

Physics 219 – Fall, 2005  
LabNotes 10

Print Your Own Circuit Board

Table of Contents

**Wednesday, November 16** ..... 1

    Personal Fabrication ..... 1

        Fabricate your own printed circuit board ..... 4

    Clone Your Own LogoChip ..... 4

    Make a cloner board using Eagle Layout Editor ..... 6

        1) Draw the schematic ..... 7

        2) Creating the board ..... 8

        3) Place the parts ..... 9

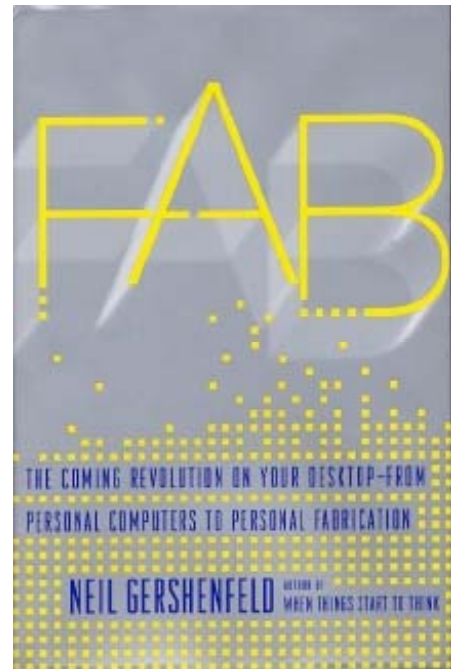
        4) Routing the signals ..... 12

    Tips and Tricks ..... 12

**Wednesday, November 16**

**Personal Fabrication**

First, the big idea: In the same way that personal computers have already revolutionized and “democratized” the production of things ranging from written documents (think Microsoft Word and laser printers), to movies (think digital camcorders and iMovie), to journalism (think blogs and podcasts), are we on the threshold of another revolution? For example, what if you could print three dimensional physical objects with the same ease that I printed these notes? The implications of this vision are well-summarized in the following *Scientific American* column that reviewed a new book called *FabLab*, which was recently by MIT Media Lab professor Neal Gershenfeld:



*Thirteen years ago I unboxed my new Apple Macintosh, plugged it into the phone line, and discovered the existence of another world. Spirited, unruly discussions on everything from quantum physics to punk rock ebbed and flowed across a borderless electronic forum called Usenet. Anyone anywhere could join in. More definitive sources of information--how to combat an infestation of pine-tip moths, join two boards with a dado joint or locate the great nebula in Orion--resided among a far-flung collection of computers called Gopher servers, a precursor to the World Wide Web. So much had been happening beyond my awareness. I felt like an African bushman turning on a radio for the first time. It wasn't just words and pictures that had been lurking out there. With the chirps and squawks of modem tones, I could download animated clocks, perpetual calendars, a gizmo that made my keyboard clack and ding like an old Smith Corona typewriter. Legions of amateur programmers were creating and distributing, largely for their own amusement, a multitude of virtual machines. I hadn't thought of it this way until I read Neil Gershenfeld's new book, *Fab: The Coming Revolution on Your Desktop--From Personal Computers to Personal Fabrication*, but I was witnessing the revival of a spirit that had been fading since the Industrial Revolution: that of the artisan.*

*While corporations like Microsoft and Oracle were employing droves of programmers to homogenize products for the mass market, these technological craftsmen were working on a personal scale. Crafting their code in home workshops, they enjoyed the same satisfaction that comes from building a bookshelf or caning a chair. Gershenfeld, director of the Center for Bits and Atoms at the MIT Media Lab--the futuristic name is quintessential M.I.T.--believes that what is true now for virtual commodities will soon apply to physical ones. Give people personal computers and they can write their own software. Give them devices called personal fabricators and they can make their own things. What this will mark, he predicts, is a return to the days before "art became separated from artisans and mass manufacturing turned individuals from creators to consumers."*

*Turning the pages, I could barely wait for the revolution to begin. With a smattering of Unix, I have been able to custom-tailor my own virtual machinery--an algorithm that checks in hourly with Amazon, recording the sales rank of my newest book; another that intercepts unwanted e-mail press releases, dispatching to persistent senders increasingly testier replies. But what about more solid stuff, like the knob that broke off the toaster? Or, even more annoying, all the extraneous, cryptically labeled buttons cluttering the TV remote control, when all I really want is On, Off, Channel, Volume and Mute? With mouse and keyboard, I could describe my needs to a personal replicator, hit enter, and wait for the*

