Circuit Design with EAGLE

Version 1.0.1
19 April 2016

Written by Kristof Aldenderfer

Table of Contents

Introduction to EDA

Firmly situated in the Information Age, you are surrounded by technology. Whether it be your cell phone in your pocket or the thermostat in your house, electronic devices will almost universally contain a **printed circuit board (PCB)** of some kind. Some are fairly simple (e.g. solar-powered lights in a garden), and some very complex (e.g. the motherboard of your computer). But they all allow for the interconnection of individual components.

How are these boards designed? Typically, an electrical engineer will model them in a piece of software called an **Electronic Design Automation program (EDA)**. And now, with the relative affordability and power of personal computers - coupled with the open-source software movement - hobbyists as well can now utilize these same tools to create their own PCBs projects!

What concepts are we going to cover?

EDAs allow the engineer - budding or otherwise - to do two things:

1. **Schematic capture**: digitally create, modify, and test circuits
2. **PCB layout**: model those circuits for fabrication

We'll be focusing today on both of these, with the goal of having a finished PCB design that you can then fabricate - either through a 3rd party fabricator, or in-house (if you have an appropriate CNC machine)!

In what context are we going to be covering them?

You'll be learning Cadsoft's **EAGLE**, which is the most ubiquitous EDA on the market. It has a few interesting quirks, which you'll see soon enough. But due to its widespread adoption (as well as the availability of a free educational version), it's the best choice for entering the EDA world.

What should I know before taking this course?

You should have already taken TechShop's two-part Electrical Engineering path: *Circuit Bsaics: DC* and *Circuit Basics: AC*; or know the fundamentals of circuit analysis.

Where can I go from here?

TechShop's Electronics and Electrical Engineering (EEE) curriculum teaches the fundamentals of modern low-power electronics. You are taking an intermediate Electrical Engineering course. *PCB Milling with the Othermill* builds off of this course and actually allows you to mill the PCB that you are designing today.

Let's get started!

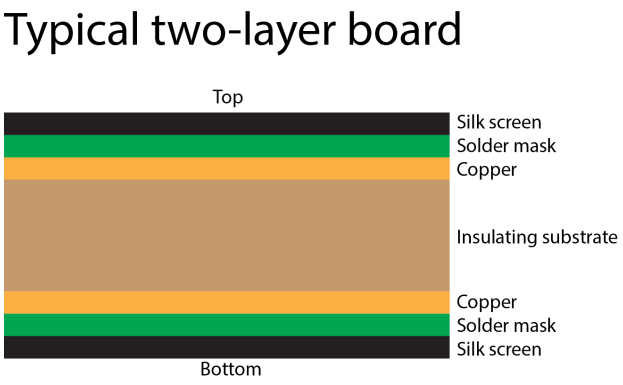
Part 0: What are we doing, anyway?

At a certain point in your electronics endeavours, you will realize that as incredible as the resources are for hobbyists to build prototypes, there is only so far you can go; you will need to design and fabricate your own electrical systems built on a PCB. EDAs allow you to design your own PCB, from prebuilt systems or from the ground up.

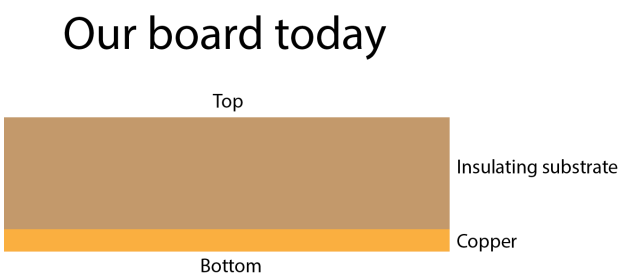
PCB Basics

PCBs come in uncountably different shapes and designs, but they are all built around the same principle: **layers**.

- At the core of the PCB is the **insulating layer** or **insulating substrate**, usually made of fiberglass (FR4) or paper pulp (FR1) and resin.
- On one or both sides of this core layer is an incredibly thin **copper conducting layer**, which is designed to transmit signals from component to component
- The board will typically also have a **solder mask**, which is a lacquer-like polymer coat used to protect copper connections from shorting to each other when solder is added
- The final layer is the **silkscreen**, which contains useful information about components and connections



Fabricating a PCB is a **subtractive process**; the copper is layer is removed to create the shapes of the connections. One of the most important considerations when designing a PCB is the number of copper layers to use, because this will determine the complexity (and ultimately the price) of the board. For this course, you'll be designing a **1-layer PCB**, on which the copper layer will be the bottom, and the components themselves will sit on the top - pushed through the board using **through-holes**.



Choosing the project circuit

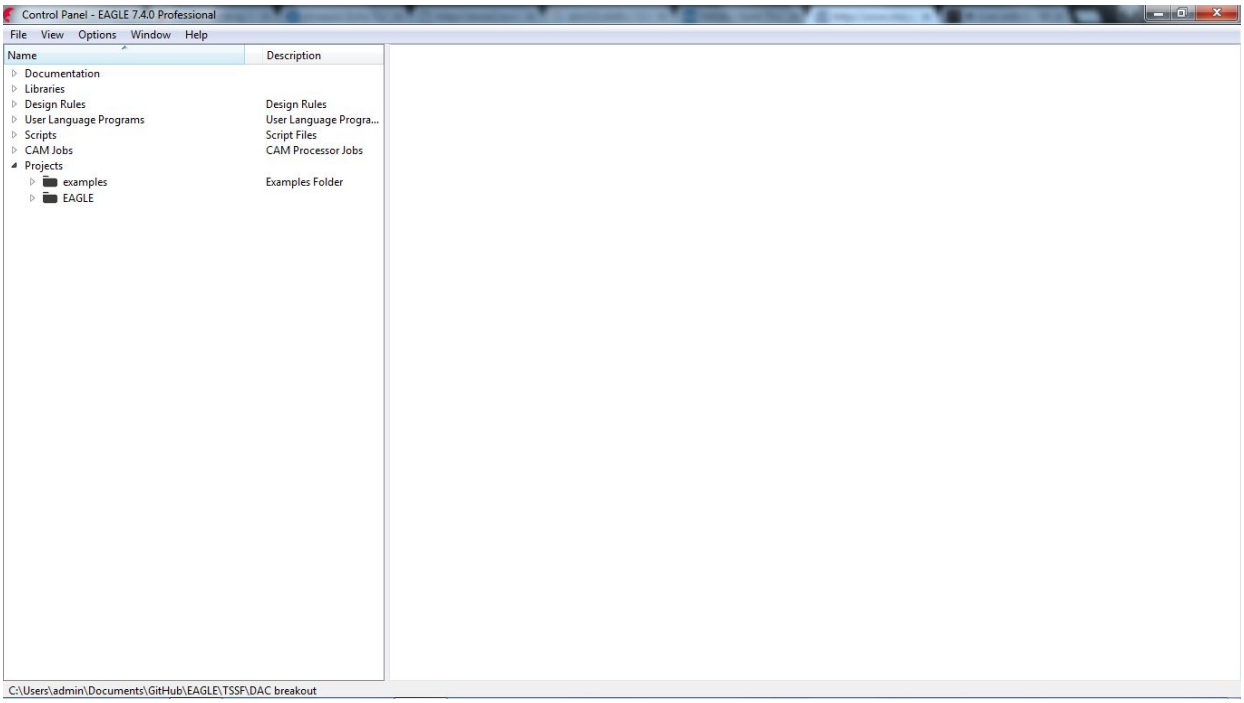
For your first EDA-designed circuit, it is vital that the scope is kept concise and understandable, while maintaining relatability. For these reasons, the circuit that you will be designing is a simple **digital-to-analog (DAC)** breakout board.

You will have had experience at this point with Arduino: a wonderful and accessible microcontroller environment which allows you to rapidly design and prototype systems that combine software and hardware. One of the finer points you may have stumbled across by now is that while these boards have plenty of Analog Inputs, *almost no Atmel board has a true Analog Output*. "Analog outputs" on Arduino are faked using a technique called **Pulse Width Modulation (PWM)**. While this is a fantastic technique for emulating analog signals on digital devices, at some point you'll likely need a true analog signal (such as a function generator or an audio signal); Arduino boards cannot produce this type of signal.

There are many digital to analog converters on the market; they're fairly inexpensive and easy to set up. You'll be creating a DAC breakout today using the MCP4822 from Microchip. (Note: the DAC you're using communicates via SPI. If you're unfamiliar with the SPI communications protocol, please ask your instructor.)

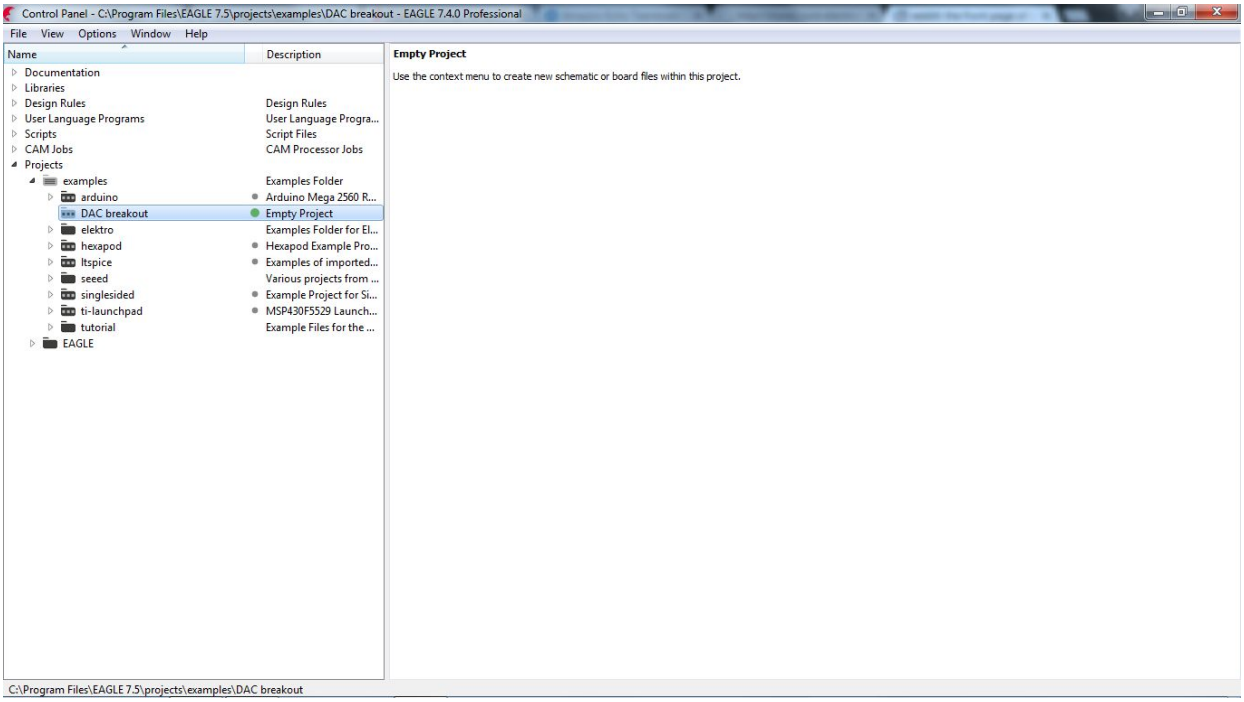
Part 1: Setup

Opening up EAGLE, you'll be presented with the window below. This is the **Project Window**, and it is effectively a browser for all things EAGLE; projects, libraries, design rules, et al.



