

Part 2: How are PCBs designed?

Ben Hurwitz, Spring 2024

Hi, and welcome to part 2 of The Hive's PCB Design With KiCAD tutorial series. My name is Ben, and in this video, I'll be going over how PCB design software is structured, and describe the basic design flow you might follow to go from an idea to a read-to-fab board. Let's get started.



PCB Design Software Overview

- PCBs are designed with specially-crafted software called, among others, “e-CAD”, “PCB CAD”, or “EDA”
- There are many different such applications with different strengths and weaknesses
- Understanding layers is key
 - Placing traces on a silkscreen layer will cause your circuit not to work (and you’ll be sad)



PCBs are designed with specially-developed computer-aided design software, known equivalently as e-CAD, PCB CAD, or EDA for “Electronics design automation”. Many different software applications and suites exist that can perform the required functions for this process, and all of them have strengths and weaknesses. However, the jargon and processes are generally very similar, meaning if you learn one, you have the tools to learn others with relative ease; it’s often just a matter of discovering the new locations of the necessary icons and settings.

The understanding of layers and how the layers are represented on screen versus their physical manifestation is critical. Place your components or object polygons onto the wrong layer and you’ll be very sad later (and probably a bit less wealthy).



PCB Design Views

- All such software involves making a *schematic*, and then arranging components on layers in the *layout/PCB*
- The *schematic* is the circuit diagram - what we might hand-draw on paper
- The *layout* (which KiCAD calls the “PCB Editor”) is what the circuit board actually looks like, in *layers*
- These views are inextricably linked together so that changes in one are appropriately reflected in the other

Note! The layout will often look *nothing* like the schematic!
And that’s fine – the schematic is for people to read, whereas the layout is for electrons to read.

PCB CAD software uses two primary views, the schematic and the layout, which can also be called the PCB view. The schematic is for the circuit diagram, what might be drawn on paper with symbols and lines for connections. The layout is the physical design of the board in layers – the size and placement of components, the actual location of routes, and the mechanical constraints. These views are inextricably linked so that changes in one will be reflected in the other.

It’s very important to recognize that the layout will often look nothing like the schematic. And that’s fine, because schematics are for people to read and understand, whereas the layout is for electrons to travel. These two views do not have the same goals.



Layers

- PCBs are designed in a stack of metaphysical layers that use polygons to indicate which objects are filling the physical space
 - Similar to Photoshop or Illustrator or other graphics design software
- There are many available layers in EDA software, many of which you will not need or use
- Here are some important ones (using KiCAD names):
 - F.Cu (and B.Cu) – the front/back copper
 - F.Silkscreen (and B.Silkscreen) – the front/back silkscreen
 - Edge.Cuts – the board’s physical outline/dimensions
 - Less used: Adhesive (for solder glue), Paste (solder paste), Mask (soldermask), and Courtyard (part body outlines)
- Software terminology will vary – Google is your friend!



As I mentioned before, layers are a crucial component to PCB design. Each layer in the software will either describe a physical layer within the actual manufactured stackup, such as copper or silkscreen, or an informational layer for the designer to add information for themselves or the fabrication house, such as dimensions, part names, project revisions, and more. These layers are similar to other graphic design tools, such as Photoshop or Illustrator, but they generally don’t have priority; overlapping polygon on different layers is typically acceptable (though not always).

There are numerous layers that you will need to become familiar with, though there are often many more within a given software package that are either left unused or you will not interact with. The most commonly used ones include those for the copper layers, the silkscreen layers, and the board’s outline layer. Less used layers include those for solder glue, paste, and mask, as well as part names and outlines.

The layer names given here are for KiCAD, whereas other software will likely use different naming conventions. Google is your friend here.



How do parts work?

- Virtually all components you add to your design will comprise two models: a *symbol* and a *footprint* (and maybe a 3D model)
- *Symbols* are the schematic representation of a part – e.g. the squiggle of a resistor, the coil of an inductor
 - Symbol connection points are called *pins*.
- *Footprints* are the physical form of the part – through-hole sizes, pitch (spacing), body shape, etc.
 - Footprint connection points are called *pads*. (Yes, even TH ones.)
- Some software only allow you to place *devices*, which are a fixed, pre-defined, symbol/footprint combo
 - This means that every different footprint for the same symbol require different devices -> e.g. dozens of different resistors
- Symbols, footprints, devices, and 3D models are stored in *libraries*.

Similar to how there are two views for the entire PCB, each part or component that are included in the design (even non-electrical ones) have at least two models: a symbol for the schematic, and a footprint for the layout. They may also have 3D CAD models as well for a 3D view or for exporting to mechanical design or simulation software. Symbols are the schematic representation of a part – think the squiggle of a resistor or the parallel lines of a capacitor – with pins to define the symbolic connections points. Footprints are the physical shape of the device, including the body, through-hole, or surface-mounted pads, and are digital representations of the component's package. The connection points for footprints are known as pads, even when they are through-holes. Some software requires you to link a symbol and footprint together into a single combined model called a device. All models are stored in libraries.



How do parts work?

- KiCAD does not have devices; in KiCAD, *any symbol may be associated to any footprint*
 - This allows a lot of flexibility, but also requires careful selection of footprints for every part to match pins and pads.
- Be careful to select the correct footprint



KiCAD does not use devices, meaning that any symbol may be associated with any footprint. This allows a lot of flexibility – you don't need to have a thousand different devices for resistors alone, and you can use the same footprint for many symbols very easily – but it also opens the door for designers to accidentally select the wrong package, and requires careful alignment of the symbol's pins with the footprint's pads.



How do parts work?

