

code, circuits, & construction

code and fabrication resources for physical computing and networking

tigoe.net | [pcomp home](#) | [blog](#) | [code, circuits, & construction](#) | [my del.icio.us links](#)

HOME

{ 2004 05 24 }

PAGES

- About
- Understanding
- Electricity
 - Breadboards
- Input & Output
 - Digital Input & Output
 - Analog Input
 - Analog Output
- Serial Communication
 - Interpreting Serial
 - Data
 - MIDI

CATEGORIES

- Arduino/Wiring
- AVR-C
- Basic stamp
- BasicX
- circuits
- code
- construction
- Java
- Lantronix
- Lingo
- Max/MSP

Stepper Motor Control

A stepper motor is a motor controlled by a series of electromagnetic coils. The center shaft has a series of magnets mounted on it, and the coils surrounding the shaft are alternately given current or not, creating magnetic fields which repulse or attract the magnets on the shaft, causing the motor to rotate.

This design allows for very precise control of the motor: by proper pulsing, it can be turned in very accurate steps of set degree increments (for example, two-degree increments, half-degree increments, etc.). They are used in printers, disk drives, and other devices where precise positioning of the motor is necessary.

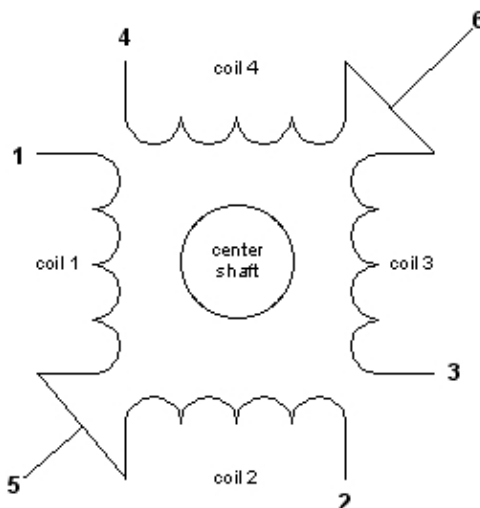
There are two basic types of stepper motors, unipolar steppers and bipolar steppers.

Unipolar Stepper Motors

The unipolar stepper motor has five or six wires and four coils (actually two coils divided by center connections on each coil). The center connections of the coils are tied together and used as the power connection. They are called unipolar steppers because power always comes in on this one pole.

Perl
 PHP
 PIC C
 PicBasic Pro
 Processing
 Uncategorized
 Unix/Linux
 XBee

SEARCH



Bipolar stepper motors

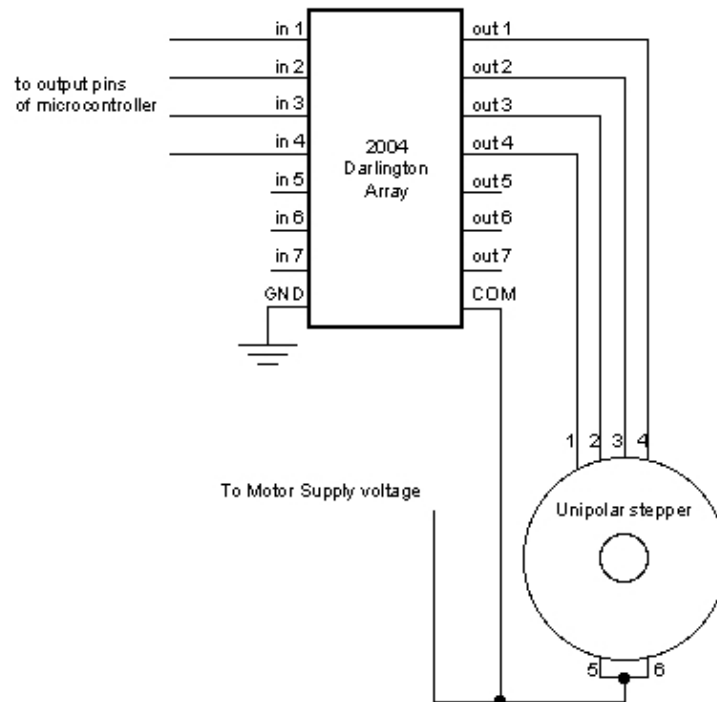
The bipolar stepper motor usually has four wires coming out of it. Unlike unipolar steppers, bipolar steppers have no common center connection. They have two independent sets of coils instead. You can distinguish them from unipolar steppers by measuring the resistance between the wires. You should find two pairs of wires with equal resistance. If you've got the leads of your meter connected to two wires that are not connected (i.e. not attached to the same coil), you should see infinite resistance (or no continuity).