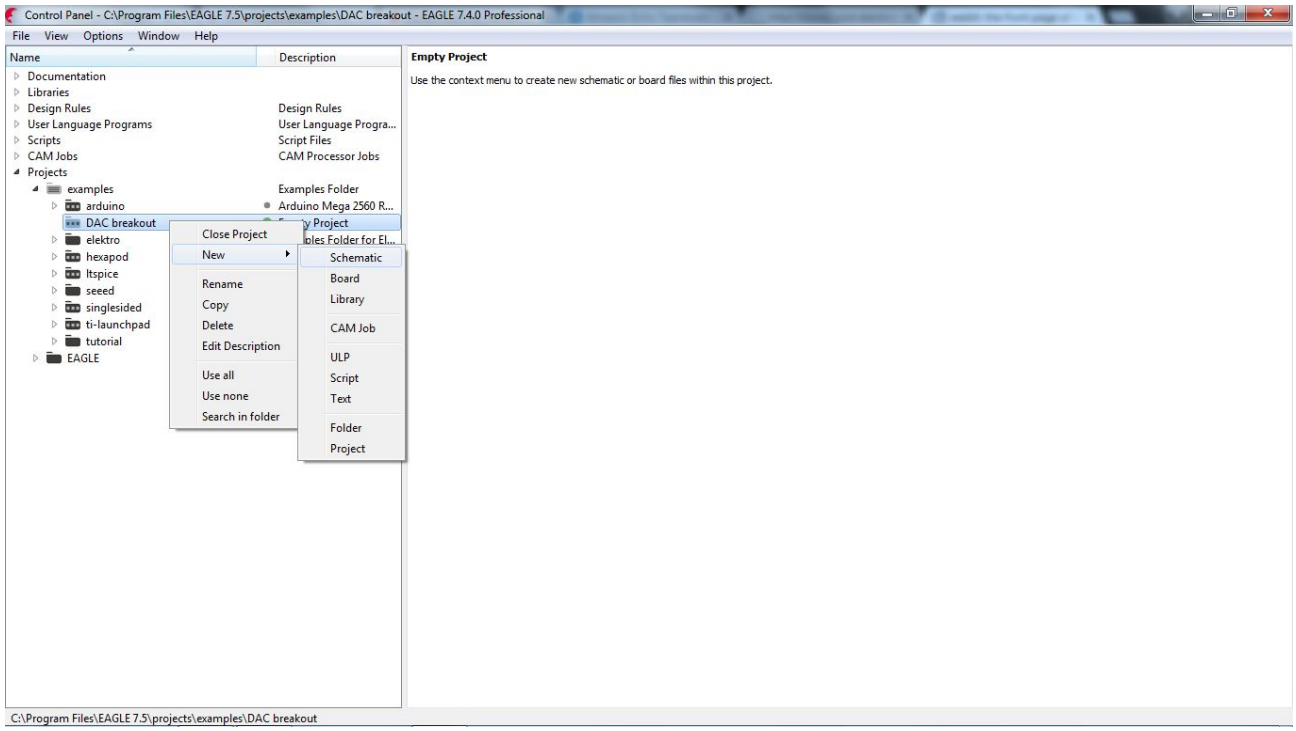
Create a new project

The first thing you'll want to do is create a new project. You can do this by right-clicking on one of the folders inside "Projects" and selecting "New Project". Rename it so that it is called "DAC breakout":

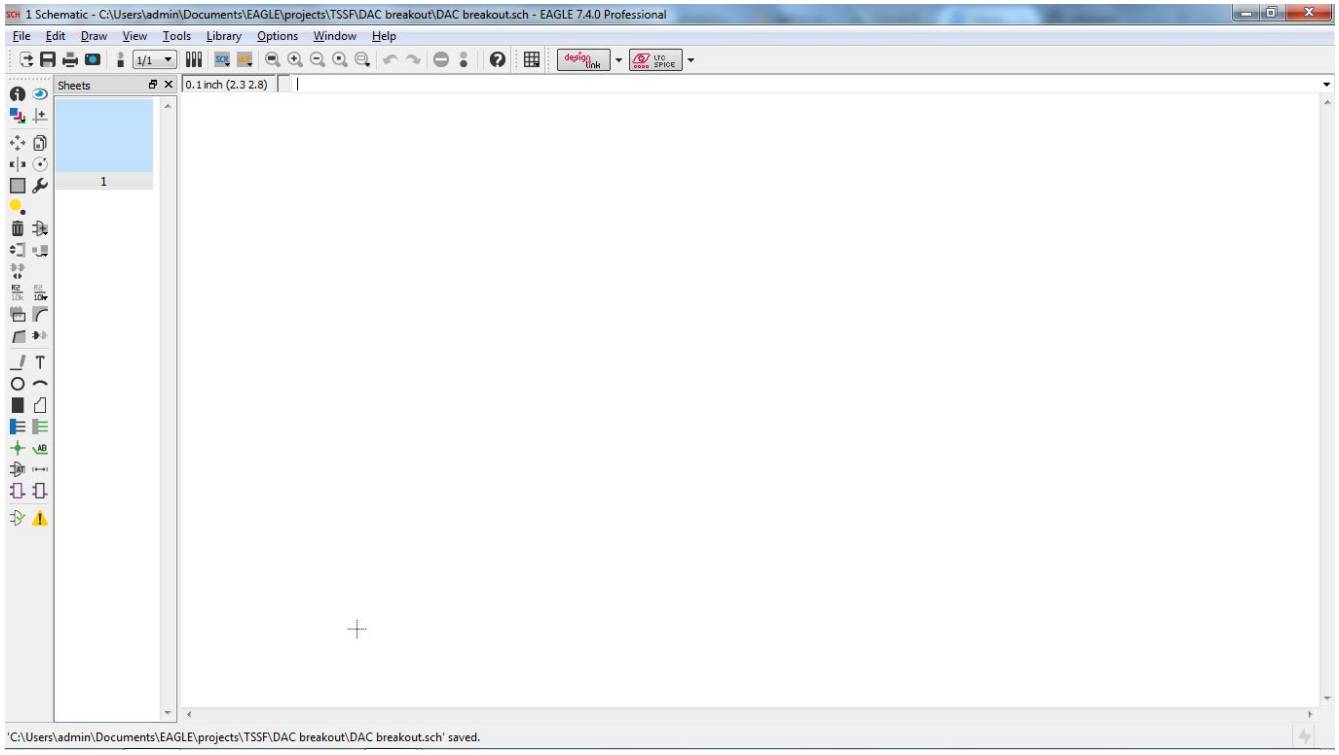


The circle to the right of the project has turned green - his means is is active! Only one project can be active at a time.

You'll want to add a new schematic to the project. Right-click on the project folder, and select *New → Schematic*.



The window below should appear. Save this so that it's got a real name! Name it "DAC breakout.sch".

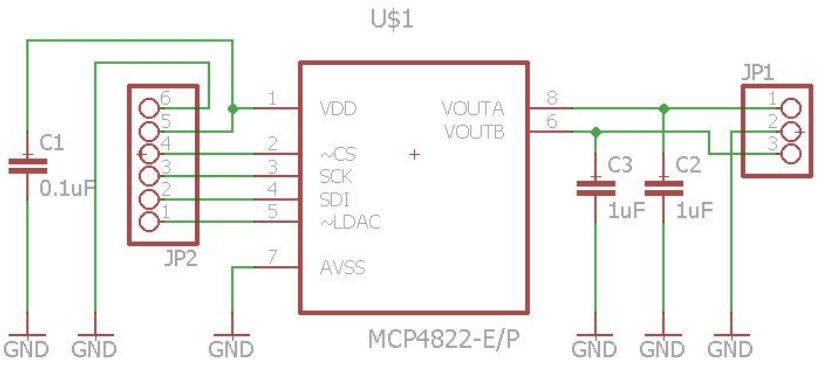


Congrats; you've created a blank schematic! Let's get to designing.

Part 2: Schematic capture

You have now entered the realm of **Schematic Capture**. This is where you will design your circuit! What you'll be doing, in essence, is defining the components and the electrical connections among them.

For reference, this is the circuit you'll be making:



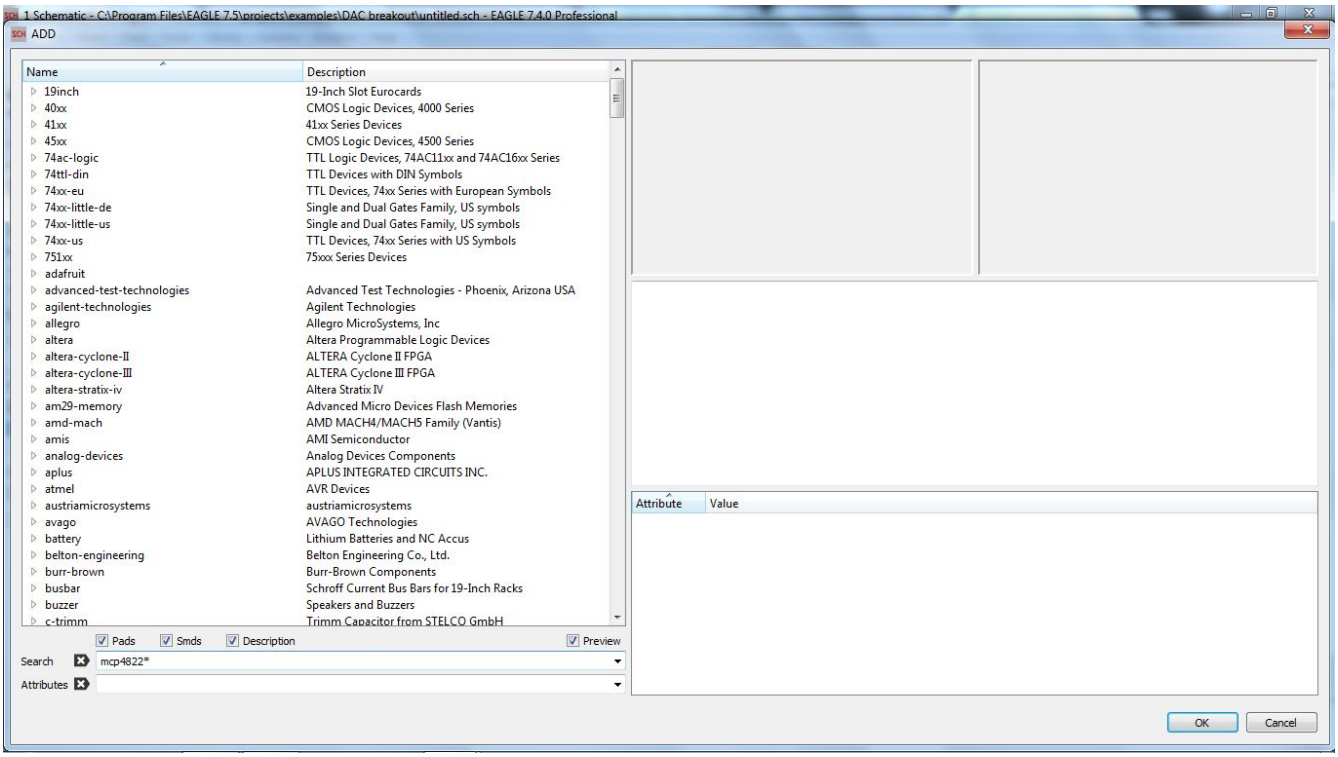
You're going to spend quite a bit of time in the Schematic Capture window; let's quickly take a look at the **tool palette**, which is on the far left side of the window (and the far left of this page). Because EAGLE was built on top of command-line scripts, its toolset functionality behaves slightly differently to most other **User Interfaces (UIs)**: each type of action requires a different tool. So for example, to delete something you cannot use the delete key - you must select the **Delete tool**.

You're going to design the schematic in four steps:

1. Add the components
2. Connect the components
3. Define names and values of components
4. Tidy up

Add the components

There are many many tools here, and you most assuredly will not be using every single one of them today. The first (and arguably the most important) tool you're going to use is called the **Add Component tool**. Click on it, and a new window should pop up that looks like the one below:

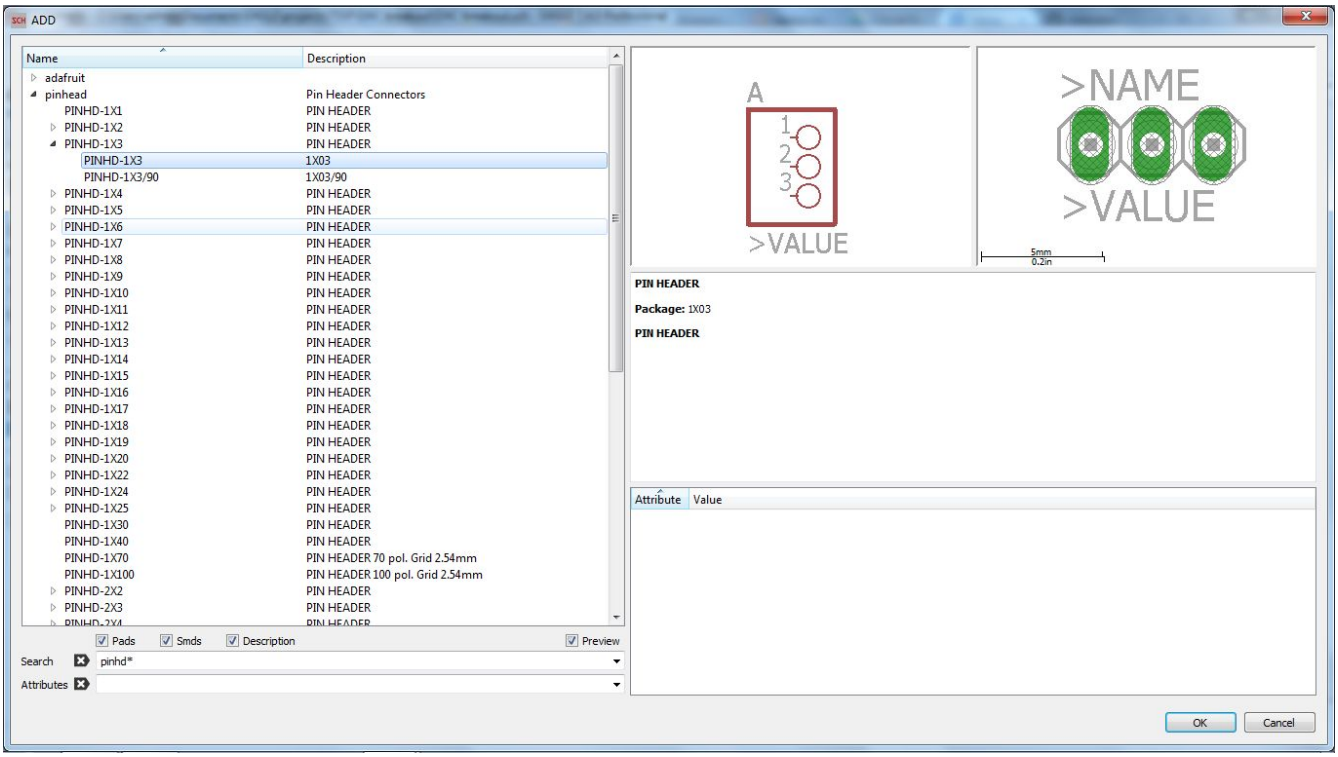


Selecting the right component is a little more nuanced than you might expect, because components will usually have associated with them both a schematic symbol and a footprint, and you need to know which you'll be using. The **schematic symbol** is only a visual representation of the components, and allows you to define the electrical connections. The **footprint** represents what the physical copper connections on the board will look like, and so selection of this aspect is very important; you'll need to know what the **package** of each component looks like in order to select these correctly. (For reference, take a look at the datasheet for the MPC4822, pages 36, 38, and 39 - it comes in *three* different packages.)

