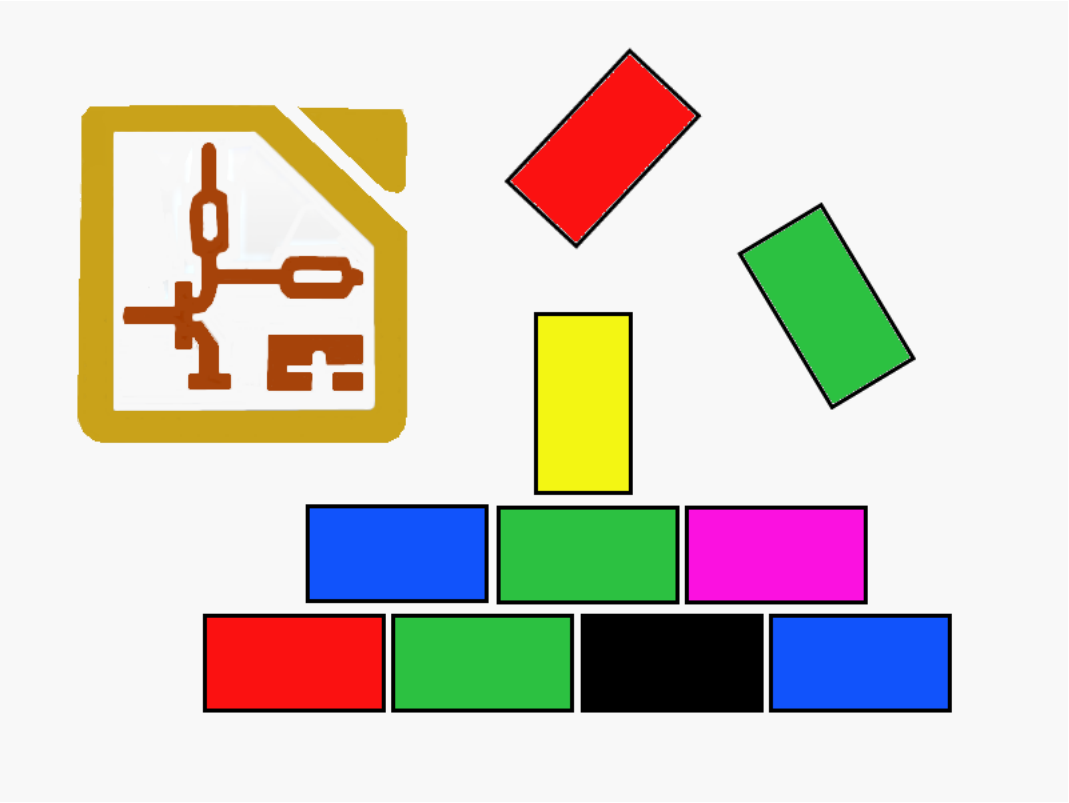

Hierarchical schematics as schematic buildingblocks at KiCad
Rev. C - Entwurf

Dipl. Ing. Bernd Wiebus alias dl1eic

May 6, 2015

Written with LaTeX
and Texmaker 3.3.4.

Contents

1 IMPORTANT NOTICE

This text is a translation of "Hierarchische Schaltplaene als Bausteine in KiCad RevB (German) vom 19 Dezember 2013".
Due to the quick progress of KiCad, some parts may not be actual anymore. So here is no notation of the canged starting of CVpcb, the altered behaviour of "append board" at PCBnew and some other minor issues.
Also the figures are not from an english but from a german version.
So regard this text as a draft.

2 Preface

Preliminary and incomplete/not ready!
No responsibility is taken for the correctness of this information!
Take this information witg a pinch of salt!
In the following text should be explained how you can use hierarchical schematics to create schematic building blocks for the schematic editor EEschema which is part of KiCAd. EEschema and KiCad are created by Jean-Pierre Charras and a team of programmers all over the world under a GNU license.

This hierarchical schematics can be used as buildingblocks, to create new schematics in a quick and easy modular way.

But i want to avoid the expression "modul" for this buildingblocks, because it is used at KiCad for footprints. So these expressions could get mixed up. But the expression "buildinblock" may be precise enough.

KiCad itself backs this options only by its flexible and transparent concept, but not with special functions. Therefore you have to use workarounds and makeshift solutions for some problems.
Those tutorial here referes to:
Eeschema Version: (2013-11-29 BZR 4513)-product Release build
Platform: Linux 3.2.0-4-686-pae i686, 32 bit, Little endian für Linux Debian Wheezy.
Due to the quick progress of the work of Mr. Jean-Pierre Charras and the KiCad team, this tutorial may be outdated.
This text is a translation of
HierarchischeSchaltplaeneAlsBausteineInKicad RevC 23Dec2013.pdf
from December 23th 2013. It is written in german.

