from being damaged during soldering.
- 

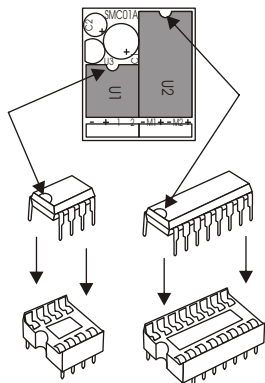
2 Solder the 16-pin socket in the area indicated U2. Make sure to align the notch on the socket with the notch on the PCB drawing. The socket allows you to replace the motor driver chip if it ever breaks.

3  Next, add the tantalum capacitor C2. You may need to bend the leads to make them straight so that they will fit. The capacitor is *polarized*, which means it **must only go in one way**. Make sure the side labeled with a “+” and a stripe goes into the hole that is also marked with a “+”. If the PCB is oriented as shown in the diagram, the lower hole is for the positive lead.

4  Now, add the voltage regulator, U3. Make sure the device is oriented as shown on the silkscreen drawing, with the flat side facing the outside of the PCB. **Caution! The voltage regulator can be damaged by static electricity.**

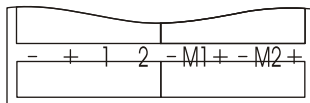
5  In this step, add your choice of electrolytic capacitor for C1. The larger capacitor is necessary if you want to run your motors off of a 24-volt power source, and you should use it if you don't care about the size of the completed motor controller. **The electrolytic capacitors are polarized, but this time the stripe identifies the negative terminal.** Also, the positive lead is longer. Make sure to match up the positive lead with the appropriate hole in the PCB.

6  Next, solder in your choice of connector. You can use either the straight or the right-angle female header provided in your kit, or use your own connector. If you prefer having leads soldered directly to the PCB, you can do so now or wait until you have the leads ready and available.

7  Finally, insert the two integrated circuits (ICs), U1 and U2, into their sockets. **Make sure that you plug them in so that the notches on the ICs match the notches on the PCB outline.** Do not solder the ICs to the sockets. If you soldered the sockets in backwards, it doesn't matter as long as the actual component is oriented correctly. **Be careful: the ICs are static-sensitive**, so take appropriate precautions and avoid touching their leads. You may need to bend some of the leads to make them fit in the sockets; if so, hold the plastic body of the IC and push the pins (gently!) against a flat surface, one row at a time.

## Connecting the Motor Controller

There are eight pins on the bottom of the motor controller for connecting it to the rest of your system. A closeup of the bottom of the PCB is shown to the right, in case you have a hard time reading the silkscreen on your board. The eight pin labels and the corresponding functions are shown in the table.



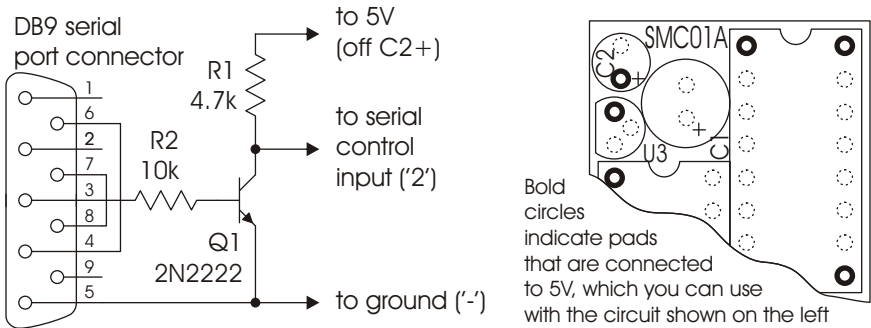
LABEL	FUNCTION
-	ground (0V)
+	positive supply (5.6-25V)
1	reset
2	serial control input
M1-	motor 1, negative output
M1+	motor 1, positive output
M2-	motor 2, negative output
M2+	motor 2, positive output

**Connecting Power. Warning: connecting power incorrectly can cause some components to explode.** Connect the ground pin to a ground terminal on your robot controller. If you have a separate power supply for just the motors, make sure that you connect the negative terminal of that supply to the same ground.