*product to emerge. If it wasn't quite right, I could tinker and try again. If someone else wanted to make one, I could post the code--the input for the fabricator--on my Web site or e-mail it to friends. The physical world, Gershenfeld promises, will become as malleable as the digital world, and we will no longer have to settle for the imperfect cobbling together of compromises available at the mall.*

*It was a little disappointing to learn that for now personal fabricators are actually rooms full of expensive equipment called "fab labs." But be patient: a few decades ago a computer equivalent to a laptop weighed tons. In a class Gershenfeld teaches called "How to Make (Almost) Anything," laser cutters, water-jet cutters, numerically controlled milling machines--the kind of tools used in CAD-CAM (computer-aided design and manufacture)--give students the feeling of mastery that comes from taking an idea into the real world. Industrialists use this equipment to make prototypes, exact replicas of items they intend to manufacture. In the fab labs, as Gershenfeld puts it, the prototype is the product. Each is designed for a customer base of one. A student who had trouble getting up in the morning made her own fiendish alarm clock. Silencing it required touching a series of sensors in exactly the right order, a task certain to rouse her awake. A visitor to the lab, the actor Alan Alda, fabricated an accessory for his digital camera: a flash periscope that raises the bulb high enough that his subjects don't come out looking like red-eyed children of the damned. Even when a fab lab can be shrunk to the size of a suitcase, most people will probably content themselves with what is offered at Wal-Mart, just as they do with what's on TV.*

*Where the revolution seems likelier to find traction is in the developing world. The best parts of Gershenfeld's book describe his adventures setting up experimental fab labs in places like Ghana and India, encouraging locals to try making tools that are unavailable or unaffordable: portable solar collectors that can turn shafts and wheels, inexpensive electronic gauges farmers can use to measure the quality of their crops, giving them an edge when they haggle with the brokers. All this may sound utopian, but it is hard not to be taken with Gershenfeld's enthusiasm. Today we have open-source software--all these free Unix and Linux programs streaming through the Net. Imagine a world with open-source hardware. Come up with a really great product, and you can share it with the world--to be hacked and modified by the people who actually use it, warrantied against obsolescence by the irrepressible nature of human ingenuity.*

By George Johnson, from the June, 2005 *Scientific American*,

### **Fabricate your own printed circuit board**

As a modest first step towards this vision of personal fabrication, we'll show how you can draw a schematic for an electronic circuit on a computer, hit "print" and have a **printed circuit board** (or "pcb") for the circuit, all ready for soldering electronic components on it, appear in my mailbox a few days later.