More information on KiCad can be obtained from here:
http://iut-tice.ujf-grenoble.fr/kicad/
and, more actual
http://www.kicad-pcb.org/display/KICAD/KiCad+EDA+Software+Suite.
A Wikipedia article about KiCad can be seen here
http://en.wikipedia.org/wiki/KiCad

table of contents

3 Hierarchical schematics and how they can be used as buildingblocks for the quick and easy creation of new schematics

KiCad supports hierarchical schematics. They are intended, to divide big and complicated schemtics into smaller schematics , called "subschematics" for making them clear and less complicated. But in reverse, they an also be used to create quick and easy new schematics from available schematics for often used circuit parts.
As an example: many circuits use an input rectifier with smoothing capacitors, fuses etc. which is often followed by a linear voltage regulator to stabilisate the output voltage. If you have a big, confusing schematic, so it might be useful to divide it by putting those circuit parts into seperated, smaller subschematics. At the main schematic, there will only remain a box with some connecions and a link to this subschematic.
But if you have created such hirarchical subschematics once, it will be easy to use them at new schematics, where you want to use the same groups of devices. You will get more quick at creating schematics. Also you can use the same or similar group of devices multiple times at your schematic. You only have to put this subschematic multiple times to your main schematic .
At last, you could drive it so far, that your entire main schematic consists only of subschematics and there connections.
Because KiCad separates the device as a symbol at the schematic from the used Footprint (Modul) at the board, the used technology doesn't matter. You will paint the device as a symbol at the schematic, and than you can decide later (at KiCad at the programm CVpcb), wich special footprint and technology you will use. But you have to take some care, making this all working without kinks. Sometimes you will have to copy and rename some schematics and related files by hand in the filesystem of your pc.
At moment, the use of hierarchical schematics as buildingblocks is only a lucky windfall profit which are nice to use, but not intended as buildingblocks. Maybe, KiCad will support this possibilities with some tools in former days or with special export formats,which would be add much more comfort, also regarding boards and footprints. Read about this at Ausblick

table of contents

4 The creating of buildingblocks as hierarchical schematics at KiCad

Of course, you could obtain schematics from other sources than your own hand, but somehow there as to be created an adequate schematic at KiCad. At this point, i assume that you know the basic functions of KiCad/EEschema, but i will repeat a short example. The key aspect will be the creation of a schematic as a reuseable buildingblock.

4.1 A

Start KiCad and crate a new, empty project . There for choose at the upper Toolbar on the left "file > new > empty project". At the folder "'Testproject"' a projectfile "'Testproject.pro"' will be created. Look at figure 1. The name suffix ".pro" is mandatory for a projectfile at KiCad. After this, you will find the file "Testprojekt.pro at the project folder. Up to this, there will no assignment to this file. Look at figure 2.

4.2 B

For creating a schematic, start the program Eeschema EESchema (Figure 3). Probably you will get a error message at first, because the schematic "testproject.sch" doesn't exist yet. Just quit this error message, and you will get a new, empty schematic. If you save the schematic once, and leave EEschema, this schematic will be saved at the folder testproject as testproject.sch . You can create more, schematics, assosiatet to this project, by creating them with EEschema. Therefore choose from the upper toolbar at File either "New" or "save actual schematic as" like usual as modern pc programs. See figure 4.

4.3 C

Draw the schematic as usual. If you want to use the schematic as a buildingblock, you should make ALL connections to the outside as hierarchical pins. With some few exceptions, i suppose it might be dangerous if you use global labels at buildingblocks from the origin on.
The reason is complexity. Imagine, you have the ground at the buildingblocks defined as a global GND. Now perhaps you have a situation, where