- Many standard components (e.g. Rs, Cs, Ls, diodes, transistors) have standardized symbols and footprints built into KiCAD that you can easily use
- ICs and non-standard components will likely not be
- If the symbol/footprint you need is not in KiCAD:
 1. Check the internet for them (work smarter, not harder)
 2. Draw them yourself
- We'll go through both methods later.



KiCAD, and most software, have many built-in libraries that can (and should) be used for standard parts, such as passives, diodes, and transistors. ICs and non-standardized components may not be as generic, and therefore may not be built into the software. One advantage of separating symbols and footprints is that a part may have a standard symbol or footprint but maybe not both; in KiCAD, that's easily handled, but in device-based software, it would be more challenging, and would generally require creating the other half of the device model.

In general, you should not be creating your own models; that should be the last resort because it's tedious, time-consuming, and prone to errors. There are dedicated companies out there that will create electrical models for free. Use these services instead to reduce the amount of busy work you have to do. I'll discuss both methods later.



Design Rules – ERC and DRC


- Both the schematic and the layout have a set of rules for allowable designs
 - Generally, the schematic rules (“electrical rules”) are fixed
 - The layout rules (“design rules”) can be changed depending on the fabrication house and/or requirements
- All CAD software can check your schematic/layout against the rule sets – the *ERC* and *DRC*, respectively

It's critical to run these checks throughout the process to avoid incompatible or nonfunctional designs!

Lastly, there are two sets of rules that your board must adhere to to be fabricated successfully. The first are the electrical rules that apply to the schematic, things like whether wires are connected and if pin types match. These are typically fixed and don't need adjusting. The second are design rules that apply to the layout, such as hole sizes and copper separation distances, and these come from your chosen fabrication house and design requirements. Any e-CAD software will be able to check your designs against the electrical rules, calls an ERC, and the specified design rules, called a DRC. It's super important to both make sure that your design rules are correct by reading your fab house's instructions carefully, and to run these checks multiple times throughout your design iterations to avoid error propagation and incompatible or nonfunctional designs.



Why KiCAD?

- Relatively low barrier to entry
- Good for introducing concepts without too much overhead
- Free and open-source
 - + Active community of support and developers
 - + Cross-platform operation
 - + No cloud operation for you to get locked into
 - No big company support
 - No fancy integrations with external software
 - A little more rough at the edges than industry-standards
- Customization with Python scripting. (Not covered here.) 

KiCAD, as I've mentioned, is just one such software suite that can make circuit boards. It has a number of advantages that I think makes it a good tool for learning this process. There is a relatively low barrier to entry relative to the industry titans, making it good for introducing concepts without all the extra overhead. It's free and open-source, meaning you can use it even if you're not with a large company, with a large and active community of support and development for cross-platform operation. Additionally, there is no cloud storage to lock your designs into. The lack of big company support may be detrimental for certain reasons, however, such as missing external integrations and advanced functionality, and it can be a little more rough at the edges due to the lack of dedicated development engineers. This missing functionality, and much more, can, however, be added through the use of custom plugins and modules, written in standard Python.



KiCAD tips and tricks

- Rolling the scroll-wheel will zoom.
- Clicking-and-dragging with the scroll-wheel will pan.
- The “Insert” key will typically repeat the last command.
- Most text inputs allow for math equations and unit conversion. Useful between metric and imperial.
- CTRL + F1 shows all the available hotkeys. Pretty nice.



KiCAD has a few program-wide shortcuts that are good to know about before we get into it. Like more CAD software, a three-button mouse is highly advantageous, with the scroll-wheel being used to zooming and panning. The insert key will typically repeat the last command (useful for placing multiple parts), and most text input will handle math expressions and unit conversions. Finally, there are many shortcuts, most if not all of which are customizable, and CTRL + F1 will show them all.



EDA Design Flow (Broadly)

1. Conceptualize the circuit. Breadboard/simulate, perhaps.
2. Part selection
3. Generate libraries and determine design rules
4. Create your schematic
 1. Add symbols to the schematic
 2. Connect symbols with wires
 3. Assign footprints to symbols (ERC!)
5. Layout your PCB
 1. Arrange footprints in layout
 2. Connect footprints with traces and planes
 3. Add finishing touches (silkscreen, teardrops, etc.) (DRC!)
6. Gerber/assembly files



Lastly, while designing any board has its own special requirements and every tool has its own quirks, PCB design broadly follows the following sequence.

First, you conceptualize and ideate the circuit, with breadboards or simulations.

Second, you select and obtain your key parts, and anything that's either specific to the design or hard to get.

Third, create your libraries and setup your project, including determining a fab house and locating their design rules and requirements.

Fourth, create the schematic with symbols, wires, footprints, and ERC using an iterative process to finalize your design into something cohesive and coherent.

Fifth, lay it out, spending the majority of your time on placing the components, then routing, and finally adding any finishing touches, while running your DRC liberally and frequently to avoid any last-minute major errors.

And finally, sixth, once you've iterated enough, satisfied your design requirements, and get a clean bill of health from your ERC and DRC, plot the required gerber and assembly files and send to fabrication.

Of course, there's more after that, like waiting, testing, debugging, more iteration, and programming, but that's generally what it takes to go from an idea to a board.



End of Part 2

And with that, we close the book on Part 2. We covered EDA software and how it works with a lot more terminology, introduced KiCAD, and gave a broad PCB design flow that we'll look to follow through the coming videos. A PDF of this video is available as well, linked in the description and hosted on The Hive's Wiki.

In Part 3, we'll start that process with an introduction to the circuit we'll be developing into a PCB, no electrical engineering knowledge required, and going through the process of part selection.

See you there!