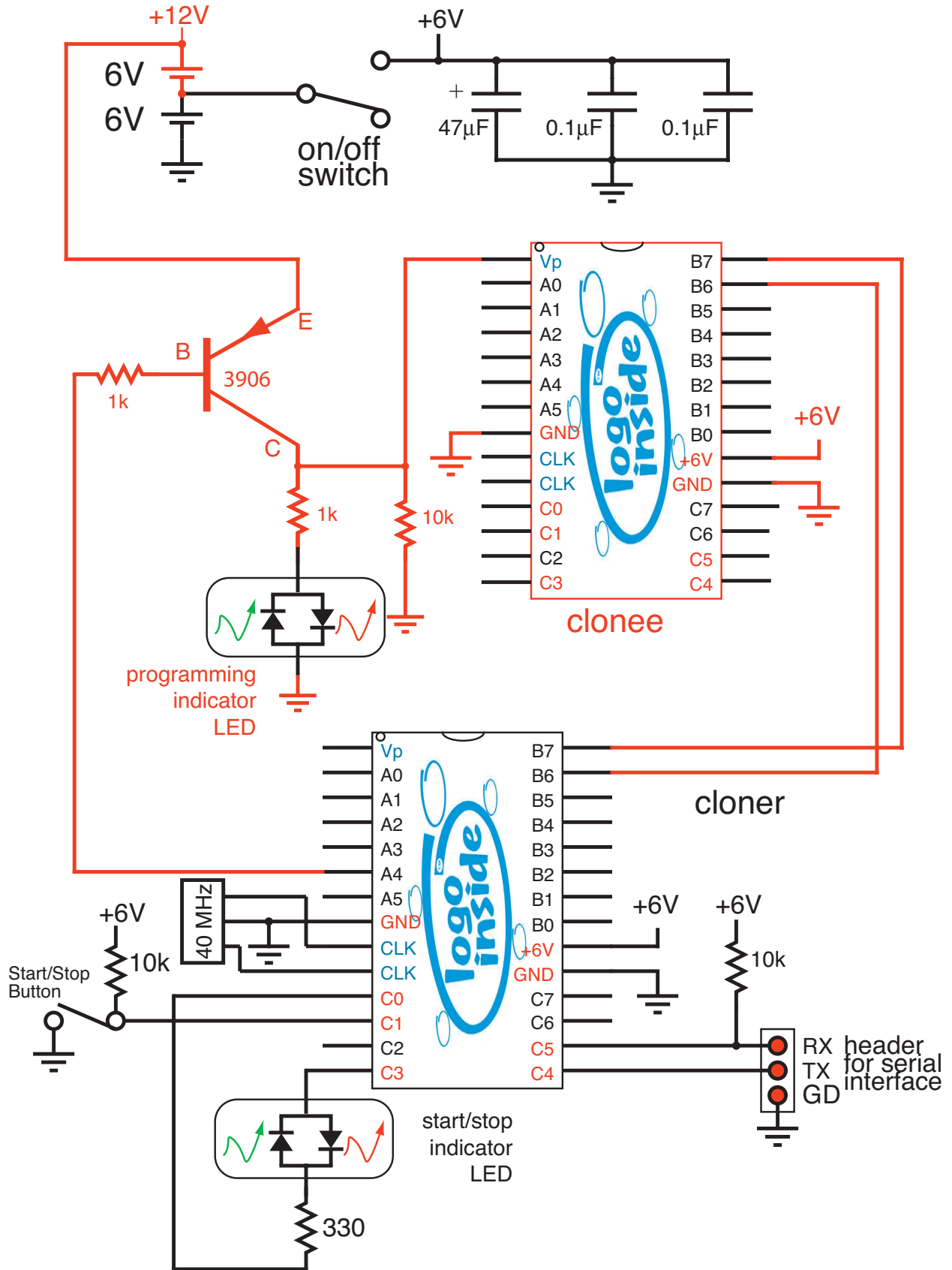
I don't want to over-sell the current state of the pcb fabrication process. The software for laying out a circuit board has a steep learning curve and is initially fairly frustrating to use. And the "printing" takes place at a professionally operated facility (in Canada, in our case) and costs about \$100 to make a few boards. (Then again, you should have seen what personal computers were like in 1980. It's possible that 10 years from now this process will be *much* simpler.) But I think it'll be a worthwhile experience for you to get a sense of the current state of affairs. To make the process manageable, we'll each try to make a board for the same circuit and send out for manufacture just one instance of the board design that seems to have come out particularly well.

What circuit board should we make? For this exercise I've selected something that seemed useful, instructive and (hopefully) doable in a single afternoon: a LogoChip "cloner board".

### **Clone Your Own LogoChip**

It turns out that once you have one LogoChip you can easily create more through a "cloning" procedure in which the LogoChip "firmware" copies itself onto a blank PIC18F2320 microcontroller.

To clone a LogoChip, we simply need to add the circuitry shown in red on the schematic below to an existing LogoChip. The new circuitry consists of a "pnp" transistor such as a "3906", a second 6 volt battery pack, and an LED (a bi-color LED like the one used for the LogoChip indicator light will do) and a 1k and a 10k resistor.



To program a blank PIC18F2320 it is necessary at times during the process for pin 1 of the chip (labeled  $V_p$ ) to be held at 12 V. (This requirement makes it hard to reprogram by accident!) The programming takes place via two lines connected between the “cloner” and the “clonee”. The process, in which one line carries the “data” and the other provides the “clock”, is very much like the I<sup>2</sup>C protocol that you saw last time.) The 12 V programming voltage is switched on and off by the cloner via pin A4, which is connected to the base of the pnp transistor. (Pin A4 has an “open collector” output, like you saw on the 311 comparator, which is necessary when connecting to voltages greater than 12 V.)

