Library management in KiCad version 5

Rene_Poschl Aug 29

Disclaimer: screenshots taken in nightly from between version 5.0 and 5.1. Version 5.0.x looks slightly different. Version 5.1.x will look like the screenshots show.

Jan 12

1 / 1
Jan 12

Short version (TlDr)

KiCad does not automatically detect libraries! There is no special library location (you might still care about good file organization). The path (variable) setup found in the preferences menu is for making your setup portable not to load a full directory of libraries.

Every library that should be available must be added to so called library tables which are managed with library managers. These are found in the preferences menu of all KiCad subprogramms (preference -> Manage symbol/footprint libraries.)

There are two tables for every asset type. One is global and the other local to the current project. (Project local entries have higher priority.)

Use the “add existing library” button of the library manager to add libraries using a file browser like tool that allows selecting multiple libs via shift or crtl plus left click.
Alternatively use the add library entry of the file menu within the footprint and symbol editors. This tool does not allow to add multiple libraries at once. (Use it to add a sinlge downloaded library to KiCad.)

KiCad splits footprint and symbol libraries. Meaning if you downloaded a lib then you will most likely have gotten a footprint and a symbol library. You need to add both of them to their respective library table.

Jan 12

For more details continue reading.

Introduction

Official KiCad documentation (Disclaimer: parts of the official docu especially on the footprint side are out of date at the time of writing. Meaning some menus look a bit different. But you should still be able to find the correct tools from the information given as long as you accept slight changes in the wording used to reference them from the menu.):

- **http://docs.kicad-pcb.org/5.1.2/en/pcbnew/pcbnew.html#footprint_editor_managing_libraries**
- **http://docs.kicad-pcb.org/5.1.2/en/eeschema/eeschema.html#manage_symbol_libraries**

The KiCad library system separates symbols and footprints in their own libraries. This does add a lot of flexibility but comes with its own set of challenges.

Symbols abstract the function of a component and communicate the interface of it to both KiCad and the person reading the schematic.
Footprints define the physical interface between the pcb and the component (The land pattern) and also include documentation information (outline, polarization mark, reference, …)
Here a more in depth explanation about the **differences between symbols and footprints.**

File types

Symbol libraries

Every symbol library consists of two files. The .lib file defines most of the symbol (graphical elements like pins and lines, symbol fields like the default footprint, …) The .dcm file holds the information that can be specialized for every symbol alias. (Description, keywords and datasheet link)

Most kicad internal dialogs will only show the .lib file. KiCad knows that there also is a .dcm file with the same file name.

Footprint libraries

The current footprint file format has one .kicad_mod file per footprint placed in a folder with .pretty suffix representing the library.
In some rare instances you might also find the pre version 4 file format. Back then a single .mod file represented a full library. (similar to how symbol libs are handled right now.) KiCad can still include them in the library setup in the same way as a its modern files.

KiCad can also directly use eagle xml libraries to extract footprints. (To do that add the library file to the footprint library table.)

3d model libraries

KiCad supports wrl and step files. It is customary that these files are organized in .3dshapes directories. (Kind of like libraries but KiCad does not really manage them. More on that later.)

The library nickname (library identifier)

The nickname is used as an abstraction between KiCad and the filesystem. Every symbol (or footprint) is uniquely identified by the combination of library nickname and symbol (or footprint) name. The colon (":") character is used as a delimiter between them.

Library tables

Libraries are not available from within KiCad by simply placing them in some specific file system location. (One major difference to how eagle handles libraries.) Every library must be explicitly added to so called library tables. (fp-lib-table for footprints, sym-lib-table for symbols) These library tables are managed using library managers. See below for an in depth explanation of these tools.

There is a global and local version of both of these tables. Every library added to the global table is visible for all projects whereas adding a library to the local table makes it only visible from the current project.
The local table entries are used if a library with the same nickname is in both the global and local library table.

“Managing” 3d models

3d models are not managed in any way. Footprints directly point to a 3d model file. It is therefore recommended to use path variables for the 3d path settings of footprints.
This means care must be taken when placing models on the file system. It is suggested to place all your models in subdirectories of a common folder. That common folder is where your personal 3d model path variable might point to.