(This situation may arise if, for example, you want to run your robot controller off of a 9-volt battery and you want to run your motors off of a 12-volt battery. You will also need an independent power supply for the motors if you want to use a personal computer as the robot controller. In that case, you might use a battery for the motor supply and use a wall outlet for the PC supply.) Connect the '+' pin to the positive terminal of the motor supply; this terminal may connect only to the motor controller, or it may connect to any other device powered by that supply. **Warning: the supply voltage may not exceed 16 volts or 25 volts, depending on which capacitor you chose for C1 in step 5 of assembly.**

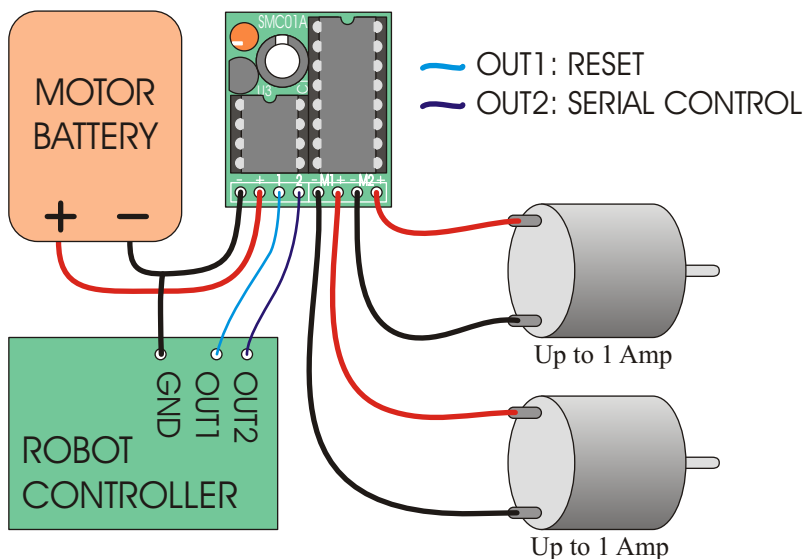
**Reset Input.** Connect this pin to a digital output on your robot controller. The reset line must be kept high (+5 V) for the motor controller to operate; bringing it low (to 0 V) for at least 2 microseconds resets the motor controller to its initial state (all motors off, waiting for its first serial command). You might also use a pull-down resistor on the reset line so that if your main controller gets reset, the motors do not keep running. We strongly recommend using the reset line, but if you do not want to use it, connect it through a 4.7 kOhm resistor to your logic supply. **After turning on the motor controller or resetting it, allow 100 ms for it to start up before sending any serial data.**

**Serial Input.** Use a pin on your main controller that can be used as a logic-level, asynchronous serial output. Serial data can be sent down this line 8 bits at a time, with no parity bit, at any rate between 1200 and 19200 baud. *Once you choose a baud rate, you cannot change it until the motor controller is reset.* **Important note:** unlike RS-232 serial lines (the standard for serial ports used to connect devices to personal computers), this line uses logic voltages between 0 and +5 V. The higher voltages used on RS-232 lines will damage the motor controller. If you need to convert RS-232 levels to TTL levels, you will need to use a level converter such as the MAX220 (made by Maxim). You could also use the simple circuit shown below. **When building circuits that connect to a PC, be especially careful because you could potentially destroy the PC's serial port. Before attempting to connect your own electronics to a computer, make sure you know what you are doing!**



The above diagram shows a simple circuit for connecting the motor controller to a PC serial port. You will need to connect one side of resistor R1 to a 5V supply, which is available on the PCB at the points indicated on the figure to the right. You can solder a wire onto one of the pads, but make sure that the wire touches only the intended pad.