When the new circuitry has been added, you can download the program called “clone.txt” (which is included in the same folder as the LogoChip software) onto the existing LogoChip. The clone.txt program contains a startup procedure which causes all of the LogoChip’s program memory (including the clone.txt Logo program itself!) to be copied to the blank PIC18F2320 microcontroller. Simply pressing the START button initiates the process, which takes about one minute (assuming the LogoChip is running at 40 MHz). If all is well, the programming indicator LED included in the new circuitry should turn on during the cloning procedure and the voltage on the pin 1 of the blank PIC18F2320 should rise to a little less than 11 V. As soon as the LogoChip’s green run light turns back to red, you’re done. You can test to see if the cloning was successful by replacing the old LogoChip with the new one and making sure it works properly.

### **Make a cloner board using Eagle Layout Editor**

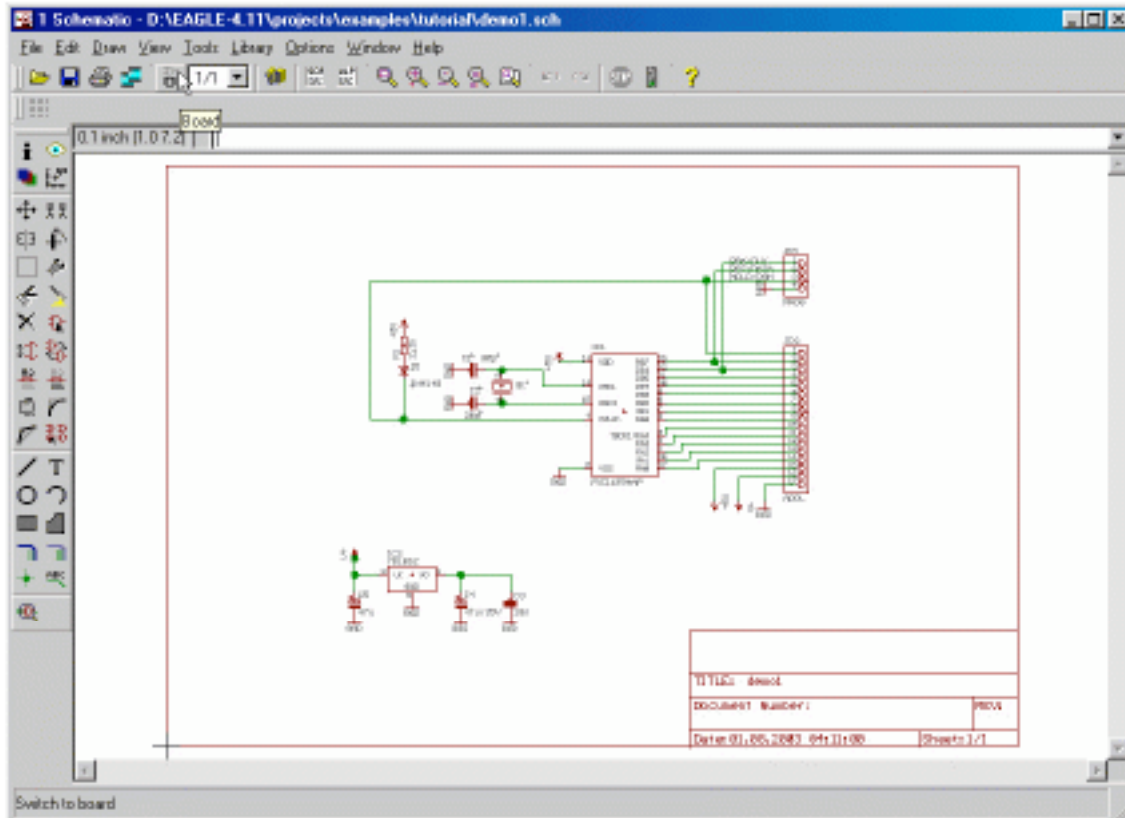
We’ll try using the Eagle Layout Editor software package from Cadsoft to create a schematic drawing of the cloner circuit. Then we’ll use the Eagle software to translate the schematic into a **layout** for the printed circuit board. The layout is produced in a form that can be shipped electronically to a manufacturer in Canada (AP Circuits) that specializes in making printed circuit boards for prototyping purposes. They’ll make the boards we’ve designed within about 24 hours and then FEDEX them back to us.