More details about adding 3d models
A suggestion for a file system organization can be found further down. (Including a suggestion for which environment variables to use and where to point them.)

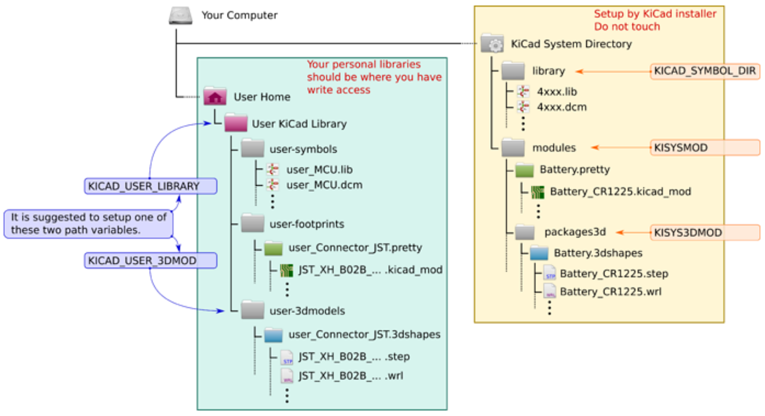
Where should global libraries live within the file system.

KiCad does not really care where you place your library files as long as you point to them appropriately. The only restriction i would suggest you hold yourself to is that you do not place personal libs into the KiCad system path directory. Furthermore do not add personal symbols, footprints or 3d model libraries into one of the KiCad supplied libraries. (The next update of KiCad would delete your work if it is within the system directory.)

I would suggest that you put any personal library into some central location within your personal user home directory. (A KiCad-library subdirectory within your documents folder for example)

Setup a path variable to either your main library directory or your 3d model library to allow your footprints to be portable. (As mentioned in the 3d model section above.) I would suggest for a single path variable to your main library folder. (Allows your full setup to be portable instead of just having 3d models portable.) Path variables are managed in the `KiCad main window -> preferences -> configure paths`.

Also remember this is just a suggestion. If you do not like this suggested setup then you can use a different one.



Personalized setup for the official library

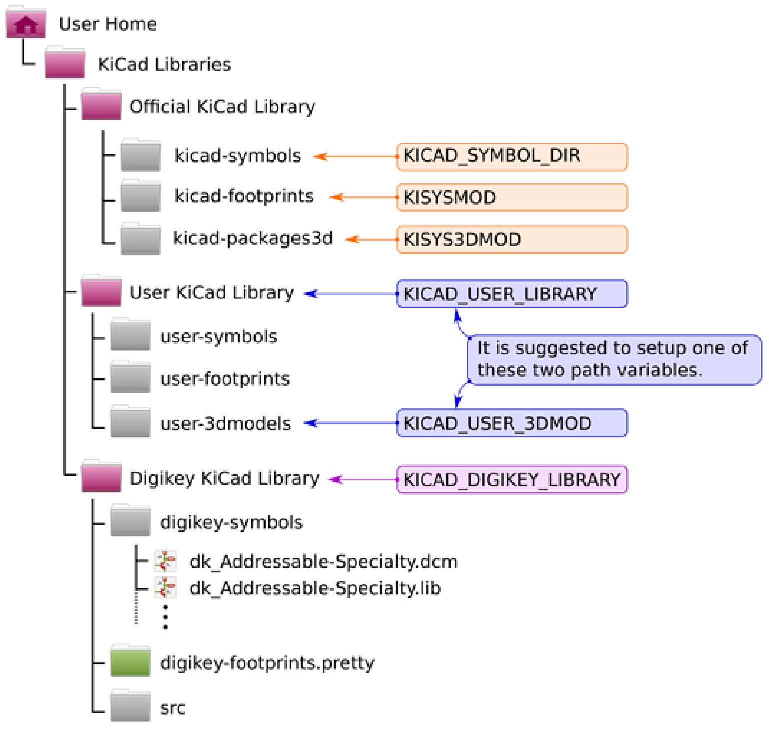
KiCad does come with official libraries that are by default installed (if you choose to do so) somewhere in a write protected area. There might however be reasons to deviate from this.