**Connecting the Motors in Dual Motor Mode.** If you are using your motor controller to control two independent motors, connect one or two motors to the pins labeled M1 and M2. You probably don't need to worry too much about the polarity, but the '+' pins go positive when the controller receives "forward" commands. If you find out that your motors turn in different directions than you expect, you can flip the wiring or just switch the forward and reverse commands on your robot controller program.

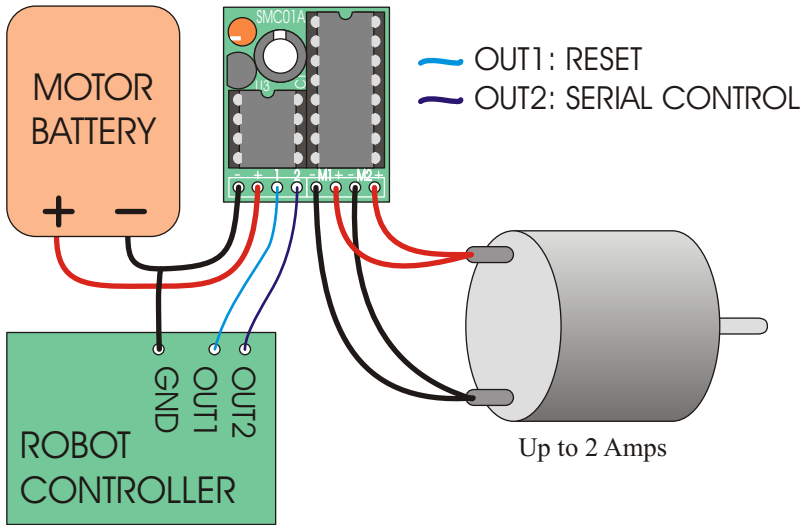


### A typical two-motor setup.

The green box labeled "robot controller" represents a main control unit that includes a battery that is not shown. This robot controller could be a microcontroller or a device such as the BASIC Stamp from Parallax. Keep in mind that the wiring you use for the motor outputs and power connections should be capable of conducting several amps. We recommend using at least 26 gauge wire (remember, smaller numbers mean bigger wires!).



**Connecting One Motor in Single Motor Mode.** If you are using your motor controller to control a single motor, you must use all four pins labeled M1 and M2. Before connecting the single motor, make sure that you have configured the motor controller for single motor mode. Connect pins M1+ and M2+ to one motor lead, and connect pins M1- and M2- to the other motor lead. **If you make these connections in dual-motor mode, you could destroy your motor controller!**



### A typical single-motor setup.

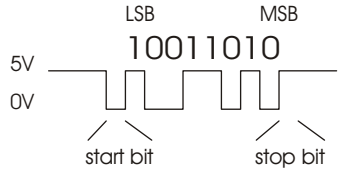
In this configuration, the two H-bridges of the motor controller are wired in parallel. Single-motor mode must be used to ensure that the two H-bridges are also controlled in parallel. This setup allows up to 2 A to be delivered to the motor.

The green box labeled “robot controller” represents a main control unit that includes a battery that is not shown. This robot controller could be a microcontroller or a device such as the BASIC Stamp from Parallax. Keep in mind that the wiring you use for the motor outputs and power connections should be capable of conducting several amps. We recommend using at least 26 gauge wire (remember, smaller numbers mean bigger wires!).

# Basics of the Serial Interface

The motor controller uses a serial interface to communicate with a main controller, which could be a small microprocessor or a desktop computer. To use the motor controller, you must program your main controller to send data with the correct format to the motor controller's asynchronous serial input, which is the pin labeled '2'.