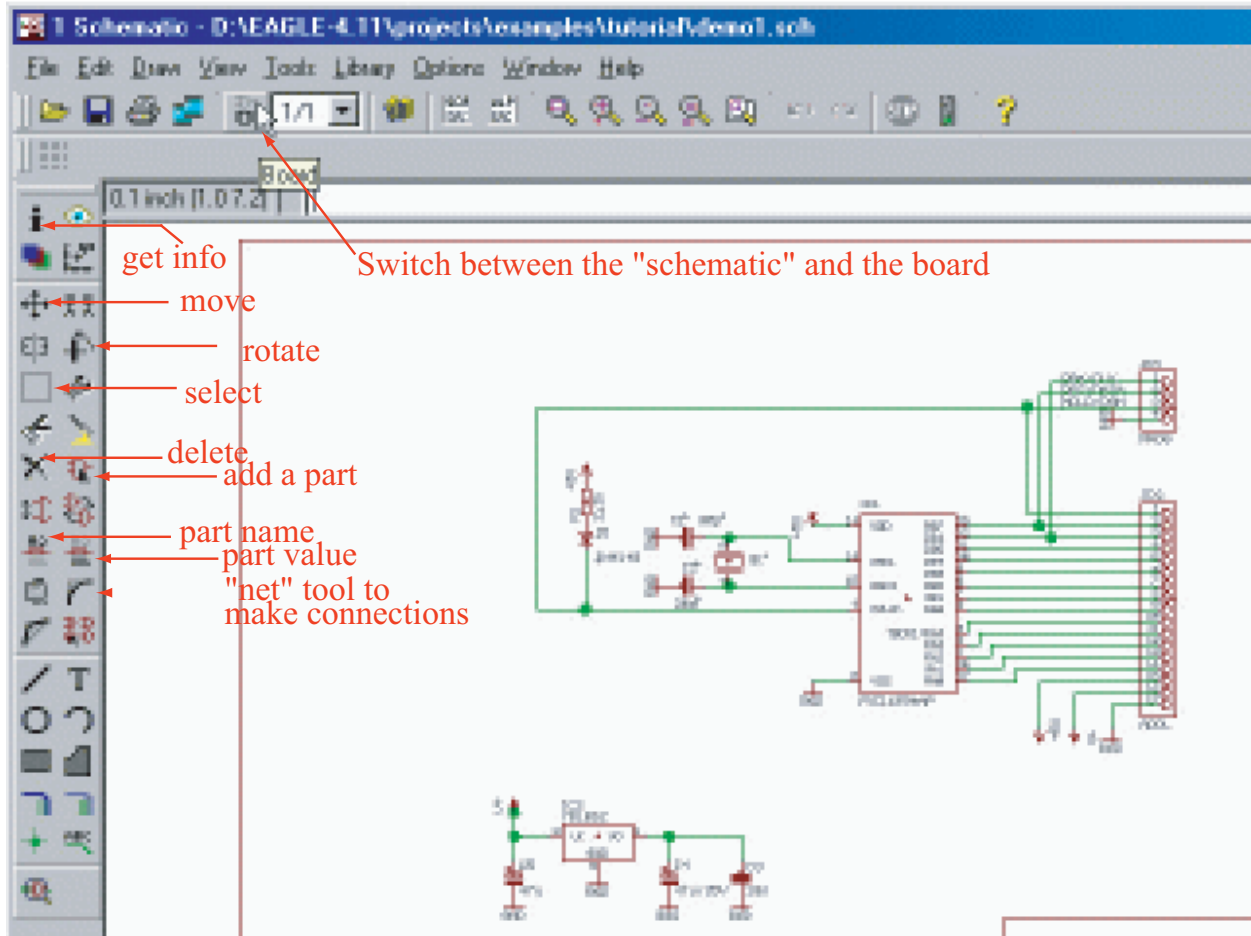
The process of learning to use software like this is best learned in a hands on tutorial format, which we’ll spend an afternoon on. Hopefully we’ll end up with something to ship by the end of the day.