One reason would be if you want to have more control over which version of the official library is used in your setup. In that case placing your version of the official lib into your user directory makes a lot of sense.

You can choose to download a specific version of the lib directly from github or clone the repo with git allowing you to easily change between versions and download new ones. (more details about that further down.) The later option can save you data volume as well as git only downloads what changed. (A local git clone makes contributing to the library easier as well.)

To get this non standard location of the official lib to work you either need to point the path variables to these new locations (and use the default library tables) or use the library manager as explained further down to remove the old libs and add the new ones. Remember that you will need to set the 3d path variable (KISYS3DMOD) in any case.

My suggestion would be to use the default library table and pointing all 3 path variables to the new library location (as shown in the picture.) This will allow you to keep the official library table including the description of libraries while still being able to have your libraries at the location of your choice.



A suggestion for the placement of third party libraries (like for example the digikey library) is included in this screenshot. These third party libraries are added in the same manner as your personal libs. This tutorial will therefore not go into any more depth on them.

Adding a library to a library table

This section handles the additon of libraries that you already have on your disk. Creating new libraries is explained in great detail in separate FAQ articles:

- **Creating a new symbol library and a new symbol in KiCad 5**
- **Creating a new footprint library**

The most powerful way to manage your libraries are the library managers found in the preference menus of all programs. Libraries can also be added using the add library buttons of the symbol and footprint editor.

Using the library manager

I show the symbol library manager in this section. The footprint library manager has the same user interface. Meaning you should be able to understand that tool without a special section about it.

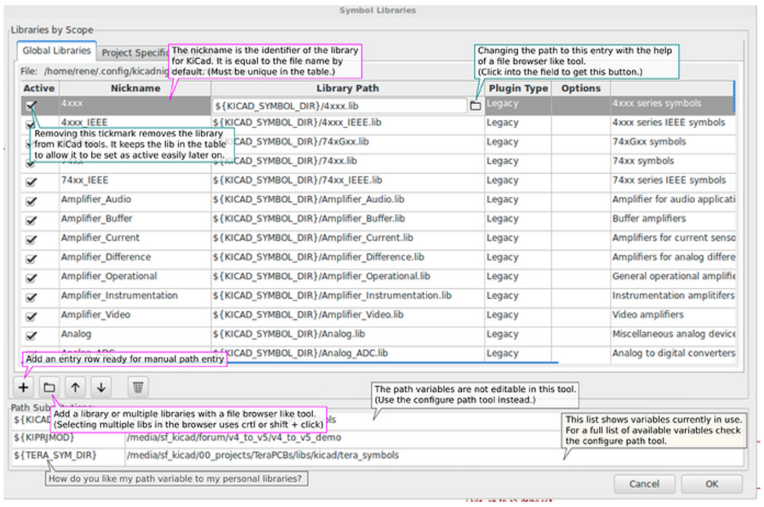
The library managers are found in the preference menus of all KiCad sub programs and in the KiCad main window. (Menu entries are “Manage footprint libraries” and “Manage symbol libraries”. Only the main window shows both options. Sub programs show only the one relevant for them.)

The library managers are used to handle both the project local and the global library tables. Switch to the one you want to edit by selecting the appropriate tab. (Above the table view of the table entries.)

Use the browse button (looks similar to a typical open button in v5.1) to add libraries with a tool similar to a file browser. Navigate to the location of your libraries using this tool. You can use shift or crtl plus click to select multiple libraries.

You can also use the add button to add a single empty entry to the table.

Note: The “Path Substitutions” view in this dialog is not intended to be a duplicate list of the configure path tool. Its purpose is to give an easy way to see which substitutions are currently performed. To edit the variables or to see all available variables check the configure path tool found in the preference menu of every kicad subprogram.



File: /home/rene/.config/kicadnig

The nickname is the identifier of the library for KiCad. It is equal to the file name by default. (Must be unique in the table.)

Changing the path to this entry with the help of a file browser like tool. (Click into the field to get this button.)

