

A Stepper Motor Controller for a PCB Router/Engraver

Jonathan Westhues
jwesthue@uwaterloo.ca

1 Introduction

This kit is a stepper motor controller. It can control three unipolar stepper motors, one each for the x , y , and z axes. It communicates with a Windows PC over RS-232 serial. The stepping waveforms are generated in a microcontroller; the Windows PC sends line segments to the microcontroller, and the microcontroller turns a request to “move in a straight line 0.100” to the left and 0.500” forwards” into a sequence of steps to the x and y motors.

This allows us to generate precisely-timed waveforms from a Windows PC; Windows does not provide accurate timing, so we could not do this if we generated the waveforms on the PC. This is good, because it allows us to drive the stepper motors faster, by ramping up the speed slowly instead of starting abruptly from a dead stop.

This kit is designed to control unipolar stepper motors. The FETs used can switch up to 2 A/phase, running from a maximum of 40 V rails. This is not a chopper controller—the circuit powering the stepper coils is like a voltage source, not like a current source, so the torque will drop quickly with speed. This kit is therefore most suitable for low-speed, high-torque applications. This kit is capable of switching reasonably high voltages and currents, however, so it may also be used as an L/R drive, by adding power resistors (off-board) in series with the stepper coils.

2 Assembling the Kit

This kit includes a professionally-fabricated single-layer printed circuit board with solder mask. It is built with all coarse-pitch through-hole components, so it should be reasonably easy to assemble.

This kit needs a power supply. You have two options for powering the kit: you can connect a PC power supply, using the Molex (hard drive power) connector SV8, or you can connect a general-purpose DC power supply using the terminal block SV1:

Molex connector, for PC power supply. The PC power supply can deliver high current at +5 V and +12 V. The 5 V supply will be used for the microcontroller. You can run your stepper motors at either 5 V or 12 V; populate wire link JP1 for 12 V, or populate wire link JP2 for 5 V. Do not populate both jumpers; this will damage the power supply. Do not populate the voltage regulator IC1; it is not necessary. Populate power resistor R16, to meet the power supply's minimum load current requirements. It is normal for R16 to run very hot whenever the kit is powered; take care not to touch it when working with the kit, and do not mount it near or in contact with flammable materials. If your power supply shuts down when you energize your stepper motors and you are running from the 12 V rail then try purchasing a smaller 5 W power resistor, to a minimum of 5 Ω .

Screw terminals, for other DC power supply. Choose a power supply whose voltage and current rating matches that of your steppers; for example, if your steppers are rated for 1 A/phase at 30 V, then choose a 30 V, 6 A power supply (three stepper motors, with two phases energized at any given time). The supply voltage must be between 7.5 and 40 VDC. Populate the voltage regulator IC1. Do not populate either jumper JP1 or JP2, and do not populate power resistor R16.

The kit ships with only the components for the first option. If you wish to populate the kit for the second option then purchase a TO-220 package 7805 voltage regulator. This is a widely available part. The components are as follows:

Designator	Value	Description
C1	1000 μF	capacitor, electrolytic
C2	10 μF	capacitor, electrolytic
C3, C5	22 pF	capacitor, ceramic
C4	0.1 μF	capacitor, ceramic
D1–14		diode, 1N4004
IC1		voltage regulator, 7805 (not enclosed)
IC2		PIC16F877 microcontroller
JP1, JP2		wire jumpers
LED1		LED indicator
Q1–12		power n-MOSFET, Toshiba 2SK2231
Q13		NPN bipolar transistor, 2N4401
R1–12, R17	1k	resistor, 1/4 watt
R13	10k	resistor, 1/4 watt
R14–15	330 Ω	resistor, 1/4 watt
R16	5 to 10 Ω	power resistor, 5 watt
XT1	4 MHz	crystal, HC-49
SV1		3-position terminal strip (not enclosed)
SV2–4		5-position terminal strip
SV5, SV7		2-position .100" header
SV6		6-position .100" header
SV8		Molex hard drive power connector
SV9		5-position .100" header (not enclosed)
SV10		6-position .100" header (not enclosed)

Use rosin-core solder and a soldering iron with a fine tip. The board has a silkscreened legend to indicate component placement. Pay attention to component polarization; use the legend as a guide.