Below are shown some screen shots that summarize the process:

### 1) Draw the schematic

When Eagle is opened, select “New Project” from the Control Panel’s file menu and then “New Schematic”. This opens the schematic editor, which is shown in the screen shot below.





### Key tools for drawing a schematic

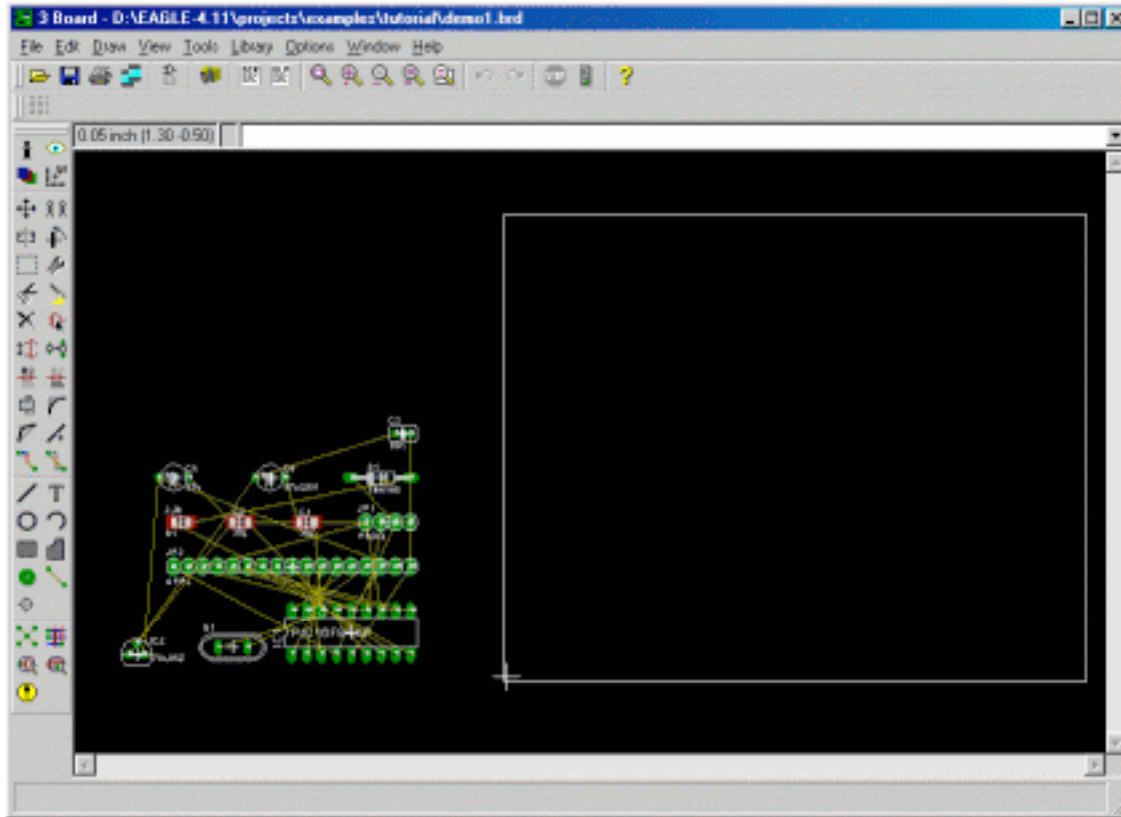
- Create a schematic drawing by using the “add parts” tool to add new electronic components to a drawing. The parts exist in a special library (called 219.lbr) that contains all the parts you’ll need to construct the cloner board circuit. (If you can’t find the 219.lbr library when you click on the “add parts” tool, select “Use” from the schematic editor’s library menu and navigate to 219.lbr, then retry using the “add parts” tool.)
- Use the “net” tool to make connections between the pins on the parts, following the schematic drawing for the cloner board given on page 5 above.

### 2) Creating the board

Creating the board from a schematic is largely automated by the Eagle software.