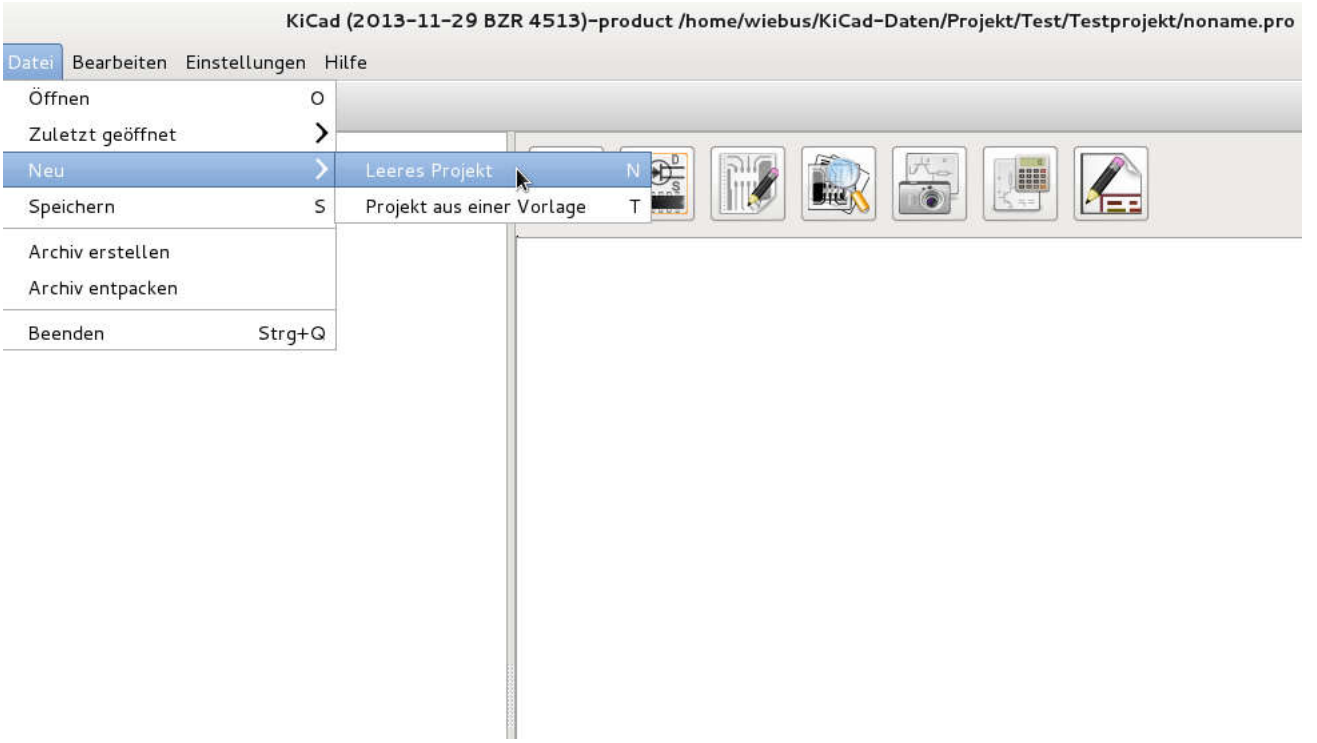

Figure 1: Creating a "Testproject" in KiCad

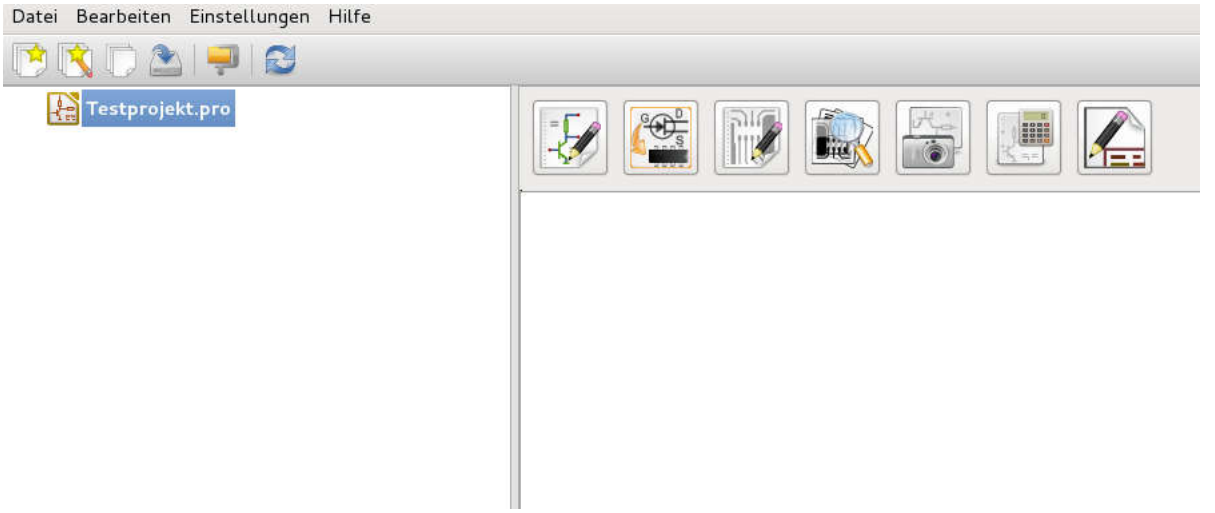

Figure 2: Creating a "Testproject" in KiCad: result

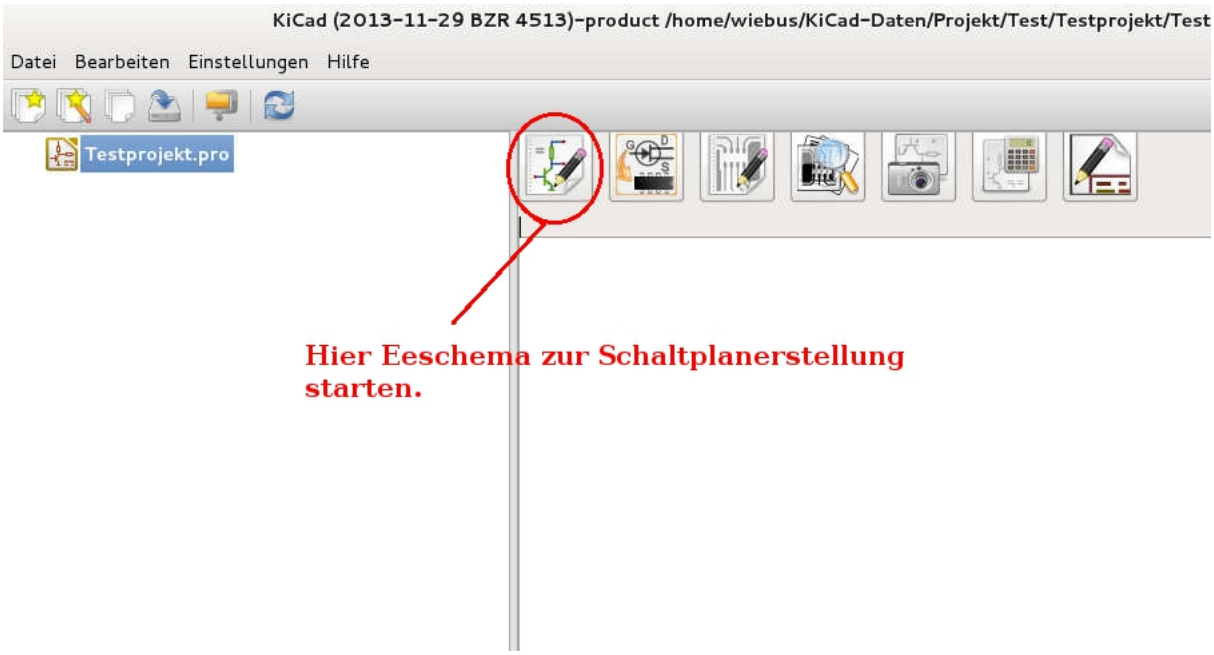

Figure 3: Start Eeschema for creating a schematic.

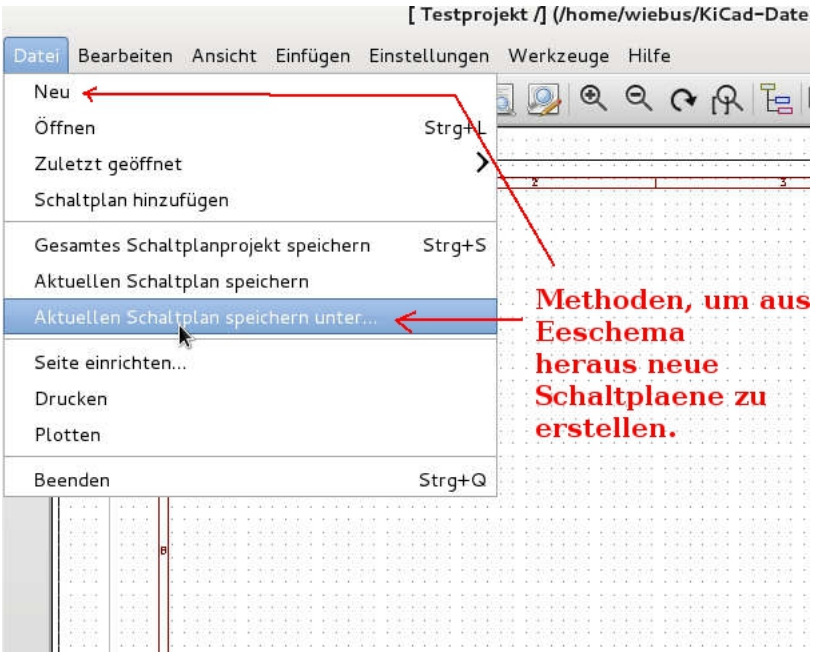

Figure 4: Creating new schematics from Eeschema.

you want to use this building block several times, but with relation to different ground systems. You will probably get a problem, because all grounds are tied together globally, but it is not obvious, that it is so, and why and where it happens.
For creating a hierarchical label you should use the button at the right toolbar (Figure 5). Those hierarchical labels will become later, if you merge the building block into your schematic, interfaces to this schematic. See figure 12.

A completed buildingblock for a LM317 regulator is shown at figure 6 as 317Regler.sch. It shows a schematic of a standard LM317 linear voltage regulator. You will find the file 317Regler.sch at the Folder "Testprojekt" where you either copied it or created it. I copied it from another folder, where i hafe a storage of many of such prefabricated building blocks.

KiCad expects to find the schematic files of hierarchical subschematics at the same folder, where you keeps the whole project and also the main schematic. . This folder is called the "project folder".
Also it is opportune, to copy this file into the projectfolder from a separate storage folder, because of safety and less complexity. If you have to tweak this subschematic for fitting into your main schematic, you want this file as copy at your project folder. If you alter it, this altering will affect only this local copy, and not the original file at the storage folder. So you can use the same buildingblock other where without being canged and corrupted.
Look out for the 317Regler-cache.lib file. This file will be created at last, if you save the original schematic and leave EEschema. Diese "-cache.lib" Achten Sie bitte auch auf die Datei 317Regler-cache.lib. Sie wird spätestens angelegt, wenn Sie den (Original)Schaltplan speichern und EEschema verlassen. Diese "'-cache.lib"' contains a symbol library with the symbols used at this schematic. It is important if you will take a library into another projekt for using it there. , where perhaps another symbollibrary exists, which will not fit to your library. Those "XXX-cache library" should have the same name with the exeption of the "-cache.lib" part aas the schematic file (normally the projekt name). This for the reason of easy overlooking and less complexity. It is not mandatory.
The project file of the building block is not needed for this purpose, with the exeption if you want to manipulate the building block original files.