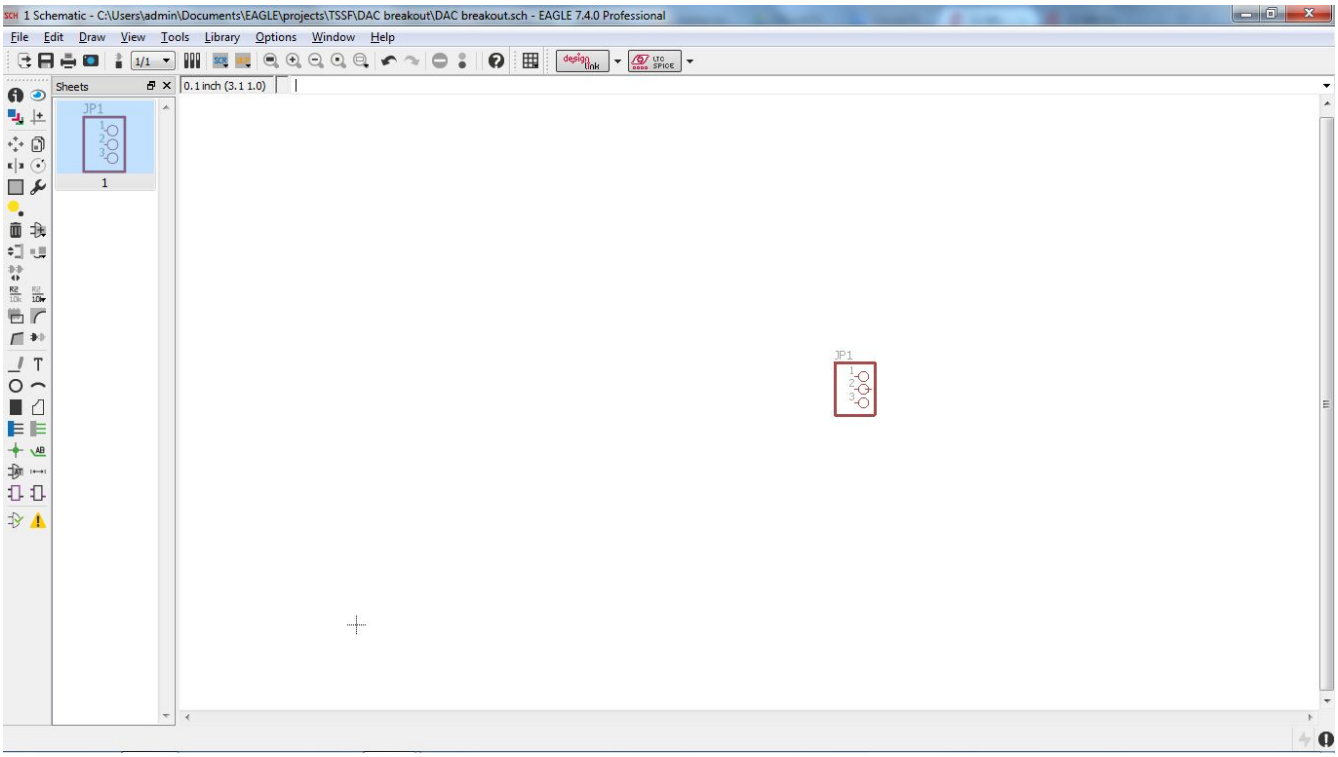
Components are organized into **libraries**; you'll be using several common ones today.

Add the headers

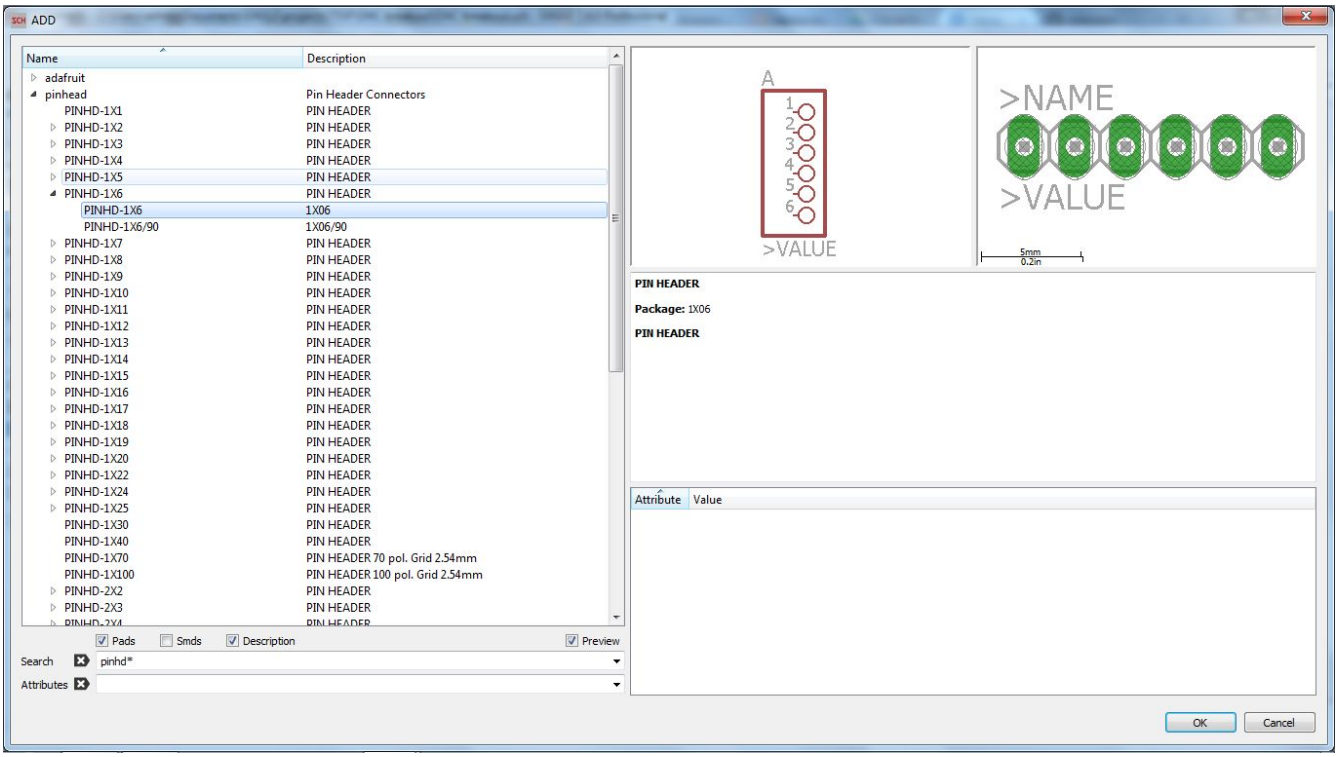
Search **pinhd***, and from the “pinhead” library, select **pinhd-2**. Select the first package; it’s not terribly important for these two components, but it will be for the rest of them.



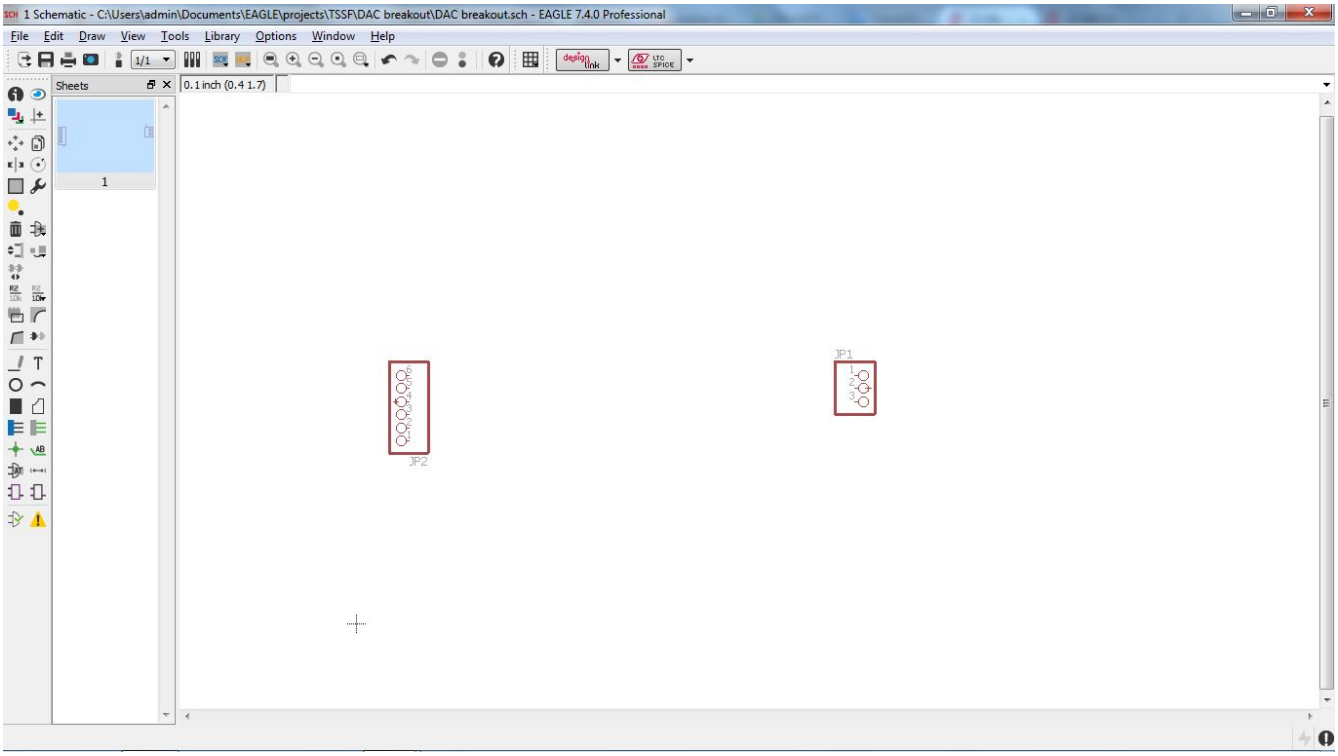
Add it to your schematic by clicking anywhere:



Then hit escape, and you'll be magically transported back to the Add Component window. Next, from the same library, add a **pinhd-6**:

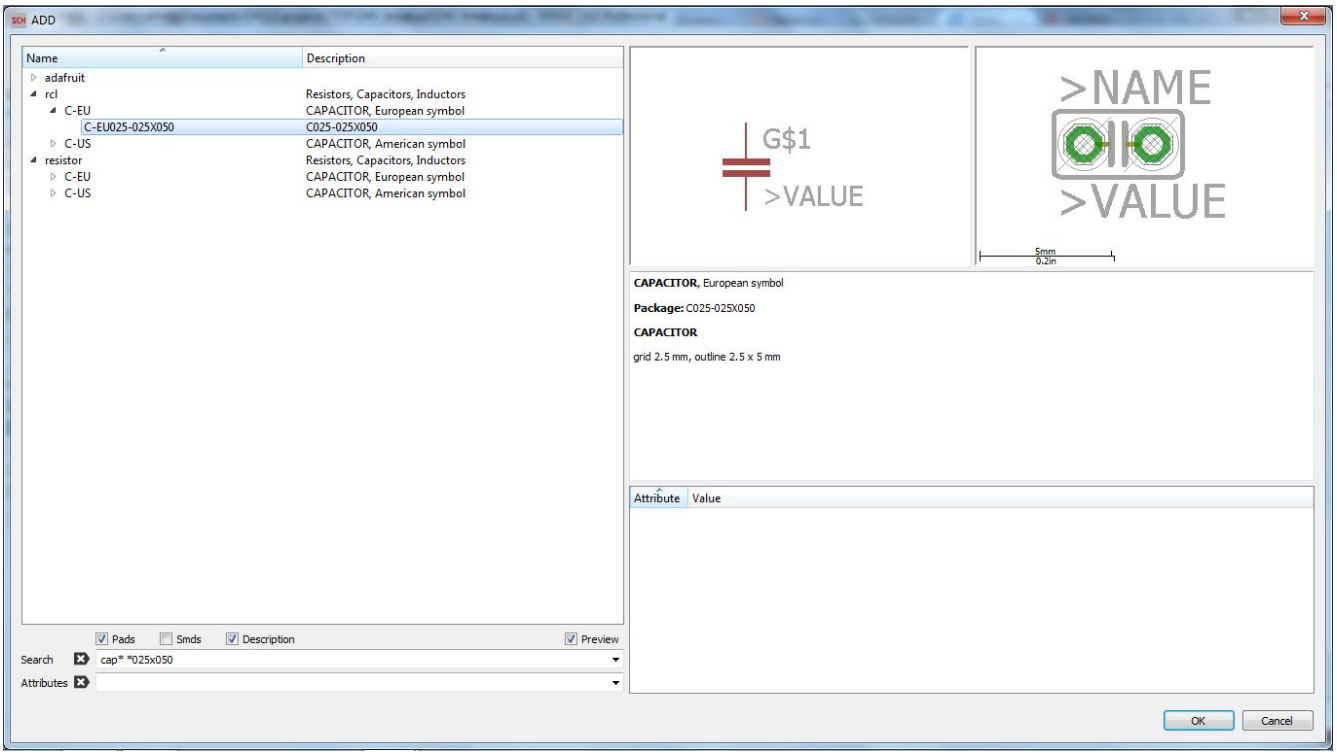


You can right-click to rotate the component before you place it:

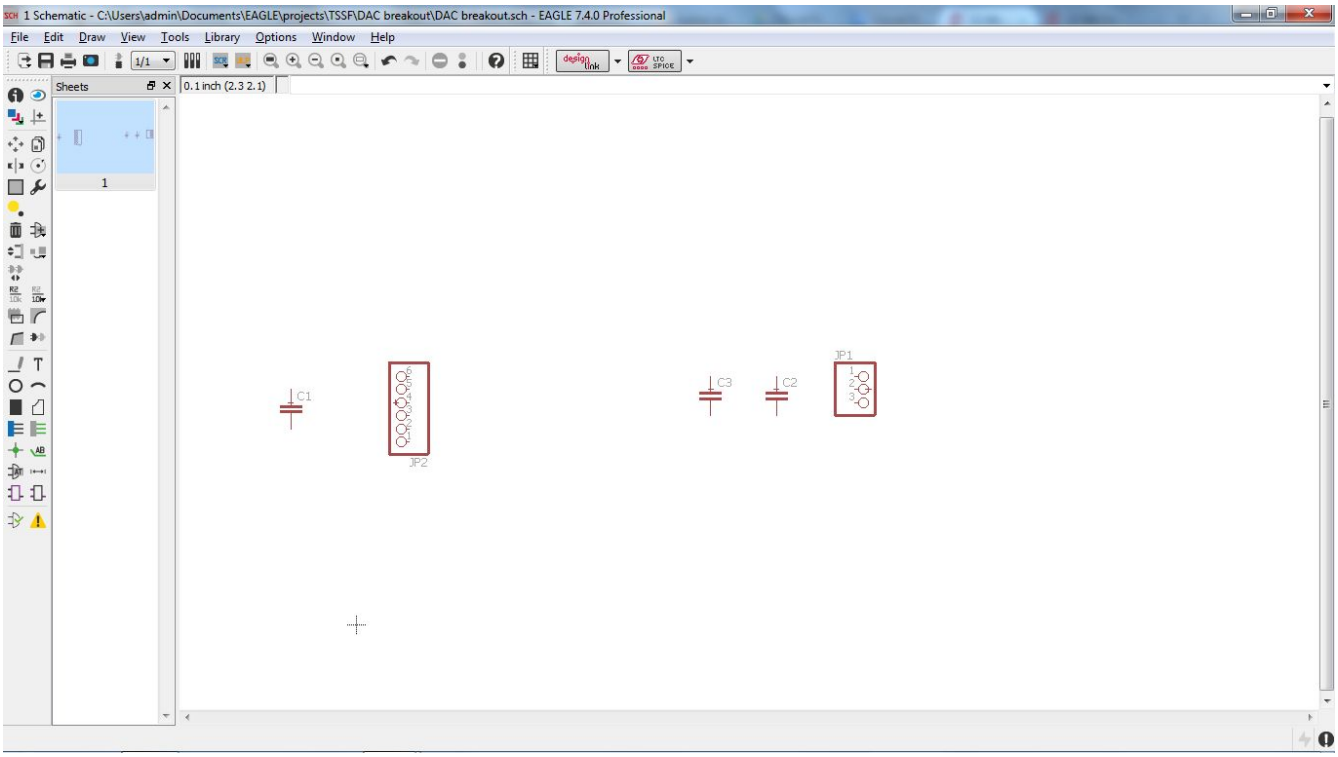


Add the capacitors

Here is where the footprint becomes important. You're going to want to use the package which is 2.5mm by 5mm. Search **cap*** ***025x050**, and from the **rcl library**, select and place three caps:

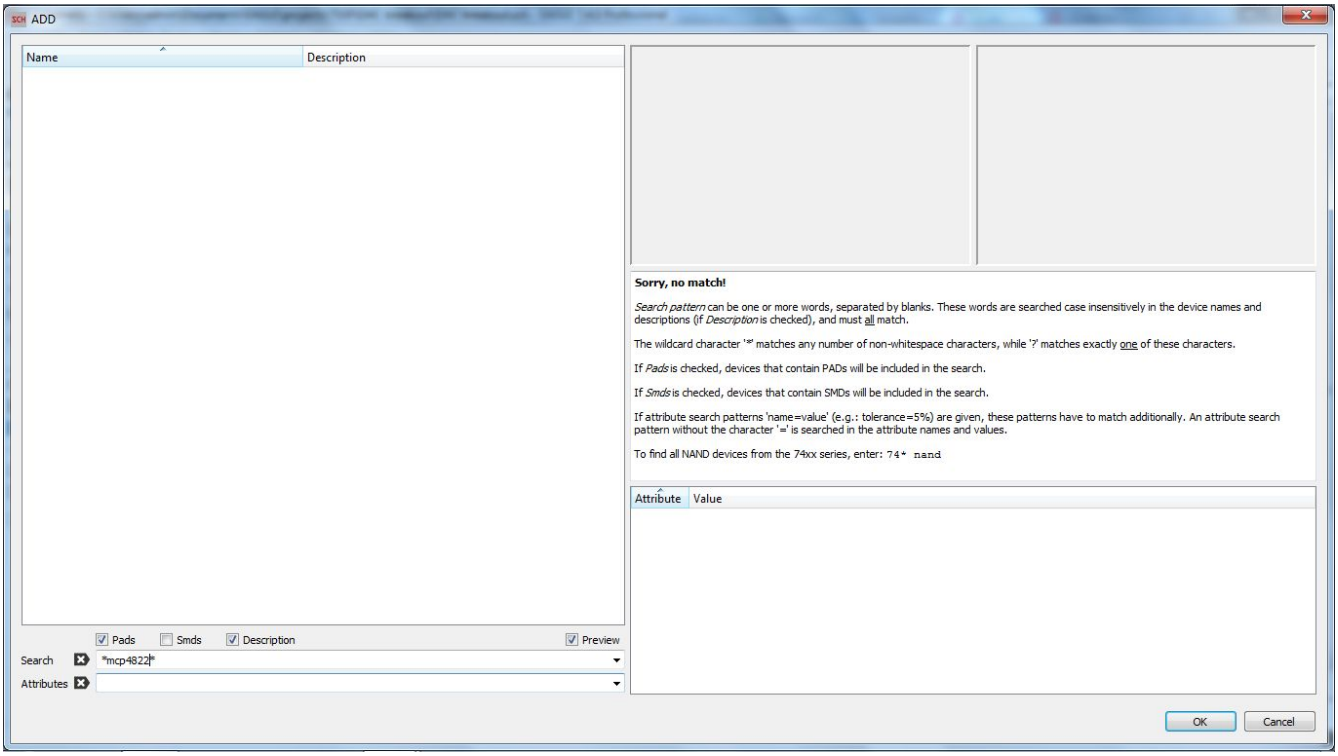


Your schematic so far will look like this (notice that you cannot see the footprint here; it will show up on the PCB layout):



Add the DAC

It's time to add the coup de gras: the MCP4822 DAC. Search for ***mcp4822***.



Wait a minute; it's not here! What's going on? EAGLE by default comes with a slew of useful libraries, but the DAC you're using isn't in any of them. You'll have to manually add another library to access its schematic symbol and footprint. This is done in 2 steps:

1. Move the new library into EAGLE's */lbr* folder.
2. Indicate in the Project Window that you want to use this library.

Move the library into EAGLE's */lbr* folder

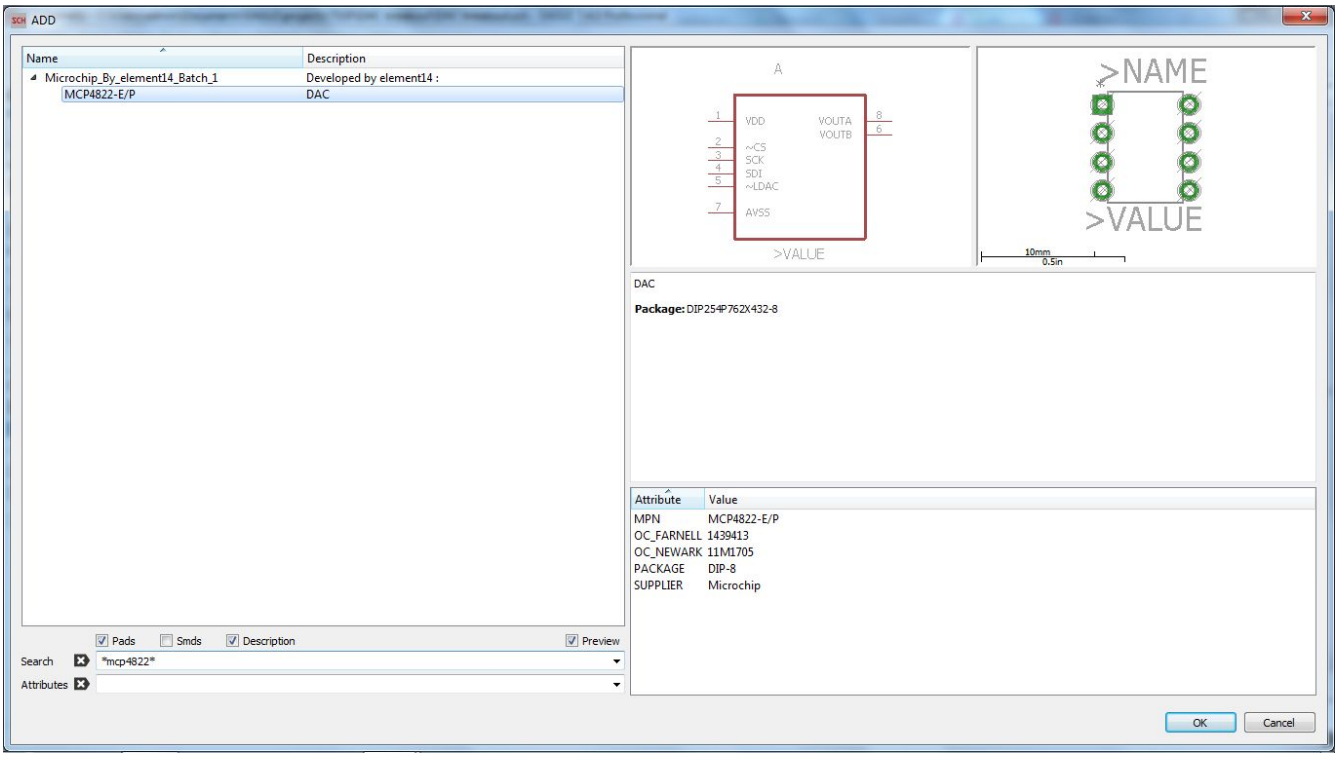
Within this course's repo, navigate to */support/lbr*. It should contain a folder named "Microchip". Grab that folder and move it into EAGLE's default libraries location:

C:\EAGLE [version number]\lbr

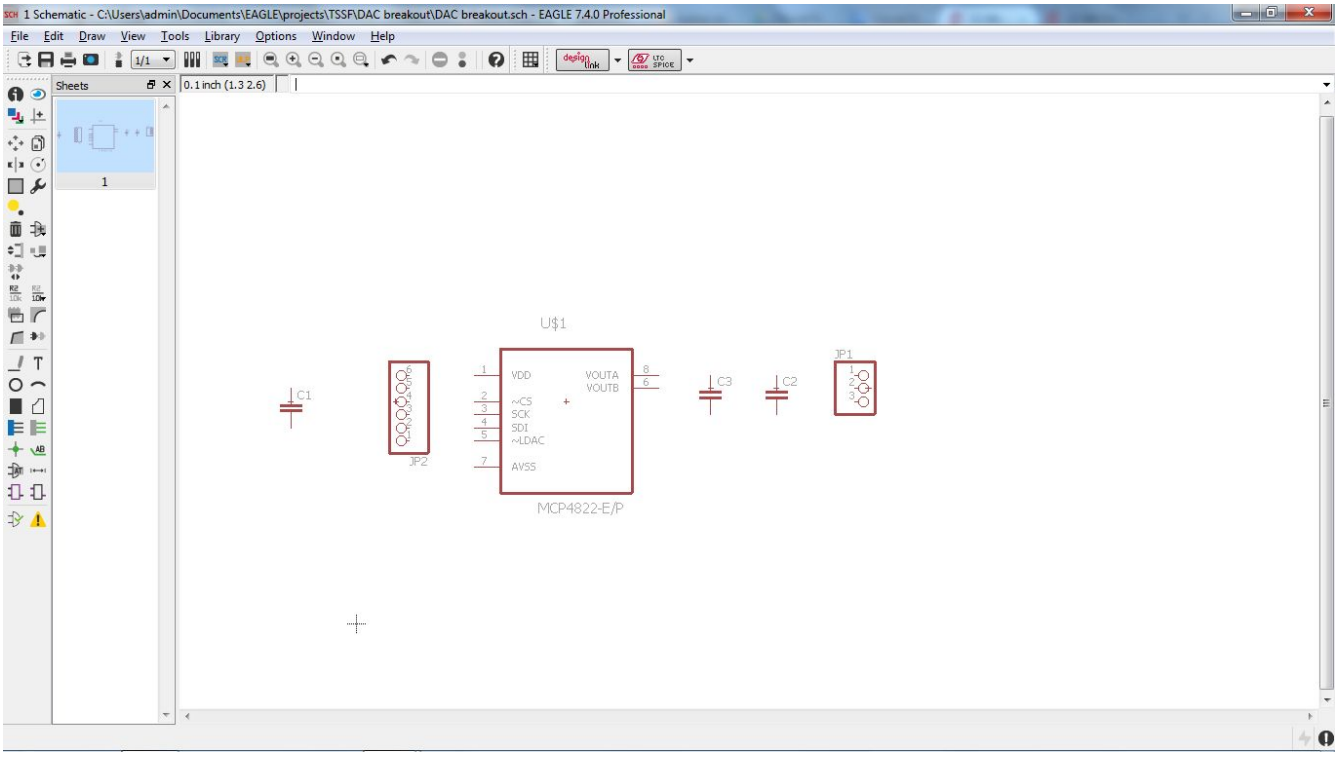
Indicate in the Project Window that you want to use this library, then add the DAC

Switch over to the Project Window, and on the top left panel, expand "Libraries". We want to mark the Microchip library as active; just like projects, active libraries will have a green dot beside them - but you can have many active libraries at a time. Find the Microchip library, and click the gray dot beside it to turn it green. You're good to go!

Now swap back to the Schematic Capture window, click “Add Component”, and search again for the MCP4822:

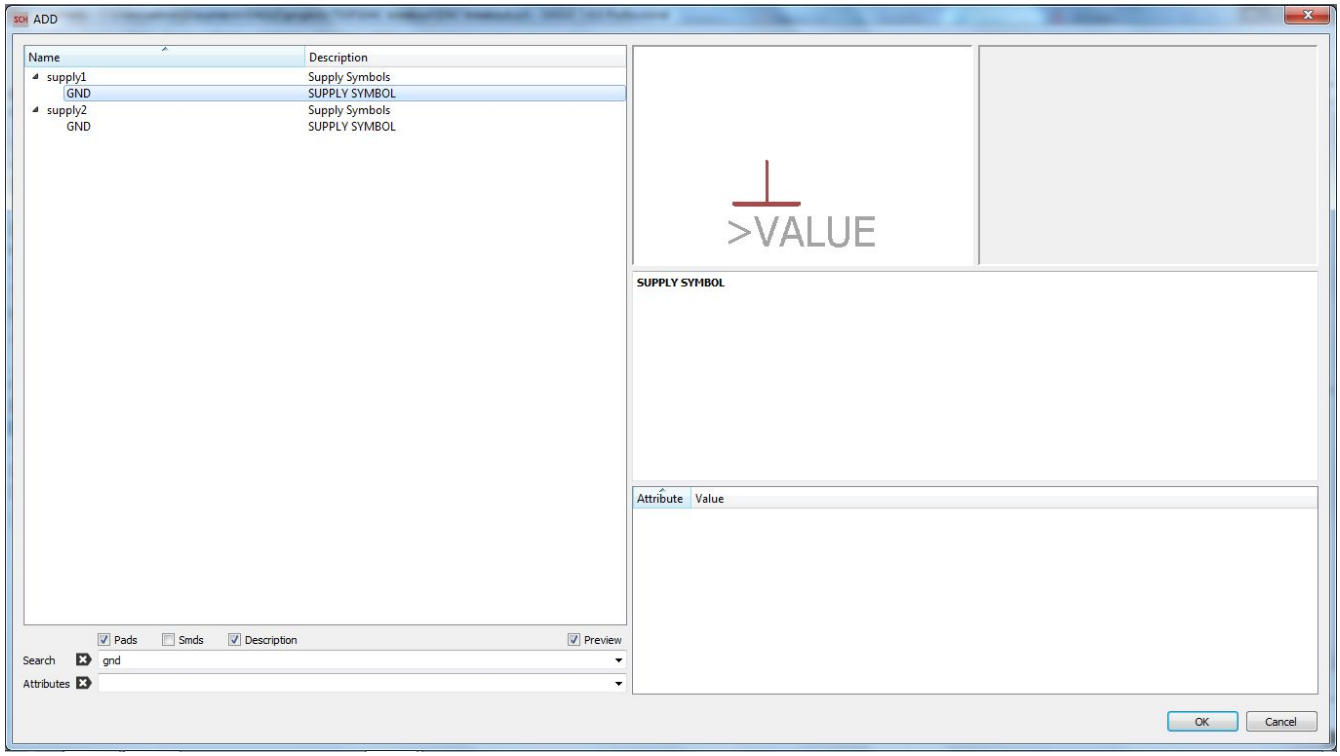


Add it to your schematic:

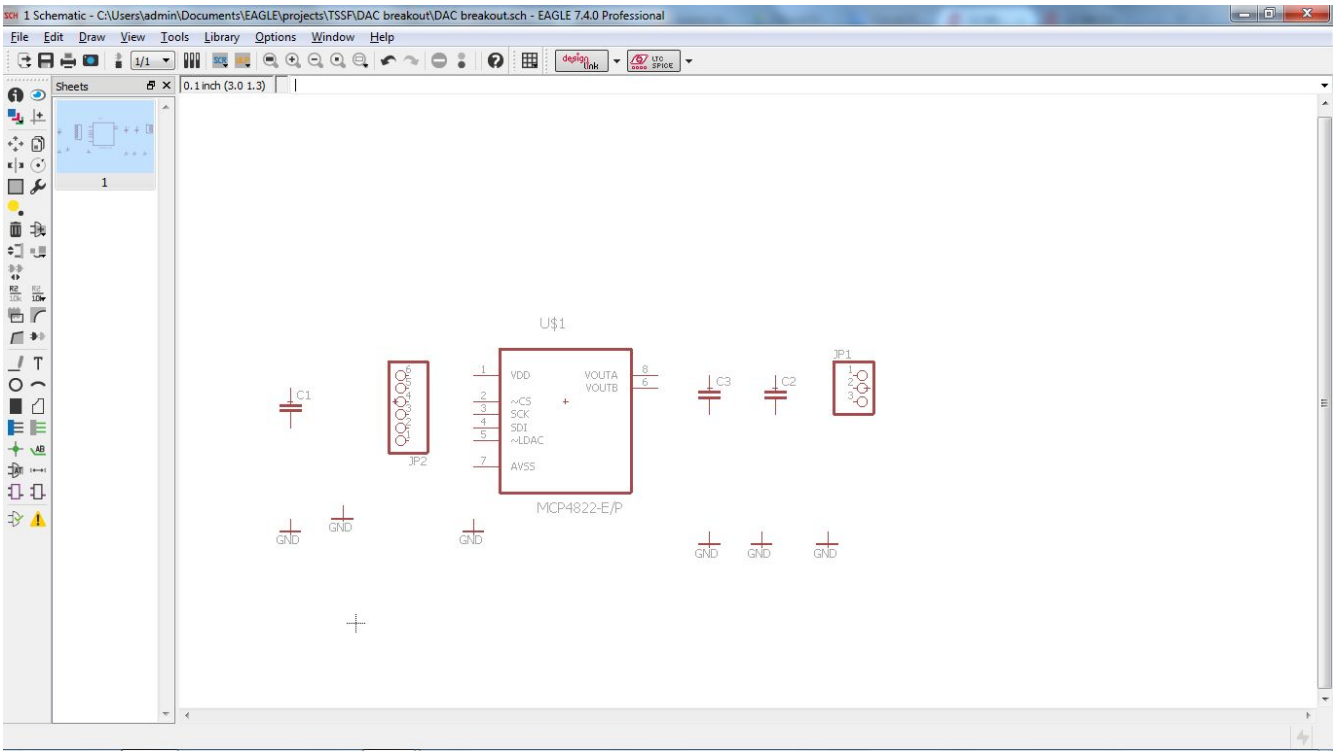


Add ground

The last “component” you’ll need to add to your board is ground, by searching **gnd**:



Add six of them. They all refer to the same thing: a voltage reference point.



Congrats; you’ve successfully added all the components to your schematic!

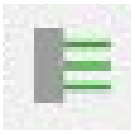
Connect the components

If the position of the components isn't optimal, use the **Move tool** to position them to your liking.

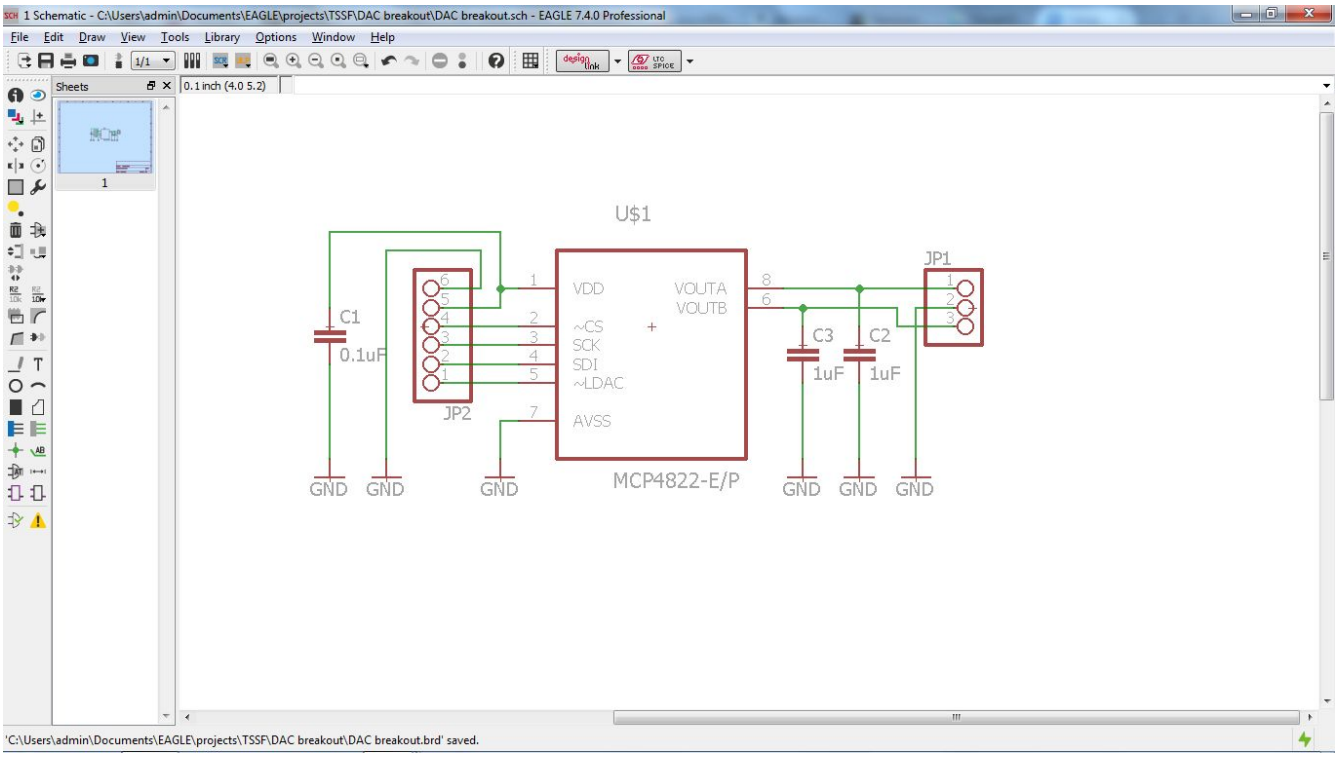
If components need to be rotated, use the **Rotate tool** (or right-click while moving using the Move tool).