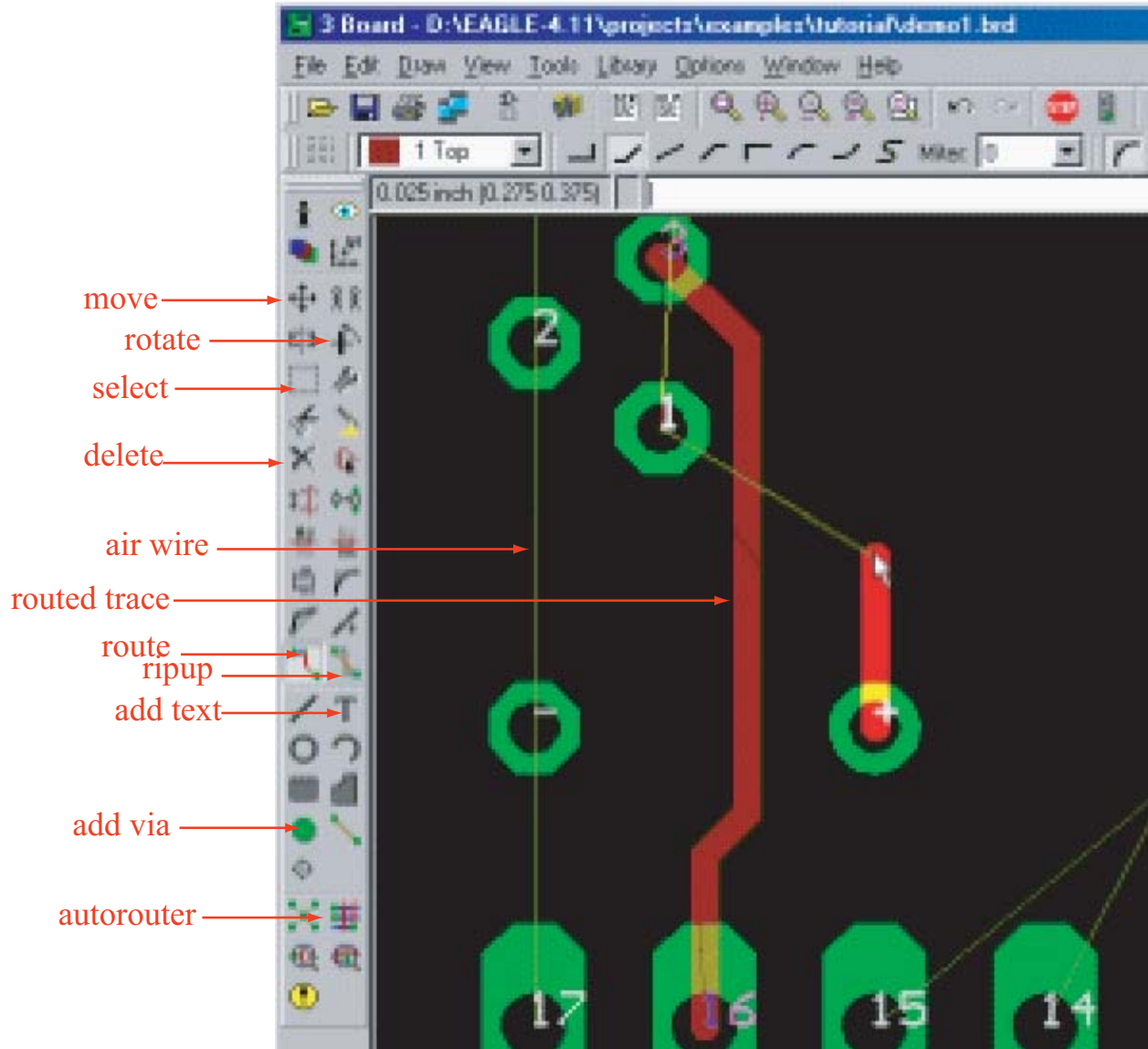
The Board command creates a new window with all the parts arranged next to a

default board outline. Each time you add a part in the schematic a new part is automatically added to the board. All the nets from the schematic are shown as **airwires**.

