Table of Contents

1 KiCAD Notes

1.1 Introduction

This document discusses issues associated with KiCAD to design circuits and printed circuit boards (PCB's.) It is meant to be instructive and constructive. I find KiCAD to be a useful tool and I am quite pleased that there is a team of people that donate their time and effort to maintain and improve it. In no way are any of the issues raised below meant to denigrate the program or the team that works on it.

1.2 Overview

This overview is not specific to KiCAD.

The vast majority of parts are made out of pieces of silicon call silicon dies. There are some passives like resistors, capacitors, inductors, switches that are not, but the silicon die case has the most issues.

A silicon die is made in a Silicon foundry and it has a number of bonding pads. The die is placed in a package and the bonding pads are electrically bonded to package "pins". Usually package pins are given a number, but some packages (like Ball Grid Arrays), have numbers and letters. The package has a part number printed onto it that uniquely specifies the vendor, package, and die. (Resistors use a color code instead of printed text.) These parts typically shipped out to numerous distributors who subsequently list the parts in the catalog with their own part number and pricing structure. The price typically varies with the total number of parts ordered and may have other constraints.

Some nuances on parts are:

- Sometimes multiple pads inside of the die are bonded together. This is extremely typical for power and ground on larger silicon dies.
- For microprocessors, sometimes a single pad can be used for multiple purposes.
- Some dies made from different vendors are equivalent to one another (second sourcing.) Thus, the circuit engineer can substitute one for the other without affecting circuit behavior.
- Sometimes a die contains multiple identical circuits withing the same chip -- e.g. multiple logic gates.
- Each part can have one or more PCB footprints. This is particularly true for parts with flexible leads.

The purpose of a schematic is to document a circuit design in a form that is more understandable by humans. A schematic consists of a bunch of visual components interconnected by lines. The lines ultimately correspond to traces on a printed circuit board (PCB) and the components ultimately correspond to specific parts. In addition to the schematic, there is a Bill of Materials (BOM) that performs the final mapping between the schematic components and part numbers that can be ordered from a distributor.

In the ideal world, an engineer generates a bill of materials followed by a schematic followed by a PCB. In the real world, there is back and forth as difficulties in the PCB layout can force changes in the schematic which can in turn change the BOM.

There are nuances associated with the schematic:

- A schematic typically represents the whole system which may be decomposed into multiple PCB's.
- A schematic will frequently span multiple pages.
- ....

The reality of PCB design is that it is an iterative process. Thus, there will typically be multiple revisions of the same PCB as problems are fixed and new features are added. Each board has a revision which is typically a letter (A, B, C, etc.) I strongly recommend that each PCB revision be stored in a separate directory. This way it is possible to easily refer to the previous revisions. A new revision is made by copying prior revision directory contents into a new revision directory. The contents of the new directory are morphed into the new revision. I personally name my revision directories "rev*a", "rev*b", etc. Once a PCB revisions is shipped off for manufacture, the contents of the revision directory are no longer modified.

1.3 KiCAD

KiCAD is an open source integrated schematic capture and PCB design program. It has a number of issues that are worth documenting and discussing. The issues discussed revolve around the general topics of:

- Component Libraries
- PCB footprints
- PCB generation

1.3.1 KiCAD Component Libraries

KiCAD component libraries are stored in .lib files, which are stored in directories. There can be multiple .lib files in a directory that are intermingled with other files such as schematics, boards, RS274X files, etc.

A KiCAD component library file name is one-to-one with the library name. Thus, the library name for the file "mylibrary.lib" is "mylibrary". Since libraries can be in multiple directories, it is possible to have multiple component libraries with the same name. This would be very confusing and really should be avoided.

KiCAD has has three basic GUI windows for managing Component libraries:

- Library Editor: The library editor is used to create, modify and store library components.
- Library Preferences Window: The library preferences window is used to manage the library directories search path and library list.
- Library Browser: The library browser is used to view the components stored in library files.

Since the Library Browser is "read-only", it does not need any further discussion.

The ultimate purpose of the component libraries is to provide components that can be placed into schematics. The engineer basically searches the component libraries until an acceptable component is found and places the component into the schematic. If an acceptable component is not found, either a new one is created from scratch or an existing one is copied and modified. The new components need to be stored back into a library.

It is never a good idea to store a new component back into the library files that come with KiCAD. The new component should be stored in a new library so that when the KiCAD libraries are updated, the new component is not lost. Unfortunately, when one of early KiCAD tutorials showed the creation of a new component, it showed the component being stored back into the original KiCAD provided library. By doing this, the tutorial skipped the issues around creating and managing libraries and but unfortuantely encourages developers to do the "wrong thing".

The library preferences window is used for managing libraries. The preference window provides an ordered list of library names and ordered list of library directories.