If components need to be mirrored (as the 6-pin header strip will), you can use the **Mirror tool**.

Then use the **Net tool** to connect the pins!

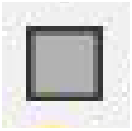


You'll now begin to draw green lines from pin to pin; these are called **nets**, and they truly define the electrical connections between components. Your schematic should ultimately look like the following:



Note: if you want to perform an action to more than one object at once, you'll need to first use the **Group tool**. For example, if you want to move three close components together:

- Choose the Group tool
- Drag a box around those components
- Select the Move tool
- Right click on the group and choose "Move: Group"



Define names and values of components

If you're going to fabricate and assemble this circuit, you'll need to give values to all the components. Some of the components already have values (e.g. MCP4822); some don't need values (e.g. header strip). But there are several which do. They are:

- C1: 0.1uF
- C2: 1uF
- C3: 1uF

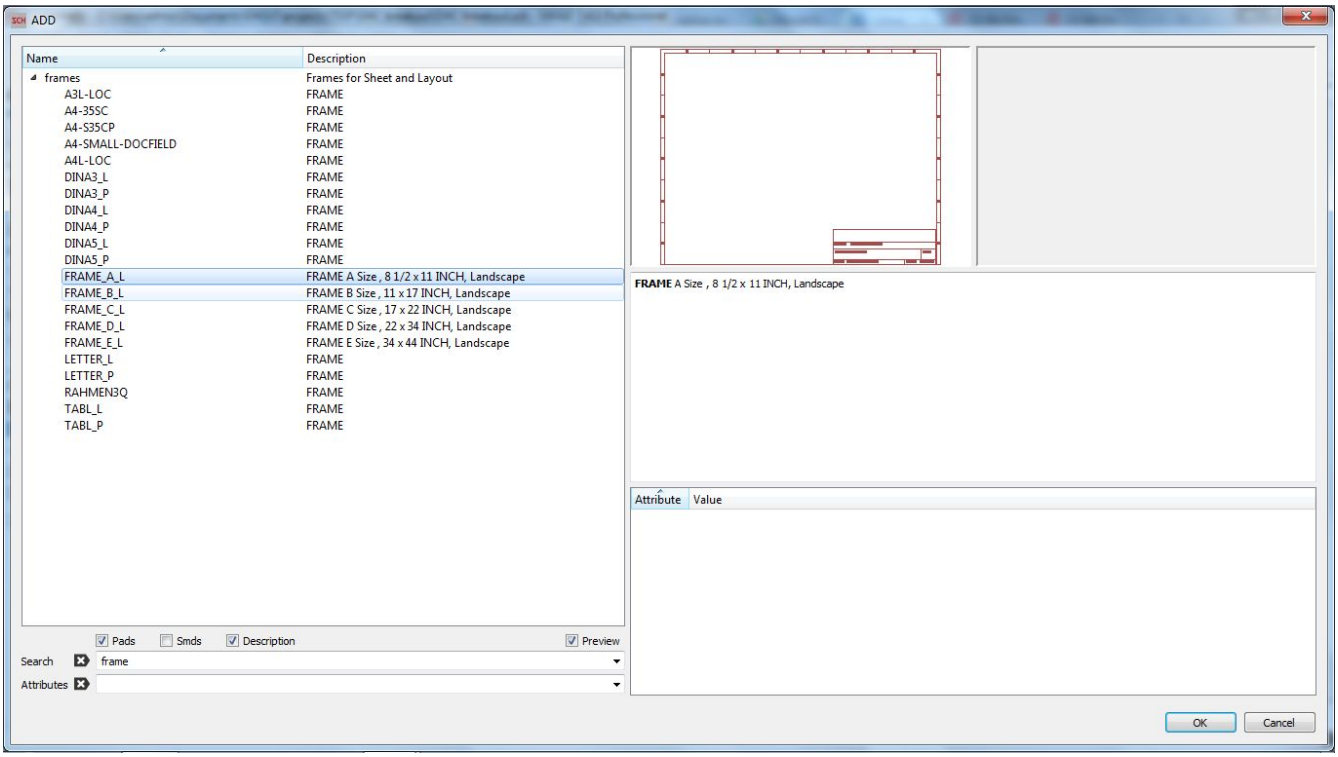
Use the **Value tool** to change their values.

Don't worry if the numbering system for your capacitors is different; as long as the values are correct, the software has no problem lining them up.

You can also use the **Name tool** to label the jumpers, if you'd like. Names have no electrical significance in EAGLE; they only serve to distinguish one component from another.

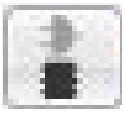
Tidy up

One last step before you design the PCB: in order to make this schematic as understandable as possible, you should clean up any stray connections, minimize overlaps, and generally tidy it up. Additionally, if your schematic ever gets printed out, you'll want a frame around it which will help other engineers identify and understand your design. Add a frame by searching **frame** and selecting the **8.5in x 11in landscape** version:

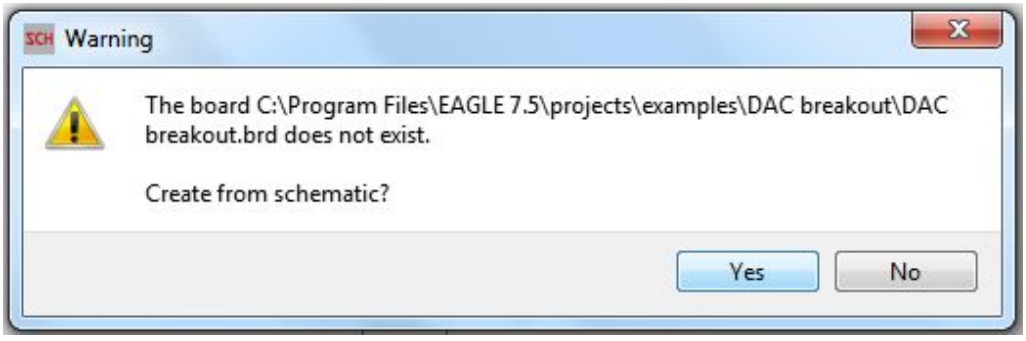


Place the bottom left corner at the same spot as the "+" overlay. Then select every part of your circuit using the Selector tool, switch to the Move tool, right-click on your circuit and select "Group: Move":

Move it all into the center, and then save. The next step is to generate a board from this schematic; click the "Generate/switch to board" button on the top toolbar:



A warning will pop up. This is to be expected: yes you *do* want to create a board rom the schematic!:

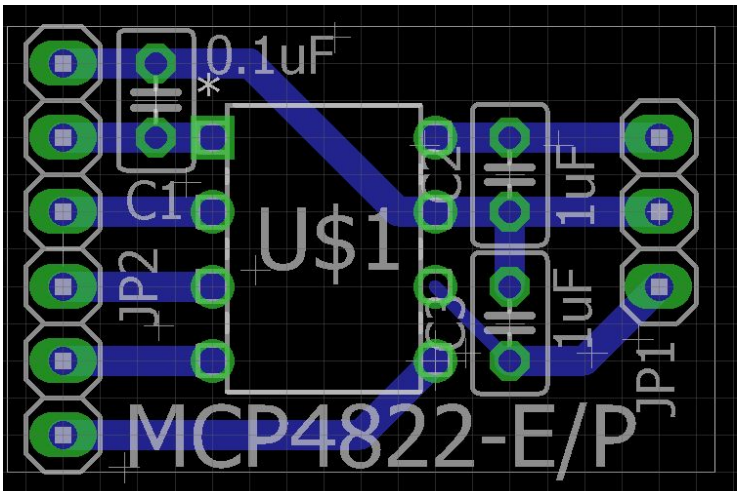


Congrats; you've designed your first circuit schematic using an EDA!

Part 3: Lay out the PCB and test

You have now entered the realm of **PCB layout**. This is where you will create the physical design of your board!

For reference, this is the PCB you'll be making:

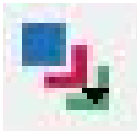


Just as in the Schematic Capture window, you've got on the far left of your screen a **tool palette**. It behaves the same way as the previous tool palette.

You're going to design the PCB in three steps:

1. Position and orient the components
2. Draw the connections
3. Test and modify accordingly

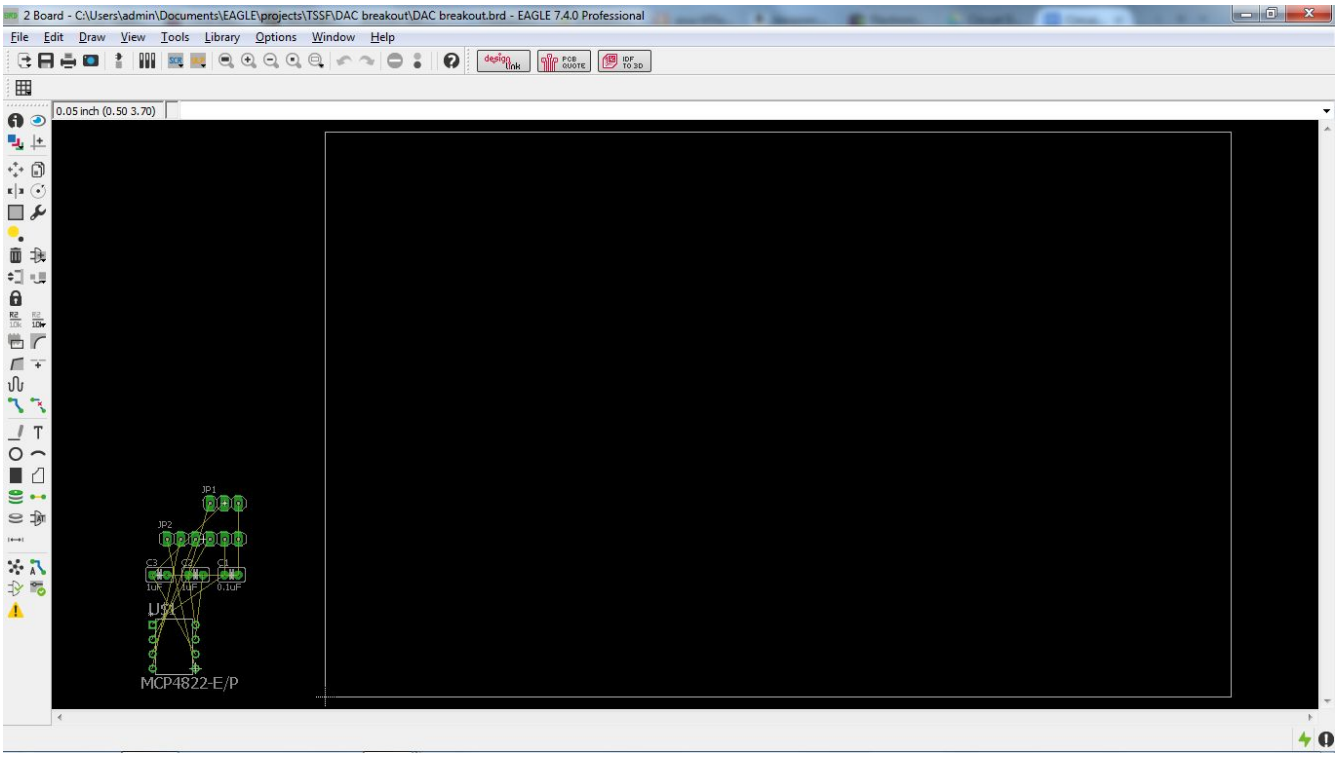
More important in the PCB layout step than anywhere else is the concept of **layers**. As discussed previously, PCBs are made of several layers of material. You're going to utilizing those layers now. There is a **Layer viewer tool**:



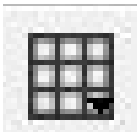
which allows you to turn on and off the visibility of each layer. When you open the layer viewer, you'll see close to 100 different layers; don't panic. The footprints you selected contain correct layer information for each component. All you'll really need to concern yourself with it layer 16: the bottom. You're ultimately going to connect all components on this layer with **traces**.