Like other motors, stepper motors require more power than a microcontroller can give them, so you'll need a separate power supply for it. Ideally you'll know the voltage from the manufacturer, but if not, get a variable DC power supply, apply the minimum voltage (hopefully 3V or so), apply voltage across two wires of a coil (e.g. 1 to 2 or 3 to 4) and slowly raise the voltage until the motor is difficult to turn. It is possible to damage a motor this way, so don't go too far. Typical voltages for a stepper might be 5V, 9V, 12V, 24V. Higher than 24V is less common for small steppers, and frankly, above that level it's best not to guess.

To control the stepper, apply voltage to each of the coils in a specific sequence. The sequence would go like this:

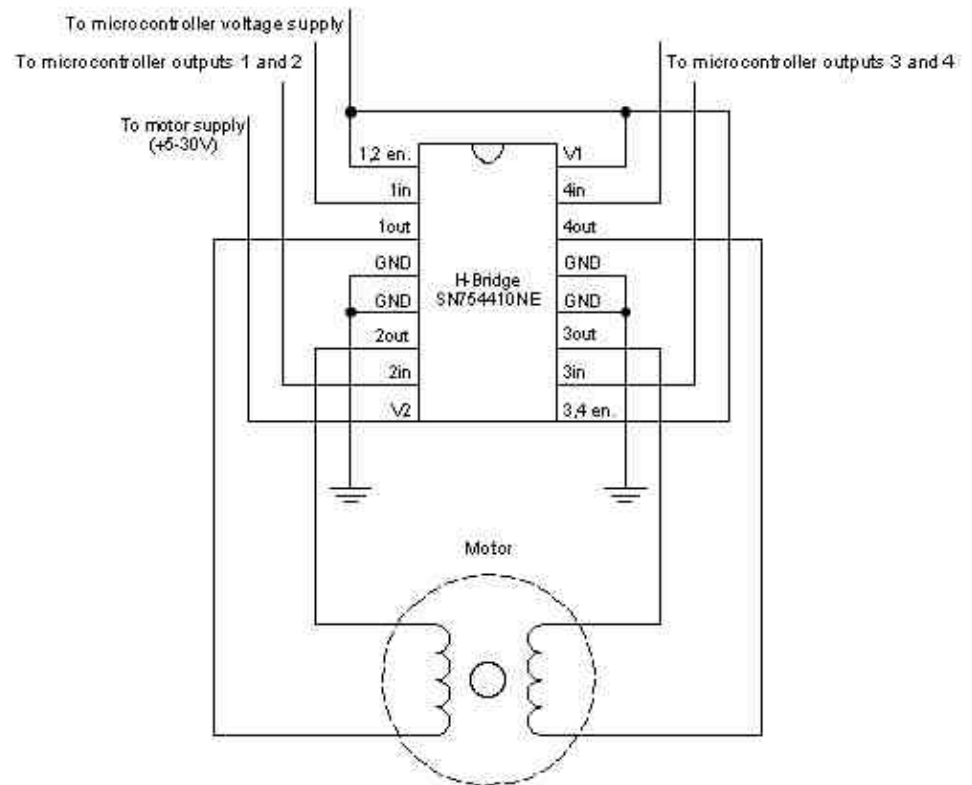
Step	wire 1	wire 2	wire 3	wire 4
1	High	low	high	low
2	low	high	high	low
3	low	high	low	high
4	high	low	low	high

To control a unipolar stepper, you use a Darlington Transistor Array. The stepping sequence is as shown above. Wires 5 and 6 are wired to the supply voltage.



To control a bipolar stepper motor, you give the coils current using to the same steps as for a unipolar stepper motor. However, instead of using four coils, you use the both poles of the two coils, and reverse the polarity of the current.