Populate the FETs Q1–12 with the tab above the white bar, the diodes with the white band above the white band, and the polarized capacitors with the plus sign on the same side as the plus sign silkscreened on the board. Place the longer lead of LED1 in the hole marked with a plus sign. If you are not powering the kit from a PC power supply, populate voltage regulator IC1 with the tab above the white bar.

Do not solder the PIC16F877 microcontroller directly to board; solder the 40-pin DIP socket, and install the pre-programmed PIC in the socket. Align the notch in the socket with the semi-circular jog in the silkscreened outline, and align the notch in the PIC with the notch in the socket.

The resistors are colour-coded as follows:

330 Ω	orange-orange-brown-gold
1 k Ω	brown-black-red-gold
10 k Ω	brown-black-orange-gold

The screw terminal blocks feature a dovetailed protrusion on one side and a dovetailed slot on the other; assemble five 3-terminal blocks together for the connections to the motor (SV2–4). Note that there are no heat sinks on the

power transistors; none are necessary. The connectors SV9 and SV10 are for future expansion and for in-circuit programming; they are not necessary for normal operation of the kit.

Apply power to the controller. The LED should flash at startup, and then turn off. Connect the controller to your PC on COM1. Ensure that you connect the serial dongle correctly; match the digit '1' molded into the female connector with the '1' silkscreened on the board. Run `serialrouter` with no command line arguments. It should connect to the controller; you should see the text

```
found controller
```

followed by some instructions. Connect a stepper motor to the x axis; if you are not sure how to do this then see the appendix. Press `w` in `serialrouter`. The stepper motor should spin. Press `x`; the motor should spin in the opposite direction.

If the LED does not flash at startup then the PIC is not running. Verify that it has correct power (+5V on pins 32 and 11), and verify that the oscillator circuit, XT1, C7, and C8, is connected correctly. If the LED does flash, but you cannot connect with `serialrouter` then the problem is with the serial connection; check the connections to the serial dongle.

If you can connect to the device with `serialrouter` but the motor vibrates instead of rotating, check that resistors R1–4 and FETs Q1–4 are correctly installed. Also check that the phase sequence to the stepper is correct.

Once the controller is working, test the y and z axes and make corrections as needed.

If you have limit switches on your milling machine, connect them in parallel across SV6. When any one of the limit switches closes, the two pads will be shorted together; the controller will sense this and stop the motors. The controller will flash the LED quickly to indicate an error condition, and it will remain in this state until you cycle power to it.

It is convenient to be able to automatically turn off the spindle motor (router, Dremel tool, etc.) when the job is complete. If you have a contactor (relay) that you wish to use to control your spindle motor, connect its control terminals across SV7. Verify that the high-voltage rail that you are using for the steppers is also suitable for your contactor. For example, if we had a 6 V relay with an $18\ \Omega$ coil then the coil current should be $6/18 = 333\ \text{mA}$. If we were using a 20 V rail for the stepper motors, then the relay coil would also get 20 V, so a current of $20/18 = 1.11\ \text{A}$ would flow. This would probably destroy the relay.

To fix this add a power resistor in series with the coil. For example, in this case an additional resistance of $18 \times (20 - 6)/6 = 42\ \Omega$ would increase the total resistance to $42 + 18 = 60\ \Omega$, so that the current would be $20/60 = 333\ \text{mA}$, as desired.

If spindle motor is switched on, but the contactor is off, then the machine is not safe! A software error could cause the spindle to be energized unexpectedly, resulting in serious injury. Arrange some way to positively disconnect power from the spindle when you are setting up work, changing tools, etc.; for example, consider unplugging it.