Active	Nickname	Library Path	Plugin Type	Options	
✓	4xxx	${KICAD_SYMBOL_DIR}/4xxx.lib	Legacy		4xxx series symbols
✓	4xxx_IEEE	${KICAD_SYMBOL_DIR}/4xxx_IEEE.lib	Legacy		4xxx series IEEE symbols
✓	[illegible]	${KICAD_SYMBOL_DIR}/74xGxx.lib	Legacy		74xGxx symbols
✓	74xx	${KICAD_SYMBOL_DIR}/74xx.lib	Legacy		74xx symbols
✓	74xx_IEEE	${KICAD_SYMBOL_DIR}/74xx_IEEE.lib	Legacy		74xx series IEEE symbols
✓	Amplifier_Audio	${KICAD_SYMBOL_DIR}/Amplifier_Audio.lib	Legacy		Amplifier for audio applicati
✓	Amplifier_Buffer	${KICAD_SYMBOL_DIR}/Amplifier_Buffer.lib	Legacy		Buffer amplifiers
✓	Amplifier_Current	${KICAD_SYMBOL_DIR}/Amplifier_Current.lib	Legacy		Amplifiers for current senso
✓	Amplifier_Difference	${KICAD_SYMBOL_DIR}/Amplifier_Difference.lib	Legacy		Amplifiers for analog differe
✓	Amplifier_Operational	${KICAD_SYMBOL_DIR}/Amplifier_Operational.lib	Legacy		General operational amplifie
✓	Amplifier_Instrumentation	${KICAD_SYMBOL_DIR}/Amplifier_Instrumentation.lib	Legacy		Instrumentation amplitifers
✓	Amplifier_Video	${KICAD_SYMBOL_DIR}/Amplifier_Video.lib	Legacy		Video amplifiers
✓	Analog	${KICAD_SYMBOL_DIR}/Analog.lib	Legacy		Miscellaneous analog device
✓	Analog_ADC	${KICAD_SYMBOL_DIR}/Analog_ADC.lib	Legacy		Analog to digital converters

Removing this tickmark removes the library from KiCad tools. It keeps the lib in the table to allow it to be set as active easily later on.

Add an entry row ready for manual path entry

Add a library or multiple libraries with a file browser like tool. (Selecting multiple libs in the browser uses crtl or shift + click)

The path variables are not editable in this tool. (Use the configure path tool instead.)

This list shows variables currently in use. For a full list of available variables check the configure path tool.

Path Substitutions:

${KICAD	[illegible]
${KIPRJMOD}	/media/sf_kicad/forum/v4_to_v5/v4_to_v5_demo
${TERA_SYM_DIR}	/media/sf_kicad/00_projects/TeraPCBs/libs/kicad/tera_symbols

How do you like my path variable to my personal libraries?

Using the symbol or footprint editor add library menu entries

Both the symbol and footprint editor have an add library entry to the file menu, the top toolbar and as entry to the right click context menu in the treeview. (This is new in v5.1 for the footprint editor.)

Clicking this button will open a browser like tool that allows adding a library. You will be asked if you want to place the library into the local or global library table after selecting a file to add.

Appendix

Location of library tables within the file system

You should not need to edit the library tables with anything other than the library managers. You might however still benefit from at least knowing where they are found.

The global library tables is in the config directory of your operating system user.

- Linux: The user’s home directory + .config/kicad (= $HOME/.config/kicad or ~/.config/kicad)
- Windows XP: “C:\Documents and Settings\username\Application Data” + kicad (= %APPDATA%\kicad)
- Windows Vista & later: “C:\Users\username\AppData\Roaming” + kicad (= %APPDATA%\kicad)
- OSX: The user’s home directory + /Library/Preferences/kicad

The local library tables are found in the project directory.

Resetting the library tables to default settings

The current version of KiCad does not come with an automatic way to reset the library tables to the default settings (**See feature request for “Reset lib tables to default”** . There is however a workaround possible. Simply delete the library tables from the file system using your operating systems file browser. (A better option might be to rename them in a way that allows you to find them later if something goes wrong. For example add .backup to the filenames)

After that start-up KiCad. When starting pcb_new you will be informed that the default library table has been setup.
Starting eeschema will bring up a dialog asking if you want to copy the default library table, create an empty one or use a custom one. The default option will bring you back to factory settings.