The motor controller expects eight bits at a time (with no parity bit) at a constant baud rate ranging from 1200 to 19200 baud (the motor controller will automatically detect the baud rate). The serial bits must be at logic levels and *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 PC-connection circuit on page 7 corrects the inverted signal coming out of PC serial ports.) *Commands sent to the serial input **must** conform to the above format or else the motor controller and other devices connected to the serial line may behave unexpectedly.*



Once you can send individual bytes correctly, you must send the correct sequence of bytes to get the motor controller to run your motors. This motor controller *interface protocol* is compatible with other Pololu serial devices such as our servo controller, so you can control multiple Pololu serial devices on a single line. The protocol requires one start byte, a one-byte device identifier, and then any number of bytes, as required by the device specified in the second byte:

start byte = 0x80	device type	data byte 1	data byte 2
-------------------	-------------	-------------	-------------

The start byte is identified by its most significant bit being set; all subsequent bytes must have bit 7 clear, giving them possible values of 0 to 0x7F (0 to 127 decimal). Whenever a byte is transmitted on the serial line, all devices on that line check to see if the byte is the start byte; if it is, then all devices check the next byte to see if the data is meant for them. All subsequent bytes, the data bytes in the diagram above, are only interpreted by the appropriate devices, while all other devices wait for a new start byte.

If you did not understand all of the details above and you just want to use your motor controller, don't worry. You just need to use the right serial settings and send the correct sequences of bytes, as described on the following pages.

**Summary:** Use non-inverted, logic-level serial transmission at baud rates between 1200 and 19200, 8 bits at a time with no parity and one stop bit.



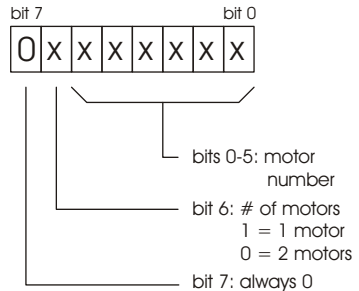
## Configuring the Motor Controller

You can configure your motor controller to control a single motor or to control two motors independently. You can also set which motor number a particular motor controller will control, in case you want to control many motors off of one serial line. During configuration, you should connect just one motor controller at a time to your serial line unless you want to configure each motor controller the exact same way. **The default configuration is for two-motor control with motor numbers 2 and 3.**

Configuration is achieved by sending a three-byte packet consisting of the start byte, a configuration command byte, and the new configuration byte:

start byte = 0x80	change configuration = 0x02	new settings, 0x00-0x7F
-------------------	-----------------------------	-------------------------

The new settings byte contains two parts: a six-bit motor number and a one-bit flag specifying one-motor or two-motor mode.



- Bits 0-5 specify the motor number(s) to which the motor controller will respond. In single-motor mode, the number you choose sets the number to which the motor controller will respond. In two-motor mode, the motor controller will respond to two consecutive numbers. If you set an even motor number, the motor controller will control that motor number and the one above it; if you set an odd motor number, the motor controller will control that motor number and the one below it. **Note that all motor controllers will respond to motor number 0 (and 1, if in two-motor mode).**
- Bit 6 specifies whether the motor controller is in one-motor mode or in two-motor mode. If this bit is clear, the motor controller will be in two-motor mode; if the bit is set, the motor controller will be in 1-motor mode.

After sending the change configuration command, the motor controller will pulse pin M1- for one-motor mode and pin M1+ for two-motor mode  $\langle motornumber \rangle + 1$  times. If you want to verify that you correctly configured your motor controller to control one motor, say motor number 3, then connect a motor or light between pin M1- and ground and check that you see 4 pulses after you send the command. (The reason for the extra pulse is so that you get some response if you set the motor number to zero). **After configuration, the motor controller must be reset (either by turning it off and back on or by using the reset line) before you can continue using it.**

```
Examples: (Using PBASIC "SEROUT" command with serial line on pin 5)
\ "84" parameter sets up 9600 baud serial communication
SEROUT 5, 84, [$80,2,2]      \2-motor mode, controlling motors 2 and 3
SEROUT 5, 84, [128,2,68]    \1-motor mode, controlling motor 4
SEROUT 5, 84, [$80,$2,$44]  \same as above, using hexadecimal
```