table of contents

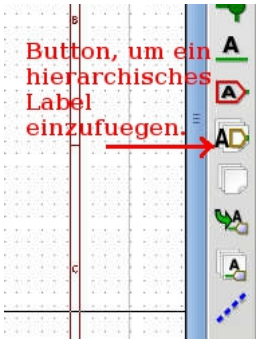

Figure 5: insert a hierarchical label

5 The insertion of buildingblocks into a schematic

5.1 A

Now you want to put this prefabricated buildingblock 317Regler.sch into a mainschematic "Testprojekt.sch" as a hierarchical schematic . Thor this, it has already to exist, regardeless of which it originates or is created.
Let us suppose, you will use this linear regulator three times at this schematic. So you have to create a hierarchical subschematic , by activating the corresponding button, like shown at figure 7. Now create the subschematic as a rectangle, by leftclicking its future left upper corner and than leftclicking its future right lower corner.
After clicking the last corner, a window will pop up, where you can insert the name of the schematic file and the name of the schematic. See figure 8. As default, a uninterchangeable but nothing saying file and schematic name is created by an algorithm . But you shoulde not use this, but better choose a memnonic name, to rerecognize schematic and the schematic file. Even don't choose the name created ba kicad, if you do not want to reuse this subschematic as a buildingblock.
File name and schematic name are not mandatory the same. at reality, this can only be, if you use this block only once at the schematic. The schematic name is allowed only once at a schematic, but several different subschematics with different names can reference to only one and thesame schematic file. Because here at this example, we will have more subschematics referencing to one schematic file, we will name the first subschematic as "Sheet317Regler-1" and referencing to the already existing file "317Regler.sch".
Now hit <Enter>. kiCad will notice, that the choosen schematic file exists already, an will ask, wether we will create a schematic from its contend .