If you do not get the default setting suggested then you do not have the library installed. A way to fix this is explained here: **I installed KiCad 5 (under Linux) but there are no libraries. (The default option for sym lib table setup is disabled)**

You could also download the library tables directly from github and manually overwrite the one in your config directory instead of deleting it.

You might even be able to only reset the entries belonging to the official library by using a text editor. This might however be more work than worth. (Adding your personal libs back should not be that hard if you have them all in one location.)

Updating the official library

KiCad does not update your personal settings on any update. This means you would miss out on libraries added after your first installation.

With the standard setup you can go with the normal reset to default options explained above.

When using the advanced setup with path variables setup you would need to overwrite the library table in your config directory with the ones you downloaded with the library.

Both of these will require you to add your personal libs back after doing this.

Another option would be to open the library tables with a text editor and remove any line defining a library of the official library. Then opening the library table shipped with your current version of the lib with the text editor and copying all lines (except the first and last one) into your library table in the config directory.

This really only pays of if you have many personal libs in different locations. (Adding any number of libs from a single location is easy) Or if you have descriptions added for your own libraries.

Options to download a particular library version (for the advanced setup)

Download zip archives from github

Every github repo can be downloaded at every release point and at the tip of the current master branch. The KiCad repos have a release corresponding to every KiCad release.

The releases can be found here:

- https://github.com/KiCad/kicad-footprints/releases
- https://github.com/KiCad/kicad-symbols/releases
- https://github.com/KiCad/kicad-packages3D/releases

Using a local git repository

Using a local git repository gives you more powerful options and can save on data volume as git only downloads what changed when you decide to get a newer version of the library.

You will need to understand a few git commands to use this way of downloading your libraries.

git clone [link to repo] (for getting the repo locally)
git pull (for getting the current state)

And possibly also **git checkout [branch or tag]** to switch to a particular release.
Checkout only works on the local repo. Meaning you might need to first download the current state of the online repo if you want to switch to a newer release. This is done with the `git fetch` command

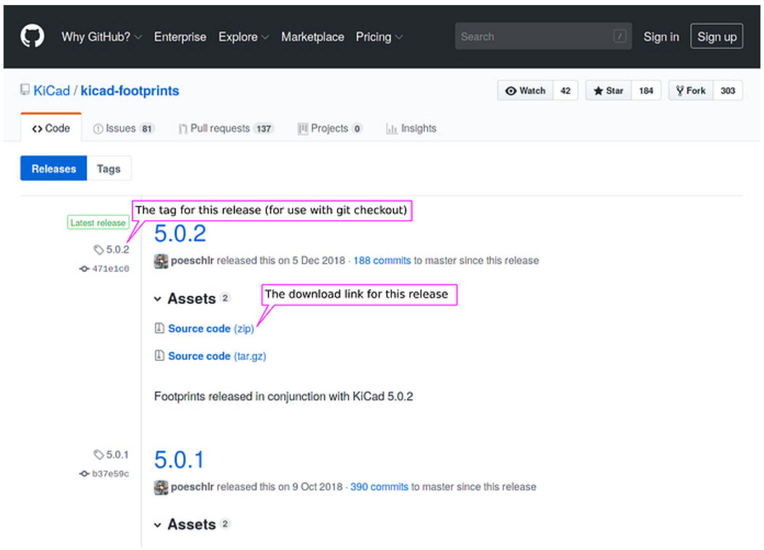
Example (Checkout a release):

```
git fetch origin # to download the current state of the online repo
git checkout 5.0.2 # to get the library to the state tagged as 5.0.2
```

To find out which releases exist look under the release pages of the repos (see previous section for the links.)

Example 2 (get current head of master -> the development snapshot)

```
git checkout master # Switch to master branch.
git pull # Download current online state.
```



KiCad / kicad-footprints

5.0.2
471e1c0

5.0.2

poeschlr released this on 5 Dec 2018 · 188 commits to master since this release

Assets 2

Source code (zip)

Source code (tar.gz)

Footprints released in conjunction with KiCad 5.0.2

5.0.1
b37e59c

5.0.1

poeschlr released this on 9 Oct 2018 · 390 commits to master since this release

Assets 2

Project Local Libraries

It does not really make sense to rely only on project local libraries. The main downside of such a setup would be that you will have a harder time finding already existing assets. (Reuse becomes basically impossible as soon as you have more than a handful of projects.)