## Using the Motor Controller

To set the speed and direction of a motor, send a four-byte command with the following structure to the motor controller:

start byte = 0x80	device type = 0x00	motor # and direction	motor speed
-------------------	--------------------	-----------------------	-------------

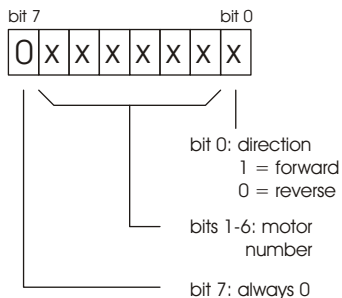
### The Four-Byte Motor Controller Command

**Byte 1: Start Byte.** This byte should *always* be 0x80 (128 in decimal) to signify the beginning of a command. The start byte is the only byte with the highest bit (bit 7) set, and it alerts all devices on the serial line that a new command is being issued. All succeeding bytes sent down the serial line must have their highest bit cleared to zero.

**Byte 2: Device Type.** This byte identifies the device type for which the command is intended, and it should be 0x00 for commands sent to this motor controller. All devices that are not dual motor controllers ignore all subsequent bytes until another start byte is sent.

**Byte 3: Motor Number and Direction.** This byte has three parts, as shown in the diagram to the right:

- Bit 0 specifies the direction of the motor. Set this bit to 1 to make the motor go forward; clear the bit to make it go backward.
- Bits 1-6 specify the motor number. All motor controllers respond to motor number(s) 0 (and 1, in dual-motor mode).
- Bit 7 must be cleared since this is not a start byte.



To obtain the complete byte 3 value from a motor number and a direction, multiply the motor number by 2 and add 1 if the direction is forward. For example, to make motor 5 go forward, byte three should be  $5 \times 2 + 1 = 11$ . To make motor 1 go backward, byte 3 should be  $1 \times 2 = 2$ . (Two efficient ways to multiply by 2 in a microcontroller program are shifting left by one digit or adding the motor number to itself.)

**Byte 4: Motor Speed.** The most significant bit must be zero since this is not a start byte. The remaining seven bits specify the motor speed. The possible range of values for byte 4 is thus 0x00 to 0x7F (0 to 127 decimal). 0x00 turns the motor off, and 0x7F turns the motor fully on; intermediate values correspond to intermediate speeds. Setting a speed of 0 in forward or reverse will cause the motor controller to brake.

```
Examples: (Using PBASIC "SEROUT" command with serial line on pin 5)
\ "84" parameter sets up 9600 baud serial communication
SEROUT 5, 84, [$80,0,5,127] `motor 2 full on, forward
SEROUT 5, 84, [$80,0,5,0]   `motor 2 off, forward (braking)
SEROUT 5, 84, [$80,0,4,0]   `motor 2 off, reverse (braking)
```



## Resetting the Motor Controller

The motor controller's reset line should normally be kept high at +5 V. Pull the reset line low to 0 V for at least 2 microseconds to reset the motor controller to its initial state (all motors off, waiting for the first serial command). You might also use a pull-down resistor on the reset line so that if your main controller gets reset, the motors do not keep running. We strongly recommend using the reset line, but if you do not want to use it, connect it through a 4.7 kOhm resistor to your logic supply. **After turning on the motor controller or resetting it, allow 100 ms for it to start up before sending any serial data.**

## Controlling Multiple Motor Controllers with One Serial Line

To control a particular motor, you must specify its motor number in command byte 3. Regardless of configuration, every motor controller responds to commands for motor number 0, and, in two-motor mode, to motor 1. To control more than two motors with a single serial line, you need to use motor numbers 2 through 63. Configure each motor controller to respond to different motor numbers, then connect them to the same serial line; each motor controller will respond only to the motor number to which it is configured.

For example, to control six motors independently with dual-motor mode, you need three motor controller boards, each with different motor numbers. All three motor controllers respond to commands for motor numbers 0 and 1. For controlling the six motors independently, use motor numbers 2, 3, 4, 5, 6, and 7. (In single-motor mode, you would need six motor controllers configured to numbers 1 through 6 since only motor number 0 is a universal motor number.)