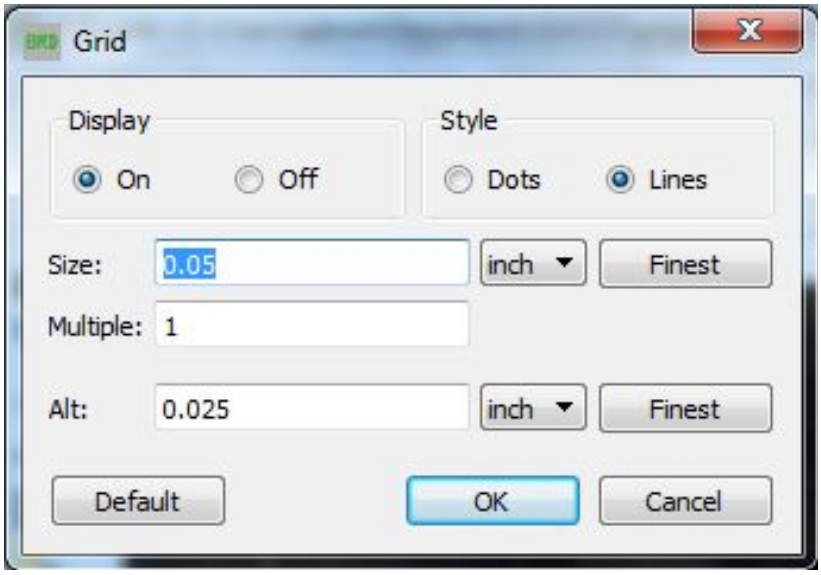
Move the components into place

EAGLE places all components in the bottom left side of the screen by default. You'll see the following window:

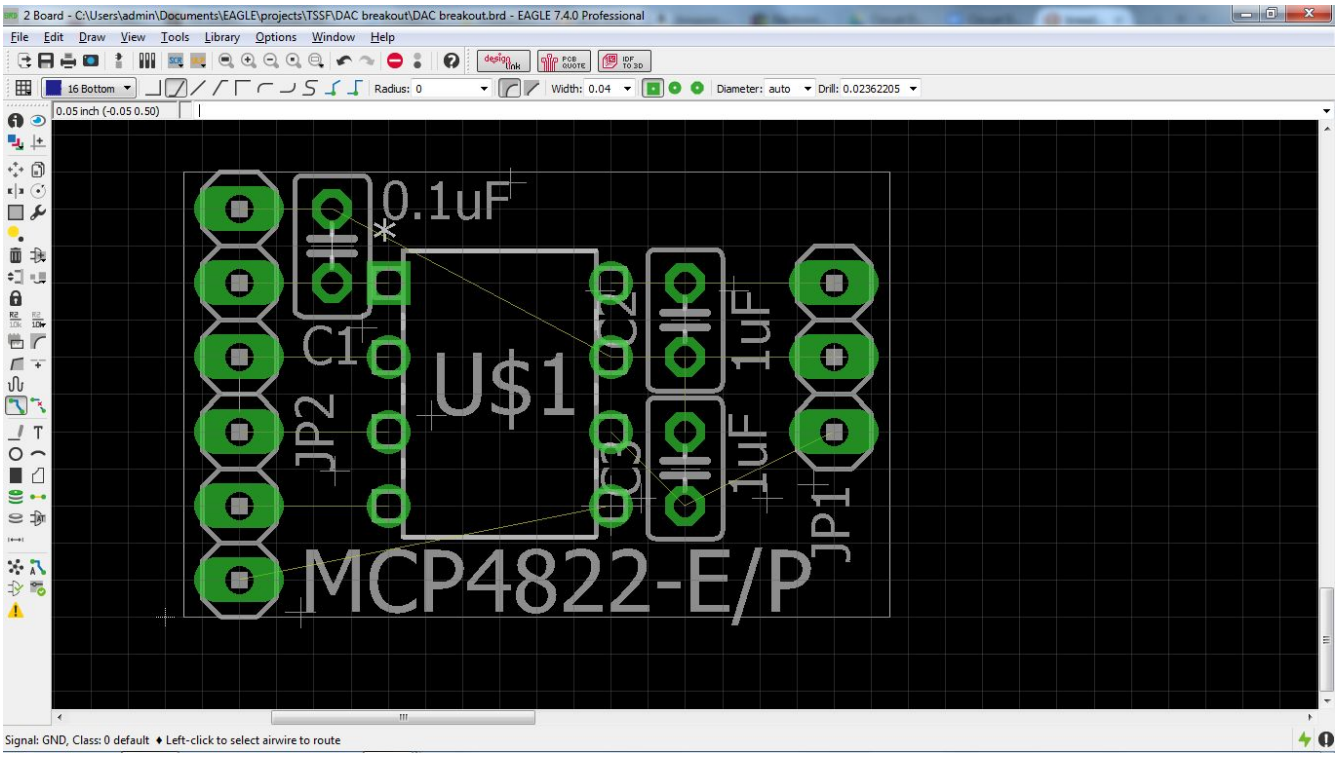


(You'll also notice that these are the footprints you selected earlier!) Select all the components (using the Group tool) and move them into the center of the screen, inside the board outline. Then, in the top left corner of the window, just above the tool palette, press the **Grid options** button. You'll want to be able to see the grid while you design. Set the options as follows:

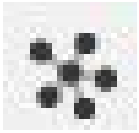




Adjust the components so that they are positioned and oriented as below. Make note of the board outline here (the thin white line around the components): it has been adjusted create a perimeter around the components, while keeping space efficient. This is the outline that will be cut out when you have your board fabricated.



At this point, you'll likely have a rats nest of yellow lines over your board. These are called **airwires**, and they're used to describe the connections you made on the schematic between the components. They don't automatically recalculate; you can manually force EAGLE to, however, by using the **ratsnest tool**!



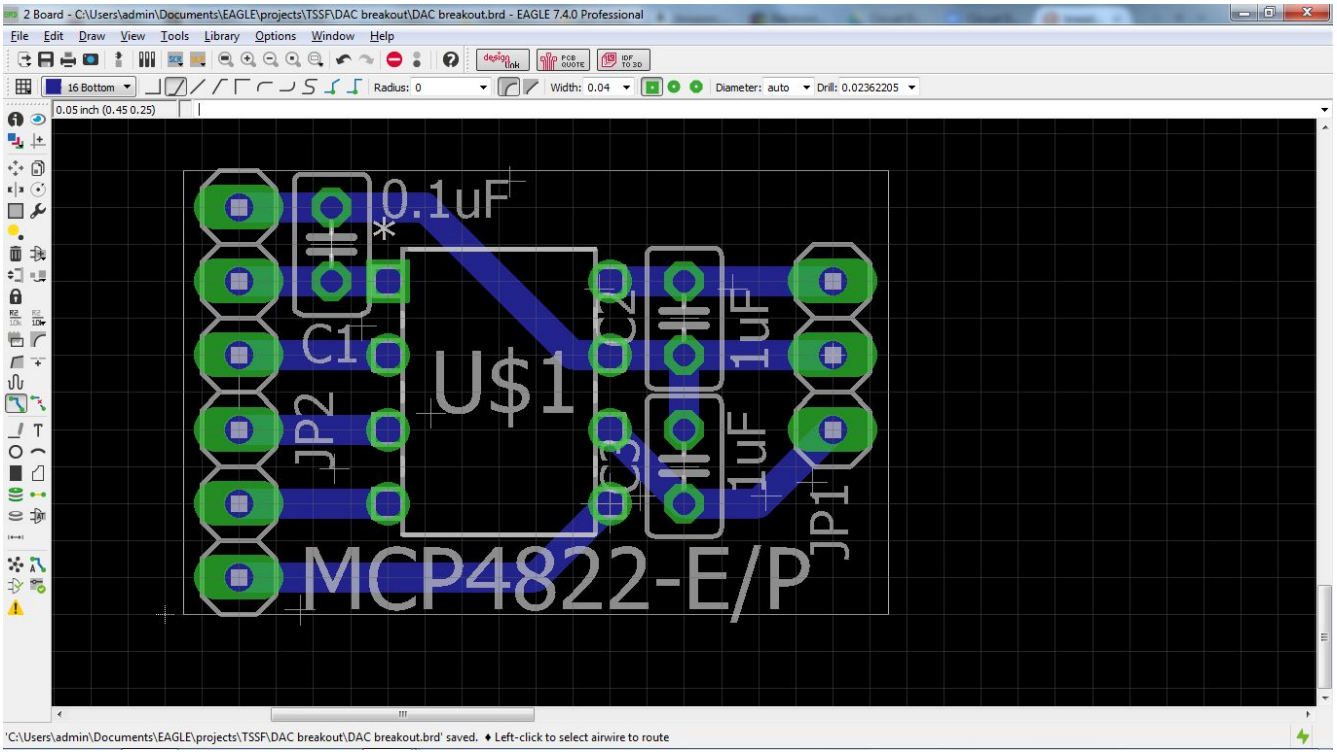
Draw the connections

Now that everything is situated correctly, it's time to manually route all the **traces** on the board. Select the **Trace tool**:

This converts the nets to true physical copper connections: traces. And you can now begin to connect each yellow ratsnest wire to its destination. But before you do, check the toolbar at the top, and match your settings:

You want to be drawing the traces on the **bottom layer**, with a **width** of 0.04. (That's in inches, by the way.)

Once you've routed everything, you should have something similar to below:



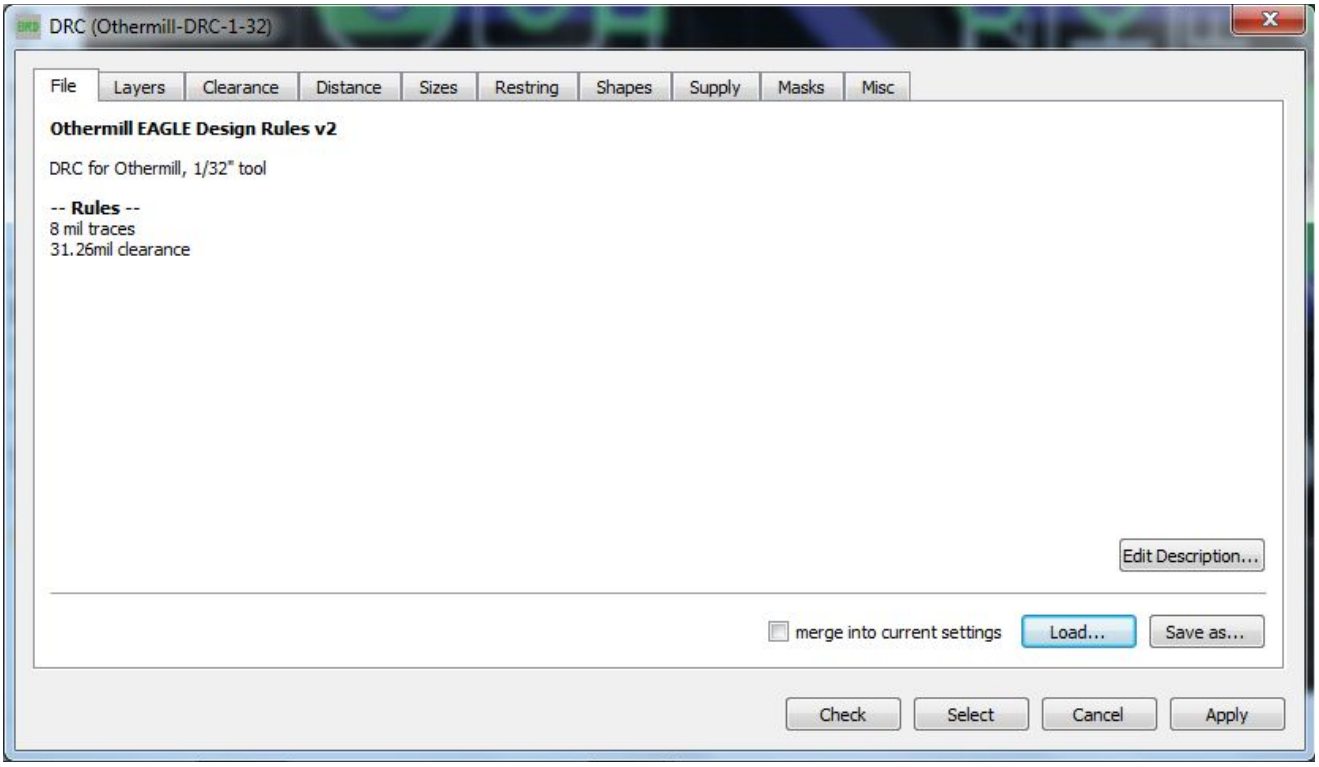
Congratulations; your board has been fully routed! Let's test it to make sure it can actually be fabricated.

Test and modify accordingly

Machines fabricate PCBs. Machines have limitations. You need to make sure your board falls within **design rules** specified by the machine you'll be fabricating the board on. The course's repo contains **design rules check (drc)** files for some common fabricators which you can check against. As this course leads directly into *PCB Milling with the Othermill*, you're going to check against the Othermill 1/32" endmill rules.