The easiest way to reverse the polarity in the coils is to use a pair of H-bridges. The L293D dual H-bridge has two H-bridges in the chip, so it will work nicely for this purpose.



Once you have the motor stepping in one direction, stepping in the other direction is simply a matter of doing the steps in reverse order.

Knowing the position is a matter of knowing how many degrees per step, and counting the steps and multiplying by that many degrees. So for examples, if you have a 1.8-degree stepper, and it's turned 200 steps, then it's turned 1.8×200 degrees, or 360 degrees, or one full revolution.

Two-Wire Control

Thanks to Sebastian Gassner for ideas on how to do this.

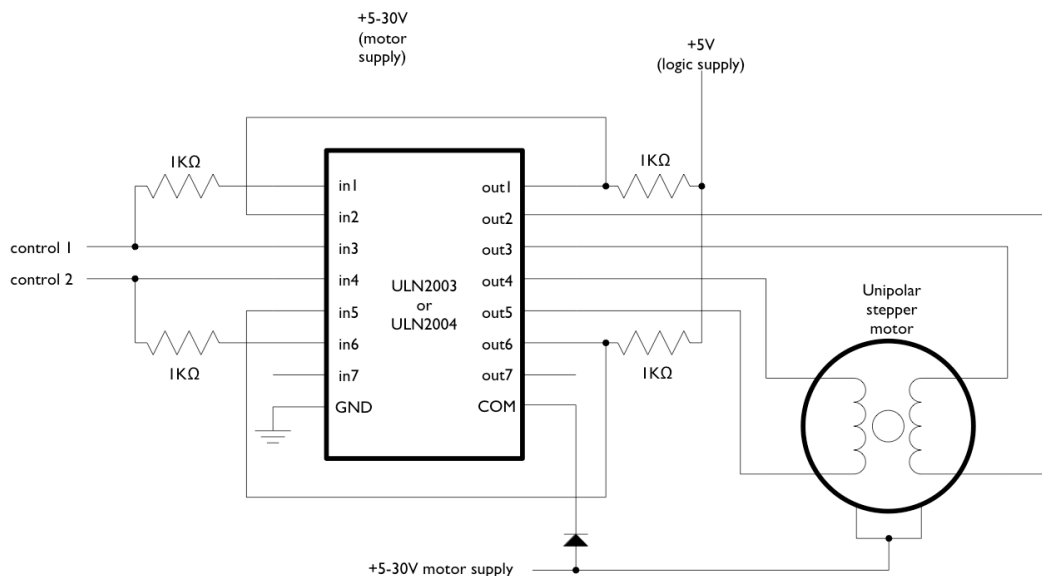
In every step of the sequence, two wires are always set to opposite polarities. Because of this, it's possible to control steppers with only two wires instead of four, with a slightly more complex circuit. The stepping sequence is the same as it is for the two middle wires of the sequence above:

Step	wire 1	wire 2
1	low	high
2	high	high
3	high	low

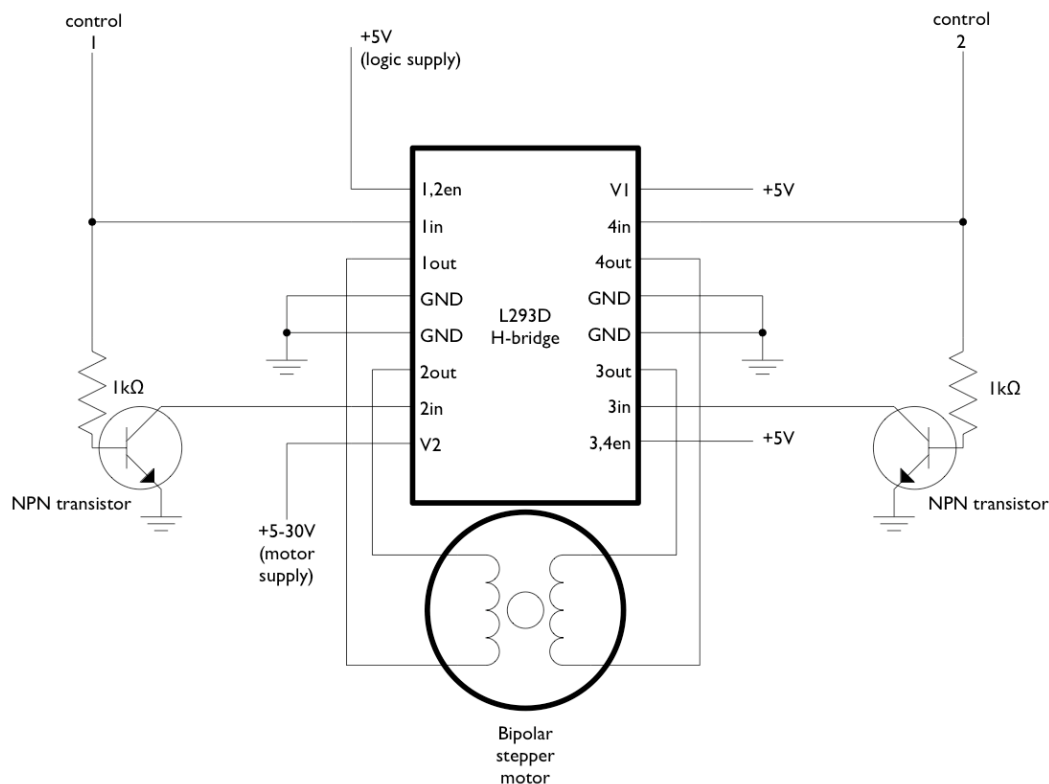
4 low low

The circuits for two-wire stepping are as follows:

Unipolar stepper two-wire circuit:



Bipolar stepper two-wire circuit:



Programming the Microcontroller to Control a Stepper

Because both unipolar and bipolar stepper motors are controlled by the same stepping sequence, we can use

the same microcontroller code to control either one. In the code examples below, connect either the Darlington transistor array (for unipolar steppers) or the dual H-bridge (for bipolar steppers) to the pins of your microcontroller as described in each example. There is a switch attached to the microcontroller as well. When the switch is high, the motor turns one direction. When it's low, it turns the other direction.

The examples below use the 4-wire stepping sequence. A two-wire control program is shown for the Wiring/Arduino Stepper library only.

Wire pins 9-12 of the BX-24 to inputs 1-4 of the Darlington transistor array, respectively. If you're using the PicBasic Pro code, it's designed for a PIC 40-pin PIC such as the 16F877 or 18F452. Use pins PORTD.0 through PORTD.3, respectively. If you're using a smaller PIC, you can swap ports, as long as you use the first four pins of the port.

Note that the wires read from left to right. Their numbers don't correspond with the bit positions. For example, PORTD.3 would be wire 1, PORTD.2 would be wire 2, PORTD.1 would be wire 3, and PORTD.0 would be wire 4. On the BX-24, pin 9 is wire 1, pin 10 is wire 2, and so forth.

BX-24 code:

```
dim motorStep(1 to 4) as byte
dim thisStep as integer

Sub main()
  call delay(0.5) ' start program with a half-second delay

  dim i as integer

  ' save values for the 4 possible states of the stepper motor leads
  ' in a 4-byte array. the stepMotor routine will step through
  ' these four states to move the motor. This is a way to set the
  ' value on four pins at once. The eight pins 5 through 12 are
  ' represented in memory as a byte called register.portc. We will set
  ' register.portc to each of the values of the array in order to set
  ' pins 9,10,11, and 12 at once with each step.

  motorStep(0) = bx0000_1010
  motorStep(1) = bx0000_0110
  motorStep(2) = bx0000_0101
  motorStep(3) = bx0000_1001

  ' set the last 4 pins of port C to output:
  register.ddrc = bx0000_1111

  ' set all the pins of port C low:
  register.portc = bx0000_0000

  do
    ' move motor forward 100 steps.
    ' note: by doing a modulo operation on i (i mod 4),
    ' we can let i go as high as we want, and thisStep
    ' will equal 0,1,2,3,0,1,2,3, etc. until the end
    ' of the for-next loop.

    for i = 1 to 100
```

```

        thisStep = i mod 4
        call stepMotor(thisStep)
    next

    ' move motor backward
    for i = 100 to 1 step -1
        thisStep = i mod 4
        call stepMotor(thisStep)
    next
loop

End Sub

sub stepMotor(byref whatStep as integer)
    ' sets the value of the eight pins of port c to whatStep
    register.portc = motorStep(whatStep)

    call delay (0.1) ' vary this delay as needed to make your stepper step.
end sub