3 Software Settings

All of the software supplied with this kit is designed to be run from a command prompt! To get a command prompt, select “Start,” “Run,” and type either `command` (Windows 95, 98) or `cmd` (Windows 2000, XP) and press the Enter key. You should then change to the directory where you have saved the controller software (`serialrouter.exe`, `winrouter.exe`). For example, if you saved them in the `pcb` directory on your C drive then you would type

```
C:\Documents and Settings\Jonathan Westhues>cd \pcb
```

```
C:\pcb>
```

The controller must be configured according to the characteristics of the machine that it will control. All software settings are made using environment variables. There are two ways to set environment variables under Windows. You can use the `set` command at the command prompt:

```
C:\pcb>set NAME=value
```

If you set an environment variable this way then it will not persist across resets. You can make a setting persistent by putting it in your `autoexec.bat`. Alternately, for Windows 2000 and later, right-click “My Computer,” select “Advanced,” and click the “Environment Variables” button.

Scale factor. The description of the board to mill consists of many polygons, whose dimensions are given in millimetres. The controller must, for example, know how many steps the stepper motor must move through in order to move exactly 2 millimetres. This will depend on the design of the stepper motor used (degrees per step), and the pitch of the leadscrew that it drives.

For example, let us say that our stepper motor is rated at $7^{\circ}30'$ per step, and it is driving a $1/4''-20$ leadscrew. This means that we have a scale factor of

$$\frac{1 \text{ inch}}{20 \text{ turns}} \times \frac{1 \text{ turn}}{360^{\circ}} \times \frac{1000 \text{ thou}}{1 \text{ inch}} \times \frac{7.5^{\circ}}{1 \text{ step}} = 1.04 \frac{\text{thou}}{\text{step}}$$

The control software wants the scale factor in units of thou per step; the scale factor need not be an integer (which is good, because in this case it is not):

```
c:\pcb>set ROUTER_THOU_PER_STEP=1.04
```

Backlash compensation. Many improvised linear motion systems will have significant backlash; for example, ordinary V-threaded rod turning in a brass nut might have 5 steps of backlash: this would mean that you could move the nut by 5 steps in the direction of linear motion without rotating the screw with respect to the nut.

We can correct this by adding an extra 5 steps to our displacement whenever we change direction. Specify the amount of backlash compensation as follows:

```
c:\pcb>set ROUTER_BACKLASH=5
```

Note that the units here are steps, not an absolute unit of length! (like inches or millimetres)

You can measure the backlash in your machine with a dial indicator. Set up a dial indicator so that it measures the linear displacement of the carriage on the axes to be measured. Jog the carriage a short distance forward, to take up the backlash. Then, jog the carriage backwards one step at a time (use **Shift+Key**), counting the steps until the dial indicator shows the carriage beginning to move. This gives you the `ROUTER_BACKLASH_IN_STEPS` value.

Motor Top Speed. A stepper motor can only run so fast before it starts losing steps. If one of the motors loses a step then it loses its position reference and cannot get it back, so the board in progress is ruined. The motor speed is configurable with an environment variable. Note that this configures only the speed when milling from a board layout; the jog speed is fixed. Use the `t` command at the jog screen to test the speed.

To configure the speed, set the following variable:

```
c:\pcb>set ROUTER_SPEED=1
```

Try 1 as an initial guess, and increase or decrease it as needed.

Milling Depth. The copper foil on a PCB is extremely thin, but usually we will want to mill much deeper than the thickness of the copper. For example, we may want to compensate for variations in the height of the table, so that we might be milling needlessly deep in the high spots but we still break through the copper in the lowest spots. We can configure the milling depth as follows:

```
c:\pcb>set ROUTER_DEPTH=15
```

This is in units of steps.

Drilling Depth. As for the milling depth.

```
c:\pcb>set ROUTER_DRILL_DEPTH=125
```

This is in units of steps. It must be less than 250.

