

Introduction. This application note demonstrates simple hardware and software techniques for driving and controlling common four-coil stepper motors.

Background. Stepper motors translate digital switching sequences into motion. They are used in printers, automated machine tools, disk drives, and a variety of other applications requiring precise motions under computer control.

Unlike ordinary dc motors, which spin freely when power is applied, steppers require that their power source be continuously pulsed in specific patterns. These patterns, or step sequences, determine the speed and direction of a stepper's motion. For each pulse or step input, the stepper motor rotates a fixed angular increment; typically 1.8 or 7.5 degrees.

The fixed stepping angle gives steppers their precision. As long as the motor's maximum limits of speed or torque are not exceeded, the controlling program knows a stepper's precise position at any given time.

Steppers are driven by the interaction (attraction and repulsion) of magnetic fields. The driving magnetic field "rotates" as strategically placed coils are switched on and off. This pushes and pulls at permanent magnets arranged around the edge of a rotor that drives the output

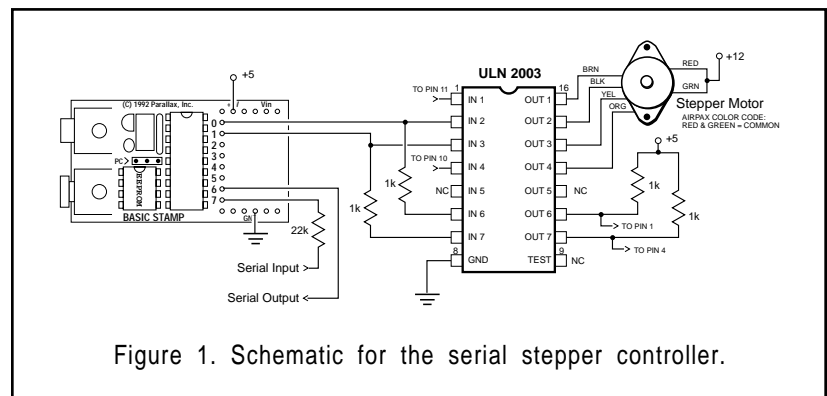


Figure 1. Schematic for the serial stepper controller.

shaft. When the on-off pattern of the magnetic fields is in the proper sequence, the stepper turns (when it's not, the stepper sits and quivers).

The most common stepper is the four-coil unipolar variety. These are called unipolar because they require only that their coils be driven on and off. Bipolar steppers require that the polarity of power to the coils be reversed.

The normal stepping sequence for four-coil unipolar steppers appears in figure 2. There are other, special-purpose stepping sequences, such as half-step and wave drive, and ways to drive steppers with multi-phase analog waveforms, but this application concentrates on the normal sequence. After all, it's the sequence for which all of the manufacturer's specifications for torque, step angle, and speed apply.

		Step Sequence				
		1	2	3	4	1
coil 1		1	1	0	0	1
coil 2		0	0	1	1	0
coil 3		1	0	0	1	1
coil 4		0	1	1	0	0

Figure 2. Normal stepping sequence.

If you run the stepping sequence in figure 2 forward, the stepper rotates clockwise; run it backward, and the stepper rotates counterclockwise. The motor's speed depends on how fast the controller runs through the step sequence. At any time the controller can stop in mid sequence. If it leaves power to any pair of energized coils on, the motor is locked in place by their magnetic fields. This points out another stepper motor benefit: built-in brakes.

Many microprocessor stepper drivers use four output bits to generate the stepping sequence. Each bit drives a power transistor that switches on the appropriate stepper coil. The stepping sequence is stored in a lookup table and read out to the bits as required.

This design takes a slightly different approach. First, it uses only two output bits, exploiting the fact that the states of coils 1 and 4 are always

the inverse of coils 2 and 3. Look at figure 2 again. Whenever coil 2 gets a 1, coil 1 gets a 0, and the same holds for coils 3 and 4. In Stamp designs, output bits are too precious to waste as simple inverters, so we give that job to two sections of the ULN2003 inverter/driver.

The second difference between this and other stepper driver designs is that it calculates the stepping sequence, rather than reading it out of a table. While it's very easy to create tables with the Stamp, the calculations required to create the two-bit sequence required are very simple. And reversing the motor is easier, since it requires only a single additional program step. See the listing.

How it works. The stepper controller accepts commands from a terminal or PC via a 2400-baud serial connection. When power is first applied to the Stamp, it sends a prompt to be displayed on the terminal screen. The user types a string representing the direction (+ for forward, - for backward), number of steps, and step delay (in milliseconds), like this:

```
step>+500 20
```

As soon as the user presses enter, return, or any non-numerical character at the end of the line, the Stamp starts the motor running. When the stepping sequence is over, the Stamp sends a new `step>` prompt to the terminal. The sample command above would take about 10 seconds (500 x 20 milliseconds). Commands entered before the prompt reappears are ignored.

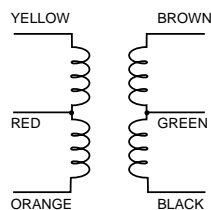


Figure 3. Color code for Airpax steppers.

On the hardware side, the application accepts any stepper that draws 500 mA or less per coil. The schematic shows the color code for an Airpax-brand stepper, but there is no standardization among different

brands. If you use another stepper, use figure 3 and an ohmmeter to translate the color code. Connect the stepper and give it a try. If it vibrates instead of turning, you have one or more coils connected incorrectly. Patience and a little experimentation will prevail.

```
' Program STEP.BAS
' The Stamp accepts simply formatted commands and drives a four-coil stepper.
Commands
' are formatted as follows: +500 20<return> means rotate forward 500 steps with 20
' milliseconds between steps. To run the stepper backward, substitute - for +.
```

```
Symbol Directn = b0
Symbol Steps = w1
Symbol i = w2
Symbol Delay = b6
Symbol Dir_cmd = b7
```

```
dirs = %01000011 : pins = %00000001 ' Initialize output.
b1 = %00000001 : Directn = "+"
goto Prompt ' Display prompt.
```

```
' Accept a command string consisting of direction (+/-), a 16-bit number
' of steps, and an 8-bit delay (milliseconds) between steps. If longer
' step delays are required, just command 1 step at a time with long
' delays between commands.
```

```
Cmd: serin 7,N2400,Dir_cmd,#Steps,#Delay ' Get orders from terminal.
if Dir_cmd = Directn then Stepit ' Same direction? Begin.
b1 = b1^%00000011
' Else reverse (invert b1).
```

```
Stepit: for i = 1 to Steps
' Number of steps.
pins = pins^b1
' XOR output with b1, then invert b1
b1 = b1^%00000011
' to calculate the stepping sequence.
pause Delay ' Wait commanded delay between
' steps.
next
Directn = Dir_cmd
' Direction = new direction.
```

```
Prompt: serout 6,N2400,(10,13,"step> ") ' Show prompt, send return
goto Cmd ' and linefeed to terminal.
```

Program listing: As with the other application notes, this program may be downloaded from our Internet ftp site at <ftp.parallaxinc.com>. The ftp site may be reached directly or through our web site at <http://www.parallaxinc.com>.