Even using project libs as a local cache might not really be the best idea. This could easily lead to having different versions of the same asset floating around in different project libs. (It increases the points at which you can introduce an additional error.) Let kicad take care of caching the assets used (footprints are included in the pcb file, symbols in the cache lib file)

3d models do not really play nice with a project local setup in the current version of kicad. (This is because footprints point directly to a file. Including 3d models would mean you need to edit the path settings inside of every footprint pointing to that model.) But 3d models are sadly not cached by kicad right now. Meaning make sure you do not change 3d models in the global lib without good reason.

Symbols with the footprint pre filled also play not nice when copying a global footprint to a project local library (Unless you keep the same footprint library name. -> This would however make it impossible to get any footprint from that global library in the future.)

Project specific symbols for micro controllers, connectors, power symbols, …

One usecase for project local libraries are project specific symbols for configurable parts. To get the best out of ERC you might want to specialize the electrical types of your connector and micro controller symbol pins. You might also want to specialize the pin names to make it clearer which alternative function to setup for a particular MCU pin or to make it easier to see what is connected to which connector pin.

Another use would be to have project specialized power symbols (To get better fitting net-names.) The reason for needing specialized symbols here is that the resulting net-name is controlled by the pin name of the symbol.

Archive a project for sharing with others (read only access)

Project local libraries can make sharing projects (for reading) easier as the reader does not need to setup libraries that way.

The reason why you would like to have at least symbol and 3d model libraries included in such an archive is because kicad does not reliably cache symbols and does not cache 3d models at all within the project files. (Symbols are only cached in the cache lib and rely on the reader using the rescue tool properly.)

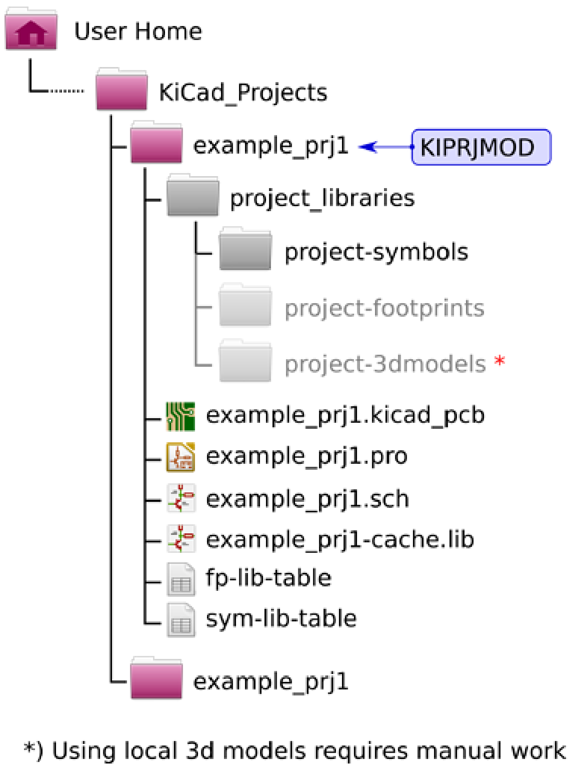
A detailed explanation on how to archive projects is out of scope here. The short answer is that this is really only viable with scripts. See for example: **https://github.com/MitjaNemec/Kicad_action_plugins/blob/master/archive_project/action_archive_project.py**

How to setup a project with local libs

The suggestion for local libraries is to place them somewhere within the project file structure. (Placing them somewhere else will negate all benefits of project local libraries.)

The symbol (or footprint) libs would then be added to the project (local) library tables using the library manager. More details see above. (Instead of selecting the global library tab select the project library tab. Everything else stays the same.)

Make sure you use the path variable KIPRJMOD for project local libraries. It always points to project root. (The library managers should do this automatically.)



Why such bad documentation?

Git and the libraries for newbies

I had KiCad 4 installed previosly. Now i updated to v5. Now i have some problems with the library setup

Kicad 5.01- Installation- can't locate libraries

GitHub Tutorials For Noobs?

100 more