## Troubleshooting Tips

If your motor controller does not work at first, it can be difficult to determine the cause because a broken unit could look just like a working one! Nevertheless, patience and meticulous attention to detail, along with these few tips, should usually help you through:

**Check all of your solder joints.** A good solder joint should look shiny, and you should have no shorts. If you have a multimeter, check that you have all of the connections as shown on the schematic on page 15.

**Double check all of your connections.** Are your logic and motor supply grounds connected? If you are using a breadboard or sockets, are the motor controller pins all making good contact?

**Double check your code.** Are your baud rate settings correct? If you cannot get your design working with the top baud rate of 19200, try lowering it to 9600, where slight timing mismatches are less likely to frustrate your efforts.

**Are you using the correct motor number?** If nothing seems to be working, start by using motor number 0, which should work regardless of the configuration.

**Are you using good power supplies?** Make sure your power supply can supply enough current for your motors. Do not use regulated power for your motor supply.

**If your motors run and then unexpectedly stop,** your motor driver chip might be overheating. You can help the situation by putting a heat sink on the motor driver chip, lowering your motor supply voltage, and putting short stops between changes in motor direction.



## Example BASIC Stamp II Program

This program, which can run on a BASIC Stamp II controller, makes motor 1 gradually speed up, then slow down, then speed up in the other direction, and then slow down again. For the code to work, pin 15 must be connected to the reset input (1), and pin 14 must be connected to the serial input (2). The interface code should look similar in other programming languages; the description below should help you in understanding the code and, if necessary, in translating it to other languages.

On line 1, the 8-bit variable `speed` is declared for later use. The serial line is then taken high, to its idle state, before the motor controller is reset by a low-going pulse on pin 15 (lines 3 and 4). A 100-ms pause on line 5 ensures that the motor controller is up and running before any serial data is sent to it.

The first *for loop* on lines 6-9 causes motor 1 to gradually speed up. The serial output is created by the `serout` statement on line 7. The first parameter, 14, specifies the pin number through which to send the serial signal. The next parameter, 84, sets up the serial characteristics to be 8 bits with no parity, non-inverted, at a baud rate of 9600. The four numbers in square brackets are the data to be sent, and they correspond to the four control bytes for the motor controller. The first two bytes should always be \$80 and 0. The second 0 makes motor 1 go backward. The speed variable, which increases every time through the loop, is the only part of the command that changes, and that is what makes the motor gradually speed up. The `pause` statement on line 8 causes the program to wait for 20 ms (0.02 seconds) before sending the next command.

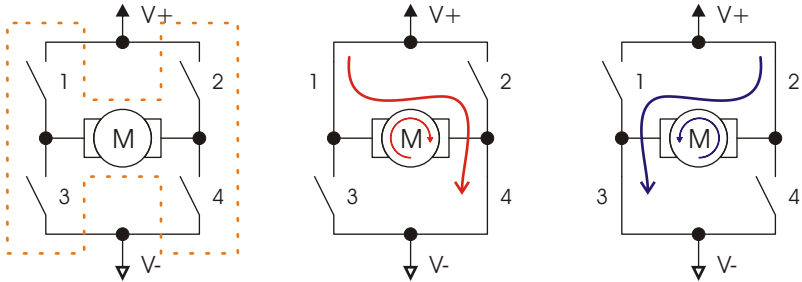
When the first loop ends, the motor is set to its full speed of 127. The second loop on lines 10-13 slows the motor back down by sending speeds from 127 down to 0. The next two loops on lines 14-21 then repeat the process, except for the parameter value of 1 in byte three, which causes motor 1 to spin forward.

```
1      speed    var    byte
2      high 14          `take serial line high
3      low 15           `reset motor controller
4      high 15
5      pause 100        `motor controller startup time
6      for speed = 0 to 127
7          serout 14,84,[$80, 0, 0,speed]
8          pause 20
9      next
10     for speed = 127 to 0
11         serout 14,84,[$80, 0, 0,speed]
12         pause 20
13     next
14     for speed = 0 to 127
15         serout 14,84,[$80, 0, 1,speed]
16         pause 20
17     next
18     for speed = 127 to 0
19         serout 14,84,[$80, 0, 1,speed]
20         pause 20
21     next
```