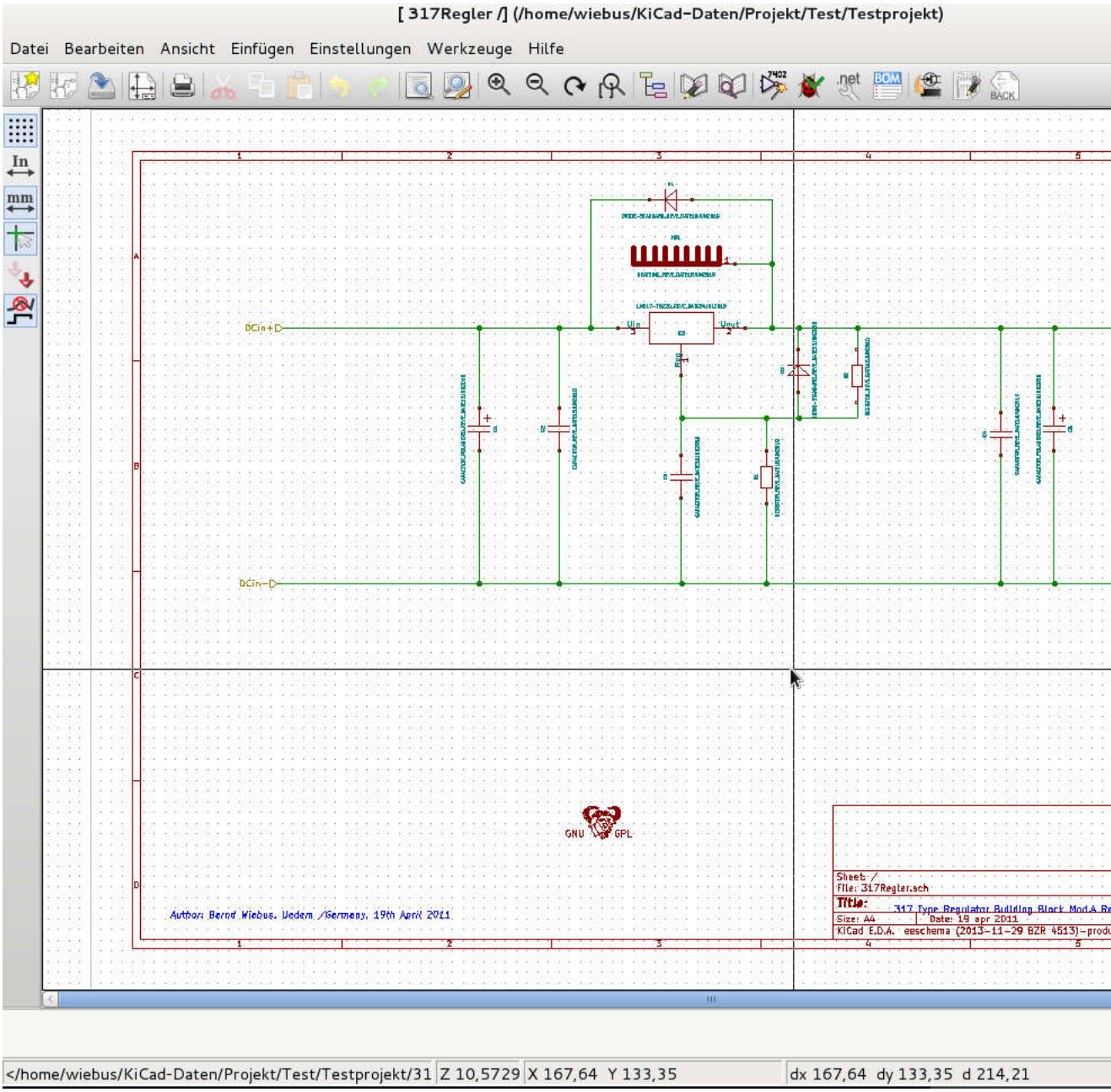

Figure 6: A schematic at KiCad

Figure 7: Button for creating a subschematic

Confirm this with "Yes".
Now you have created a mostly completed subschematic, like you can see at figure 9.

table of contents

You can open this subschematic either by doubbleclick to it, if you have NOT activated a tool, or by choosing the navigate button from the upper tool bar (figure 10), where you get a overlook of the used subschematics by there hierarchical tree.
If you do so, you will see, that all symbols are replaced by boxes with questionmarks . This means, that the above mentioned cache-library is not inserted to the list of used librarys (figure 11). To isnert the library into the library list, choose >library from the upper tool bar. A window pops up. there you can choose the button "add" from the right side, and choose from the appearing file-menue the cache-library for the symbols appropriate to the subschematic file.
At this example case the file "317Regler-cache.lib" at the folder "Testprojekt" . Now the subschematic should look like figure 12.
Here you have to remember that the original of the subschematic is tied with hierarchical labels to the superordinated schematic .
If not, you should do it now. But i suggest to do it every time if you design a subschematic, especial if it is designed for use in a library of buildingblocks.