Time for router to come up to speed. This kit can switch power to the spindle motor by driving an external contactor. We must wait for the spindle to come up to speed before attempting to make a cut, however. This environment variable sets the time, in milliseconds, that the software will wait between energizing the contactor and taking the first cut.

```
c:\pcb>set ROUTER_SPIN_UP_TIME=4000
```

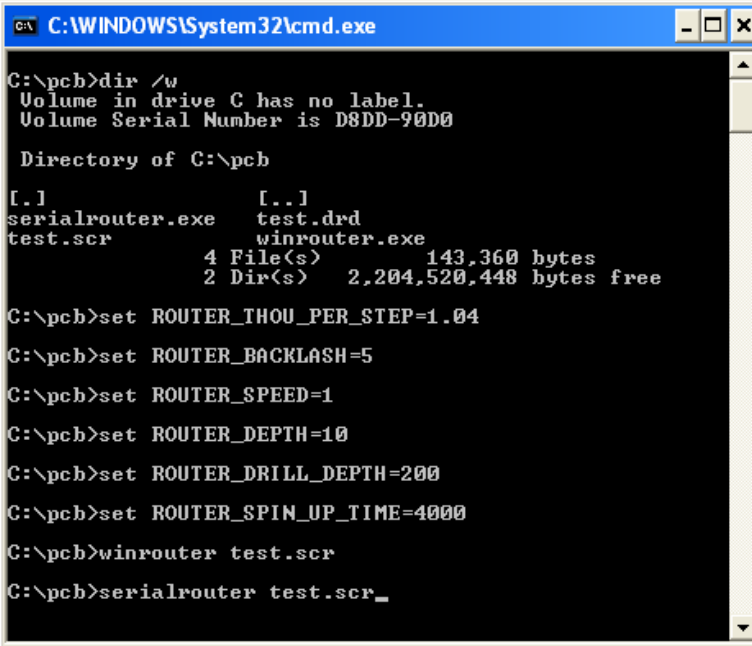
If you are not using an external contactor then this setting is not relevant; apply power to the spindle manually, and wait for it to spin up before you start milling.

HPGL scale fractor. This software can work from either EAGLE script (`.scr`) files or HPGL (`.plt`) files. This setting only applies to HPGL files. Usually, but not always, HPGL files are specified with units of 1/40 mm, so that we should:

```
c:\pcb>set ROUTER_HPGL_UNIT_PER_MM=40
```

However, we can change this if an HPGL file was generated with different settings, or simply to scale the image to the desired size.

Once you have made all these settings, you are ready to run the tools (`winrouter`, the previewer, and `serialrouter`, the controller). Your command prompt window, with everything set up, should look something like this:



```
C:\WINDOWS\System32\cmd.exe

C:\pcb>dir /w
Volume in drive C has no label.
Volume Serial Number is D8DD-90D0

Directory of C:\pcb

[.]          [..]
serialrouter.exe  test.drd
test.scr         winrouter.exe
                4 File(s)      143,360 bytes
                2 Dir(s)    2,204,520,448 bytes free

C:\pcb>set ROUTER_THOU_PER_STEP=1.04
C:\pcb>set ROUTER_BACKLASH=5
C:\pcb>set ROUTER_SPEED=1
C:\pcb>set ROUTER_DEPTH=10
C:\pcb>set ROUTER_DRILL_DEPTH=200
C:\pcb>set ROUTER_SPIN_UP_TIME=4000
C:\pcb>winrouter test.scr
C:\pcb>serialrouter test.scr_
```

4 Milling PCBs

Generating the Toolpath. The instructions here are given for EAGLE; EAGLE is convenient, because it ships with a script to generate a toolpath from a board layout. First, open the layout in the board editor. From the “File” menu, select “Run.” Choose `outlines.ulp`. Choose the following settings:

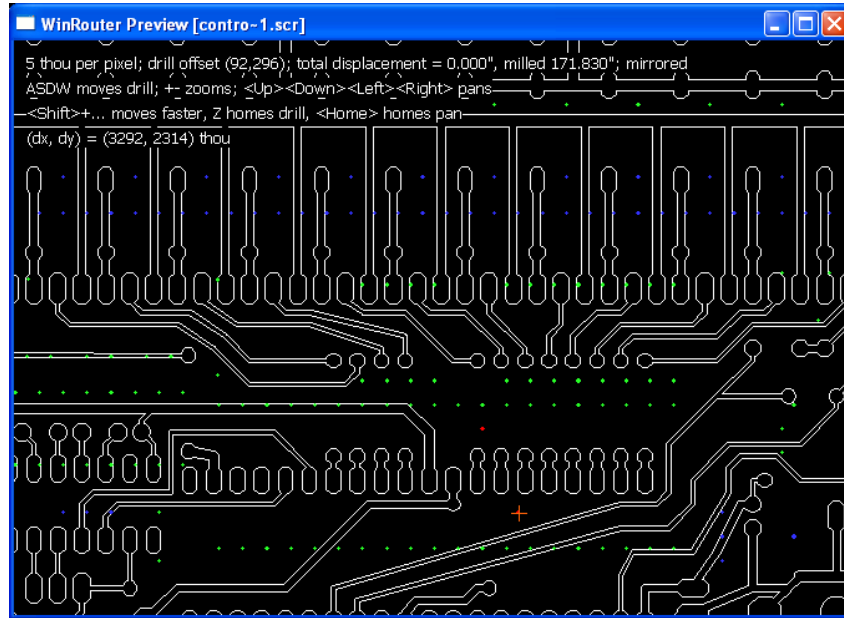
- Device: Script
- Width: the width of the trench your cutter cuts, in millimetres
- Layer: bottom, for a single-sided board

The cutter width would typically be around 0.3 mm.

Specify an output file in the same directory in which you saved the two programs, `serialrouter.exe` and `winrouter.exe` (`c:\pcb` in the example above) and click “OK.” Then create an Excellon drill file (“CAM Jobs,” from the “Control Panel” window). Place this in the same directory, so that you have `proj.scr` and `proj.drd`. Then, run from the command line

```
C:\pcb>winrouter proj.scr
```

This will display something like what is shown below:



Note that the holes (coloured dots) will not be aligned with the traces; use the ASDW keys to fix this. Verify that everything appears as it should be, and take note of which hole appears not just with a dot but also with a red crosshairs; this hole will be drilled first. You can move the mouse pointer and look at the coordinates shown at the top of the screen; verify that the distances between some features are correct, both in units of steps and (press *i*) inches.

Also note that the board is shown mirrored; for the solder side, this is what we want.

Now that you have the two files required to mill the board. Connect the controller to your computer and apply power. Now run the controller software on the PC:

```
C:\pcb>serialrouter proj.scr
```

Then follow the prompts. You will start out at the jog screen, which allows you to position the table or gantry on your milling machine. The commands are as follows:

```
w,s,x -- x axis forward, stop, reverse  
W, X -- move x axis one step forward, reverse  
e,d,c -- y axis forward, stop, reverse  
E, C -- move y axis one step forward, reverse  
r,f,v -- z axis forward, stop, reverse
```



```
R, V -- move z axis one step forward, reverse
j    -- disable the r and v commands
qp   -- exit jog screen (type both letters)
```

Note the `j` command; this is useful, because it is easy to destroy a tool by inadvertently pressing `r` or `v` and forcing the stationary tool into your workpiece. After pressing `j` you can only move the z axis one step at a time, which makes mistakes less likely. Also note that a two-letter sequence is required to exit the jog screen and start milling; again this is intended to decrease the destructive potential of typos.

Before you start milling the board you must position the tool so that it is just above the workpiece. The easiest way to do this is probably to place a thin shim (a scrap of aluminum foil, for example) between the tool and the workpiece, and lower the tool one step at a time (using the `R` command) until you feel a drag on the foil when you slide it back and forth. Then position the cutter appropriately in the x and y directions. If the power to your spindle is not under computer control (that is, you are not controlling it with a contactor controlled by this board) then turn it on now. Type `qp` to start milling.

Once the software finishes milling the board, it will return to jog mode. At this point you can stop the spindle and change tools from a milling cutter to a drill bit. Then position the drill bit directly over the centre of the pad that was marked with a red crosshair in `winrouter`. Again lower the tool until it is just barely touching the workpiece. Type `qp` to start drilling.