# How the Motor Controller Works

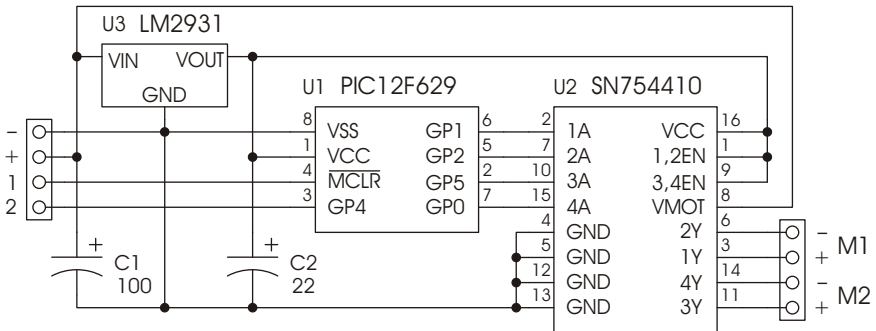
The motor controller uses *H-bridges* to turn motors forward and backward (see the dotted 'H' in the left figure). H-bridges have four switches, which are turned on in pairs to allow current to flow into the motors in both directions, as shown below. In the left figure, all four switches are open, and the motor is turned off. When switches 1 and 4 close, the motor turns in one direction; when switches 2 and 3 close, the motor turns the other way. Integrated circuit U2 contains two H-bridges, allowing bidirectional control of two motors.



A technique called *pulse width modulation* (PWM) is used to control the speed of the motors. The microcontroller (U1) is a little computer that controls the H-bridge switches. It turns the switches on and off very rapidly (600 times per second) and varies the percentage of the time that the switches are on to achieve the speed set by the serial interface. For a higher speed, the switches are on a larger fraction of the time than for a slower speed. At the maximum speed of 127, the switches are left on. The momentum of the motor shaft keeps the shaft spinning at a constant speed that can be varied smoothly over all 127 different speeds.

U3 and the capacitors provide a steady 5V power supply for the microcontroller, which cannot run at the full motor voltage provided to the board at the '+' and '-' pins.

The complete schematic diagram of the motor controller is shown below:



## The Pololu Dual Serial Motor Controller

For a robot to interact with its environment, it must be able to convert electrical signals into motion. However, the power requirements of *actuators*, electrical devices capable of producing motion, are typically so high that normal digital circuitry cannot drive them. In addition, precise motion control requires constantly changing the signals sent to the actuators, leaving the control circuitry with little time to attend to other tasks.

The Pololu motor controller bridges the gap between robot controllers and power-hungry actuators. Using one serial output from your robot controller, you can independently set each of two small DC motors (the kind typically found in remote-control cars and motorized toys) to go forward or backward at any of 127 different speeds. To control additional motors, you can connect multiple motor controllers to the same serial line. The motor controller is compatible with the Pololu Servo Controller, so you can control an almost unlimited number of motors and servos with one serial line. Because of its small size, the motor controller can fit almost any robot design

### Specifications

PCB size.....	1.00" x 0.85"
Motor ports.....	2
Speeds.....	127 forward, 127 backward, and brake
Maximum current.....	two motors 1 A each, one motor 2 A
Supply voltage.....	5.6-25 V
Data voltage.....	0 and 5 V
PWM frequency.....	two motors 600 Hz, one motor 750 Hz
Serial baud rate.....	1200-19200 (automatically detected)