table of contents

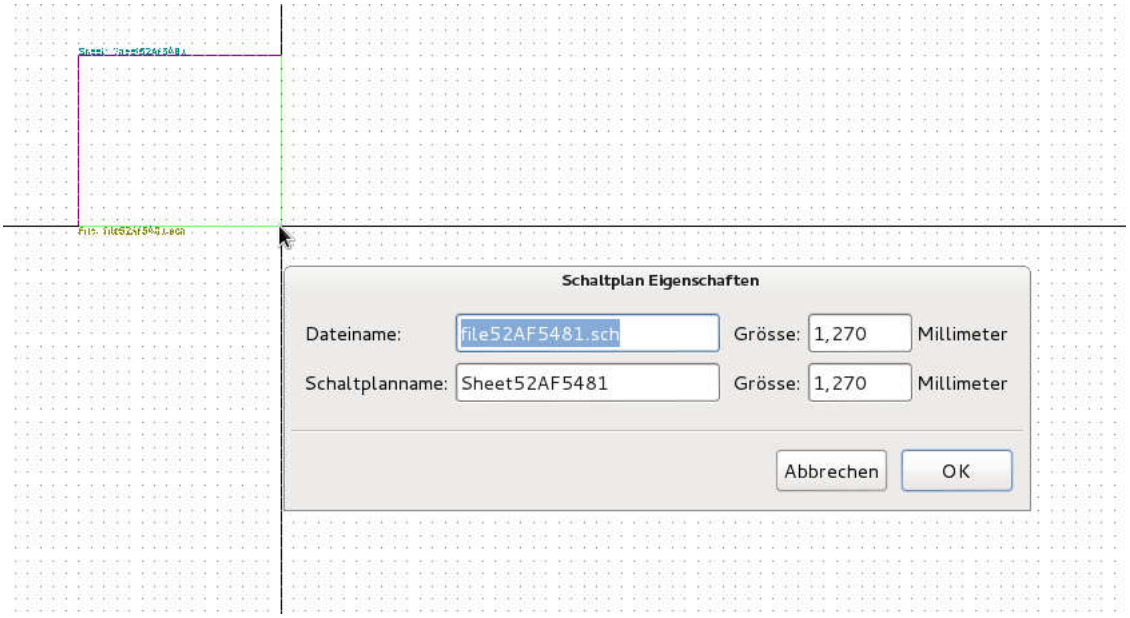

Figure 8: Creating a subschematic

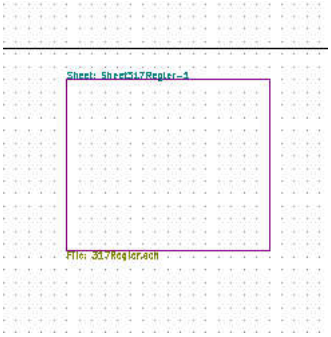

Figure 9: Subschematic ready

Figure 10: Button for navigation between hierarchical schematics

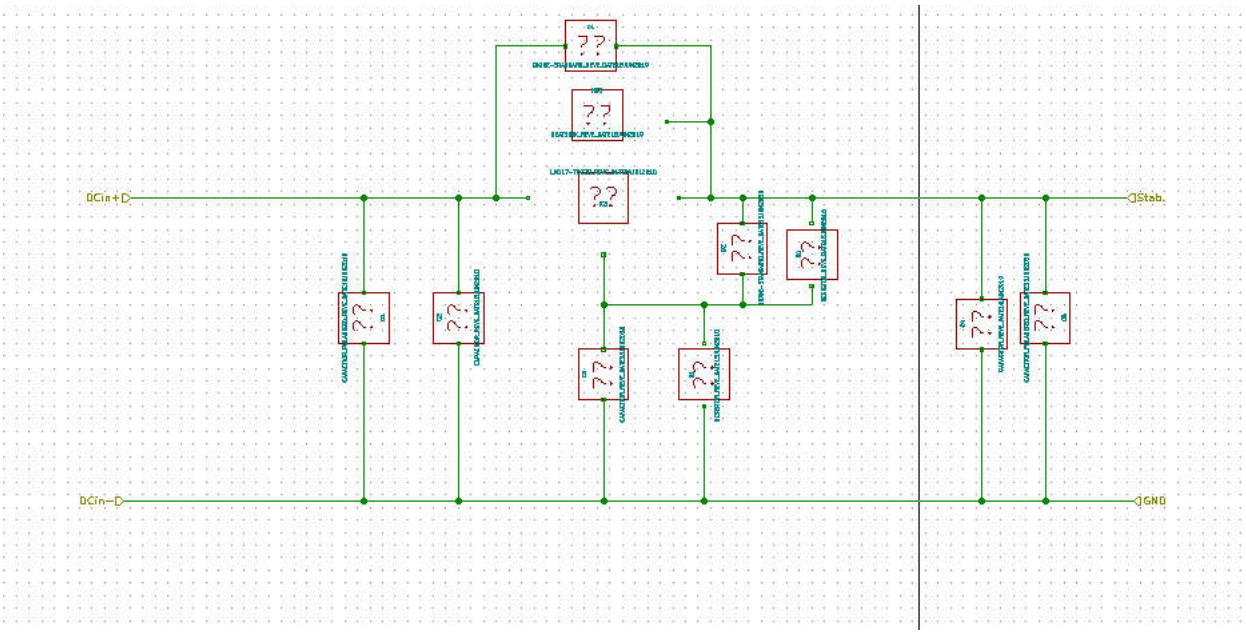

Figure 11: Missing Cache-Bibliothek for the symbols at a subschematic

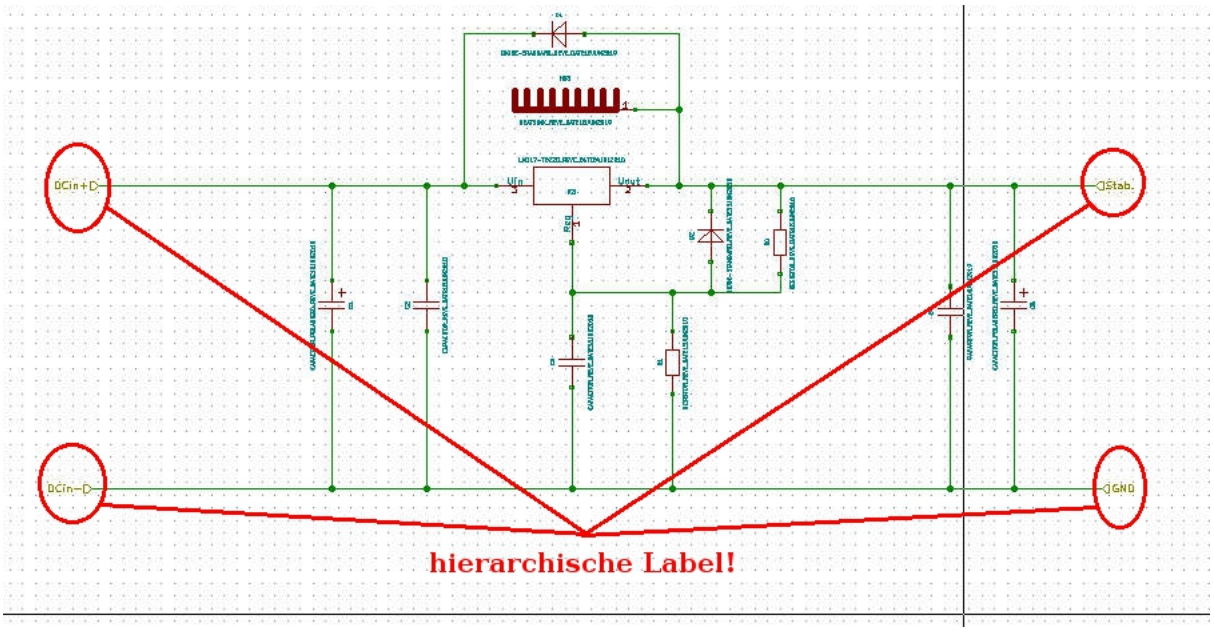

Figure 12: Hierarchical subschematic with hierarchical labels

5.2 B

At the main scheamtic should be used three of this hierarchical subschematics. One already exists, so we have to create the another two. This can be done the same way as the first, but referencing all to the same "317Regler.sch" file like the first, but choosing different schematic names like "Sheet317Regler-2" und "Sheet317Regler-3". . but this is perhaps a little bit clumsy. It is simpler, if you just mark the existing subschematic as group, and place them as a copy. After this, you have to open this new created subschematic by a right doubbleclick for editing, and change the schematic names to "Sheet317Regler-2" und "Sheet317Regler-3".
Now save the whole schematic project.
Remember: The name of the schematic is not mandatory identical to schematic file.