When the software finishes drilling, it will again enter the jog screen. This gives you an opportunity to lift the tool out of the way and move the table (or gantry etc.) to remove the finished board.

To go to the jog screen without actually milling a board, run the serial control program with no arguments:

```
C:\pcb>serialrouter
```

5 Milling from an HPGL File

HPGL stands for “Hewlett-Packard Graphics Language.” It is a vector graphics format—it is a set of lines, like an AutoCAD `.dxf`, not a set of pixels like a `.gif`. It was originally designed to control pen plotters. It is important to us because many graphics packages can produce HPGL files; for example, CorelDRAW can convert text in a TrueType font to an HPGL file, so you can use it to engrave text.

There is a later variant of HPGL called HPGL/2. This is a binary format, not a text format. This software does not support HPGL/2.

If you have an HPGL file, you can engrave it with this software. Save it to the same directory where you saved `serialrouter.exe` and `winrouter.exe`, and then run:

```
C:\pcb>winrouter file.plt
```

The file must end in .plt! This is how the software knows that it is HPGL. Verify that it is displayed correctly, and then engrave it:

```
C:\pcb>serialrouter --nomirror file.plt
```

Note that we have specified `--nomirror` here. For a PCB we are usually working with the top view of the bottom of the board, so we must mirror it when we mill it. We probably don't want to mirror our HPGL file, so we explicitly tell `serialrouter` not to. In `winrouter`, press `m` to toggle whether the toolpath is shown mirrored.

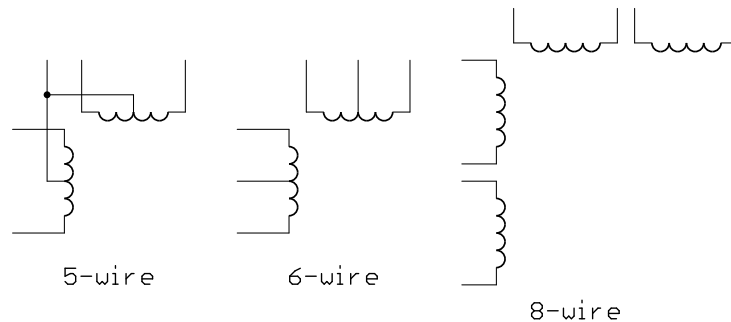
All of the settings discussed above (for depth, speed, etc.) apply to HPGL files. Remember to set the `ROUTER_HPGL_UNIT_PER_MM` environment variable.

May 2004, Waterloo

Appendix: Stepper Motors

A stepper is a kind of electric motor. We can control the position of the stepper motor's shaft very exactly; for example, we can tell it to rotate through exactly 75° clockwise. This is good, because if we keep track of the instructions that we have sent to the stepper then we always know exactly where it is, at least relative to its starting position. However, if we tell the stepper motor to move too fast then it will not be able to keep up; this is bad, because we will lose our position reference and never recover.

There are many different kinds of stepper motors. This kit is designed to control unipolar stepper motors. A motor suitable for use with this kit will effectively have two centre-tapped coils; they might be connected in one of three ways, so that your motor might have 5, 6, or 8 wires:



The kit has five terminals per stepper, so we need the motor connected as in the leftmost schematic. Note that the 6- or 8-wire configurations can be made to look like the 5-wire configuration by connecting multiple wires together as the common. The common terminal for each axis is marked with a white mark silkscreened on the board.

If you buy new steppers then they should come with a datasheet that will say how they are connected. If you have surplus or salvaged steppers, however, then you will have to figure out the wiring yourself. An ohmmeter is useful for this; the coils have some DC resistance, so in a 5-wire motor, for example, the resistance from a phase to another phase will be twice the resistance from a phase to common.

If you get the phase sequence wrong then the stepper may vibrate back and forth instead of rotating, or it might rotate backwards. If one of your x or y axes rotates backwards then everything you mill will be mirrored.

There is an excellent tutorial on stepper motors on the Internet at

<http://www.cs.uiowa.edu/~jones/step/>