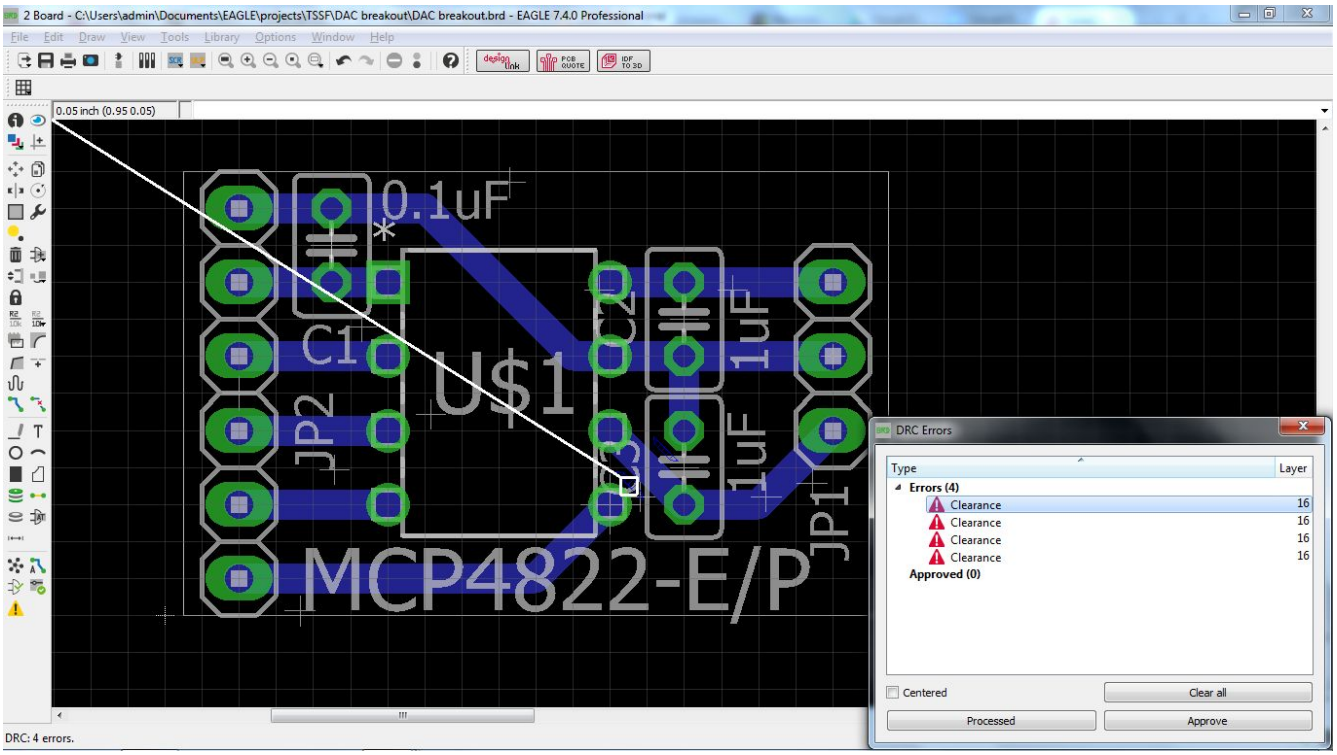
Hit the **DRC tool**:

Hit the "Load" button, then find and select the Othermill 1/32" DRC.

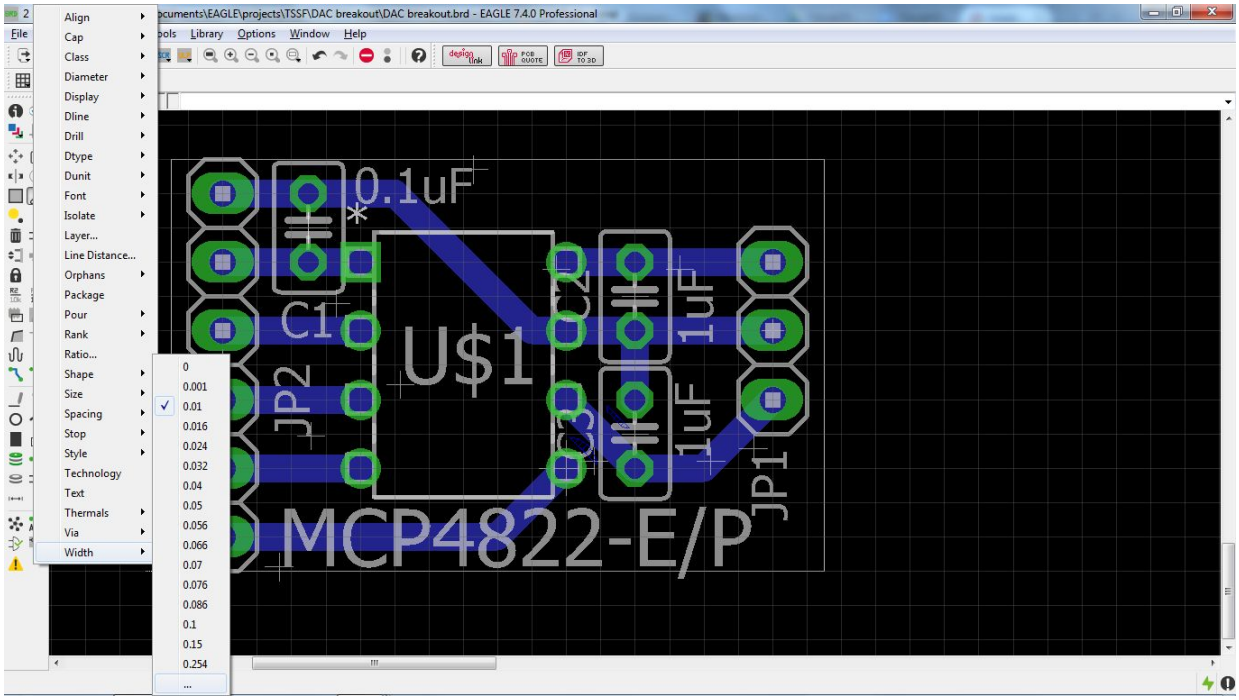


Then hit "Check".

You'll see that, based on the design rules, the board below isn't going to be fabricated properly unless it is modified. These errors are saying that there isn't enough space between the trace and the pads for the tool to fit.



Let's fix this using the **Change tool** to make the trace narrower so that the tool can fit in between it and the copper pads. Select the Change tool, then drop to *Width → 0.01*

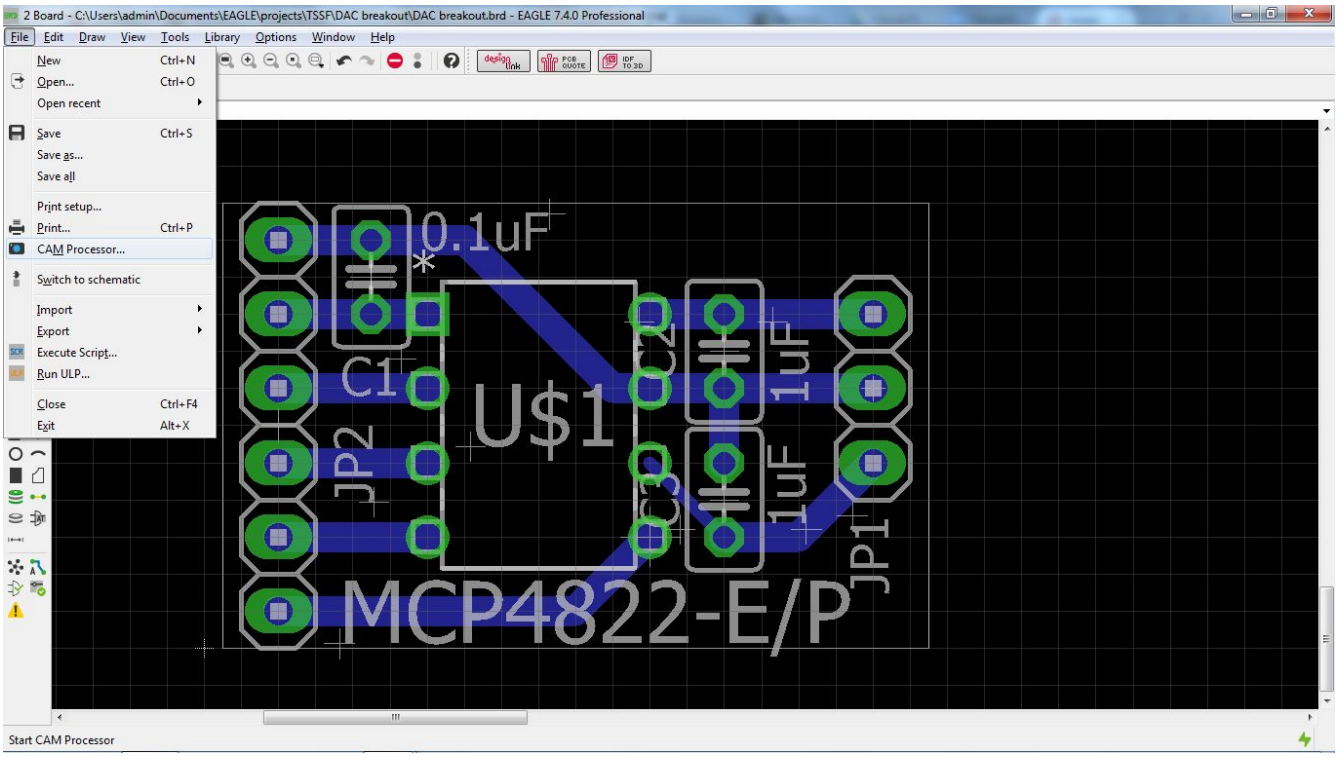


Click on the offending trace to change its width. Now re-run the DRC, and it should generate no errors. Congrats; your PCB is designed!

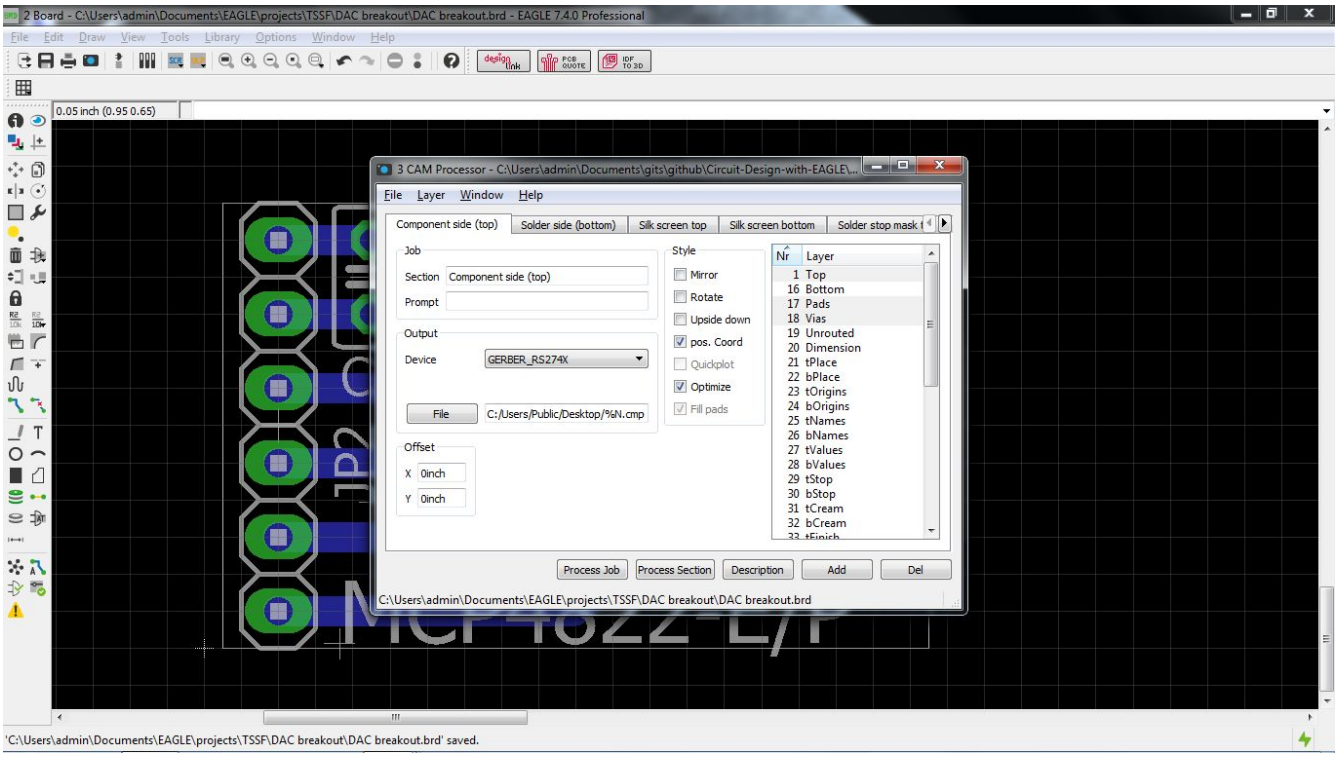
Appendix A: generate gerbers

Some PCB fabricators are beginning to accept straight .brd files, but for those who don't, the **gerber** is the standard format for boards. Gerbers are ASCII-readable vector files. If your fabricator doesn't accept .brd files, they will most certainly accept gerbers.

Go to *File → CAM Processor*



Once there, open the custom job associated with this project; it's in the repo.



Hit "Process Job" and it should generate several files on the Desktop. Zip all those files up into an archive, and you're good to go!