The ordered library directory list provides a search path for looking libraries. It is likely that this list is searched from top to bottom, but that could be incorrect. There is no way to change the search order. The search path order becomes a non-issue if you ensure that each library is uniquely named. It is a very good idea if each library has a unique name. Confusing things happen when there are libraries with the same name in different directories.

When a new library directory is added ordered directory list, I strongly recommend using a relative directory path. (KiCAD will provide a pop-up window that asks.) That way the entire project with all of the revision directories can be picked up and moved to another location without having to individually edit the library preferences to point to the new locations.

The ordered library list just lists the library names. Presumably it searches the list from top to bottom for component names. Unlike library names, duplicate component names are likely to happen, so order really matters. Unfortunately, the ordered library list does not list what directory the came from. (In other works, KiCAD provides plenty of rope to hang yourself on this issue.)

While it might seem intuitive to be able to create a new empty library and then start stuffing new components into it, KiCAD does not do it that way. Here is a sequence of steps that work:

1. Create a new component using the editor.
2. There is an icon on the editor (an open book with a star on it) that is used to store the component into a new library. You will get a pop-up that warns you that there is more work to be done.
3. Now bring up the library preferences window by clicking on [Preferences] => [Library].
4. Using the bottom portion of the window add the directory that contains the new library to the library list. I strongly recommend using a relative path. (KiCAD will ask you after you add a path.)
5. Now go to library list and add the library to the list. I recommend using the [Up] button to move it to the top.
6. Click on [OK].
7. Go [Preferences] => [Save Preferences]

If you skip the last step, steps 3 through 6 will need to be repeated each time you bring up KiCAD. After performing these steps, you can access your newly created component.

Now that you know how to create a new component library the question is where to put the new library. The following options come to mind:

- In the revision directory.
- In the project directory (one level above the revision directory.)
- Some centralized location where can be shared amongst multiple projects.

I have waffled on this issue over the years and have ultimately concluded that having one commong library that you share amongst all your projects is a good idea -- **BUT** right before you ship the design off to be manufactured, the libraries should be copied into the revision directory.

How do changes propagate through libraries and into the schematic? The answer is that I do not really know. I have been burned by thinking that once a component is copied into a schematic it never gets changed by upstream changes where the component came from. The bottom line, is you think you may come back to a revision at a later date, you should have everything manually copied down into the revision directory. This is a pain in the rear, but that is the only safe solution.

The next question is what components should be placed in the private library. Obviously any new components get placed there. In addition, I make sure that any component that is placed onto a schematic is first placed into my private library first. There are two reasons for this. If the KiCAD libraries get updated and change a component, my subsequent revisions will not look different. Also, it can be hard to remember just where a component came from when doing a PCB revision. If the part is copied first, there is no memory required. (Indeed, the person doing the revision may not be the same as who did the previous revision.)

So how are components copied? I use the following steps:

1. Set [Select Working Library] (the red/yellow/blue triple gate icon) to the "from" library.
2. Click on [Load component from current working lib] (the red gate with green arrow on top icon).
3. Set [Select Working Library] to the "to" library.
4. Click on [Update current component in current library] (blue arrow to red square icon).
5. Click on the [Save current library to disk] (floppy disk on top of open book icon).

It would be nice if it were bit easier, but in my opinion it is worth it to get all components into a single library.

That more or less covers component library management. The next set of issues concerns the components themselves.

1.3.2 KiCAD Components

Basically a KiCAD component consists of the following:

- A name
- A default reference designator
- A number of "pins"
- A bunch of additional drawing to supplement the pins.
- An optional description, keywords and documentation file
- An optional alias list
- And an optional footprint filter

Further Notes:

- For a left/right version of a component, the "de morgan" stuff can be used. The steps are:

    1. Create the one version of the part.
    2. Click on [Edit component properties].
    3. Check off [As Convert].
    4. Use the Demorgan radio buttons (next to PDF button) to switch between views.
    5. When both views look OK, save it.
- To support multiple versions of the same component.

    1. Create the "generic" version first and give it the generic name in the library.
    2. Make a copy of the component with the less generic name.
    3. Click on [Add and remove field and edit field properties] (large T icon).

        A. Mark the value field as not visible

        B. Add another field "Visible Part Name" and and set its value to the generic part name. Make

sure it is visible.
4. Get the visible part name where it is desired. The old value field is still visible in the editor, but it does not show up in the schematic.

1.4 Footprints

{Add stuff about footprints here.}

1.5 Requests For Enhancements:

- A design rule check for Schematic wires or PCB traces that are not vertical, horizontal, or at 45 degrees.
- A way to drag a via that is constrained to V/H/45.
- A way to restart a trace in trace V/H/45 trace drag mode. Right now I have to delete two segements worth and redraw.
- It is not possible to plant a via on the same line as an IC pad, if the pad is not aligned with a grid. Instead, a small wire segment is left behind that connects to the via.
- When dragging a segement in slope mode where one end connects to a via, the segment disconnects from the via. It would be nice to have the segment remain connected to the via.