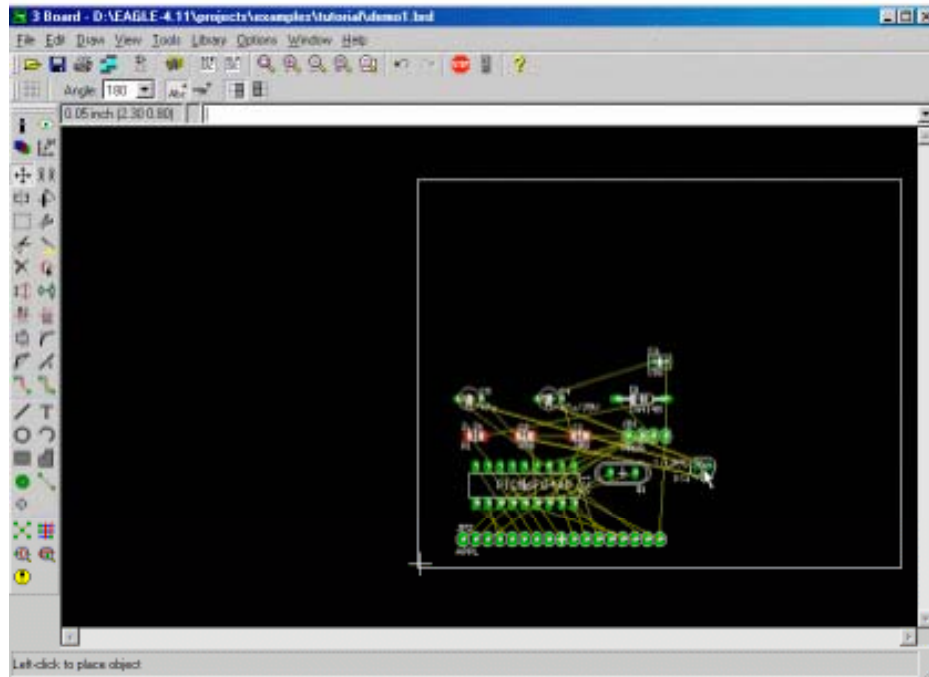


### 3) Place the parts

Use the select, move and rotate tools to place the parts within the confines of the white square (which represents the boundaries of the circuit board). As you move the parts the airwires will stretch and twist like rubber bands, maintaining the desired connections.



**Key tools for laying out a board**



### Tips and Tricks for Placing Parts

Layout design is an art honed through much practice, so don't be discouraged during this first attempt. Here is some advice:

- Try to be neat. Try to place “related parts” near each other and use the rotate tool to minimize twisted airwires. (You can learn a lot by looking at some professionally designed boards. Look, for example, at how nicely things line up.)
- Place parts in “logical” locations. the start/stop indicator should be near the start/stop button for example.
- Leave a “reasonable” amount of space between parts. Too far apart leads to big boards; too close and it will be topologically impossible to make all the necessary connections.
- The selection point on a part is located at the “white cross”. When using tools such as rotate, delete or move, click on the cross of the part on which you want to operate.
- Save all frequently!! In my experiences Eagle crashes occasionally.
- All parts must be placed inside the “white square”, which represents the

physical boundaries of the board that will be produced. Do not *increase* the size; it represents the maximum board size allowable for the freeware version of Eagle. You should on the other hand feel free to decrease its size; the cloner board should easily fit into a smaller area.

#### 4) Routing the signals

Manual **routing** (turning the airwires into actual traces) is done with the **route** tool.

Once you select the route tool, you simply pick up an airwire, adjust the angle with the right mouse button. (You select the layer on which the trace is to be drawn from the selection box located on the left side of the top toolbar.)

You can add **vias**, which are holes used to connect traces from one side of the board to the other, with the “add via” tool. Red lines are metal traces on the top (component side) of the board, blue traces are on the bottom (solder side) of the board. Green areas represent **pads**. We are using only **through-hole** parts (as opposed to **surface mount** parts) so all our pads have holes drilled in them. Things in white don't “print”; they're just to help you see where the physical boundaries of the parts (and the board) are.

#### Tips and Tricks for Routing

- Use the **ripup** tool to undo routes; if you really want to start over select everything and type “ripup;” in the command box at the top of the screen.
- Most conveniently, you can also let the **Autorouter** do the routing.
- Use the “text tool” to add labels to the board. These labels, which will print “in metal”, can serve to provide helpful information (telling you what plugs in where, for example) and to personalize your design (e.g. sign your work!).