```

PicBasic Pro code:

```

start:
    High PORTB.0

    ' set variables:
    x VAR BYTE
    steps VAR WORD
    stepArray VAR BYTE(4)
    clear

    TRISD = %11110000
    PORTD = 255
    input portb.4
    Pause 1000

    stepArray[0] = %00001010
    stepArray[1] = %00000110
    stepArray[2] = %00000101
    stepArray[3] = %00001001

main:
    if portb.4 = 1 then
        steps = steps + 1
    else
        steps = steps - 1
    endif

    portD = stepArray[steps //4]
    pause 2

GoTo main

```

pBasic (Basic Stamp 2) code:

```

' set variables:
x          var    byte
stepper    var    nib
steps      var    word

' set pins 8 - 10 as outputs, using DIRS to do so:
dirs.highbyte = %00001111

main:
    steps = 200
    gosub clockStep
    pause 1000
    gosub counterClockStep
    pause 1000
goto main

clockStep:
    debug "counter" , cr
    for x = 0 to steps
        lookup x//4, [%1010,%1001,%0101,%0110], stepper
        outs.highbyte.lownib = stepper
        pause 2
    next

```

```

return
counterclockStep:
  debug "clockwise", cr
  for x = 0 to steps
    lookup x//4, [%0110,%0101,%1001,%1010], stepper
    outs.highbyte.lownib = stepper
    pause 2
  next
return

```

Wiring Code (for Arduino board):

This example uses the [Stepper library](#) for Wiring/Arduino. It was tested using the 2-wire circuit. To change to the 4-wire circuit, just add two more motor pins, and change the line that initializes the Stepper library like so:

```

Stepper myStepper(motorSteps,
motorPin1,motorPin2,motorPin3,motorPin4);

/*
Stepper Motor Controller
language: Wiring/Arduino

This program drives a unipolar or bipolar stepper motor.
The motor is attached to digital pins 8 and 9 of the Arduino.

The motor moves 100 steps in one direction, then 100 in the other.

Created 11 Mar. 2007
Modified 7 Apr. 2007
by Tom Igoe

*/

// define the pins that the motor is attached to. You can use
// any digital I/O pins.

#include <Stepper.h>

#define motorSteps 200 // change this depending on the number of steps
// per revolution of your motor
#define motorPin1 8
#define motorPin2 9
#define ledPin 13

// initialize of the Stepper library:
Stepper myStepper(motorSteps, motorPin1,motorPin2);

void setup() {
  // set the motor speed at 60 RPMS:
  myStepper.setSpeed(60);

  // Initialize the Serial port:
  Serial.begin(9600);

  // set up the LED pin:
  pinMode(ledPin, OUTPUT);
  // blink the LED:
  blink(3);
}

void loop() {
  // Step forward 100 steps:
  Serial.println("Forward");
  myStepper.step(100);
  delay(500);

  // Step backward 100 steps:
  Serial.println("Backward");
  myStepper.step(-100);
  delay(500);
}

```



```
}  
  
// Blink the reset LED:  
void blink(int howManyTimes) {  
  int i;  
  for (i=0; i< howManyTimes; i++) {  
    digitalWrite(ledPin, HIGH);  
    delay(200);  
    digitalWrite(ledPin, LOW);  
    delay(200);  
  }  
}
```

For more on steppers, see the [DC motor notes](#) on this site.

Posted by tigoe on Monday, May 24th, 2004, at 8:15 am, and filed under [BasicX](#), [PicBasic Pro](#), [Arduino/Wiring](#).

Follow any responses to this entry with the [RSS 2.0](#) feed.

« [PICBASIC PRO DEBUG STATEMENT](#) [ANALOG IN USING RCTIME](#) »



(cc) tigoe | Thanks, [WordPress](#) | [Barthelme](#) theme by [Scott](#) | Valid [XHTML](#) & [CSS](#) | [RSS](#): [Posts & Comments](#)