table of contents

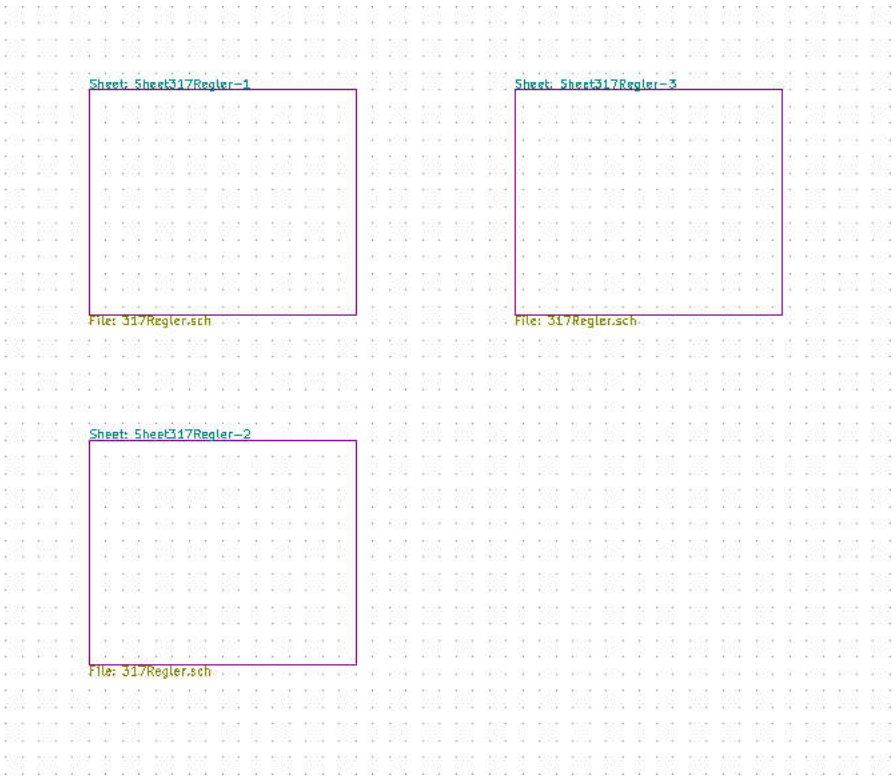

Figure 13: Three subschematics

6 Summary: hierarchical schematics as "Buildingblocks"

The use of buildingblocks simply is the use of hierarchical schematics by using schematic files which already exist prefabricated.
Therefore you reference to this existing file at the creating of a hierarchical subschematic.
This already existing schematic file has to be at the project folder, and also the path to the symbolcache of this schematic file has to be inserted into the list of the used symbollibrarys.

table of contents

7 Alternate procedure/procedure for subsequent modifications

The alternate succession, first the creating of hierarchical subschematics and then the adding of buildingblocks also works, but is more clumsy. But it is useful to know this way, too, because it must be taken if you want to alter existing schematics.

Therefore create the hierarchical subschematic first. Then choose from the upper menue bar the menue "file" and then "save all schematics". There will be created empty subschematic files with the choosen schematic name. Now close Eeschema, choose your favourite filemanagement program and replace all those files with files of the same name, which are created by copying and renaming from the building block original files. Do not forget to insert the Symbol-cache files into the list of used symbol libraries. This could be happened, if you see somewhere questionmarks instead of symbols , like shown at figure 11.

table of contents

8 The wiring of the buildingblocks

To connect this buildingblocks, go to the main (root) schematic and choose the import of hierarchical from the right tool bar. (Figure 14). Now click

Figure 14: Button for importing hierarchical Pins from subschematics

into a subschematic, and you get a hierarchical label from this subschematic as a pin at the "outside" of the subschematic. . You can move it with the mouse, and place it with a mouseclick. Repeat this, until you have all connections of the hierarchical schematic as pins at the outside at the symbol of the subschematic. Look at figure 15. Those pins corresponds to the hierarchical labels inside the subschematic. Those pins can be placed and wired with each other and with other devices like normal pins at KiCad. Look at figure 16.

table of contents

9 The dependencies between footprints and values - basics

If you create two or more subschematics, which references to the same buildingblock, this is not a problem at first. The annotation can be aware of those teo ore more schematics. Devices at the same position at different subschematics will be counted correct. and shown with those references multiple times at the netlist. . You can allocate them with footprints at CVpcb werden. You can also allocate different footprints/modules to devices with different referencenumbers. Wether this is useful or not, you have to decide at the situation.

But because this subschematics are connected to the same schematic file and there is only one value allocated, so it is forced, that different devices

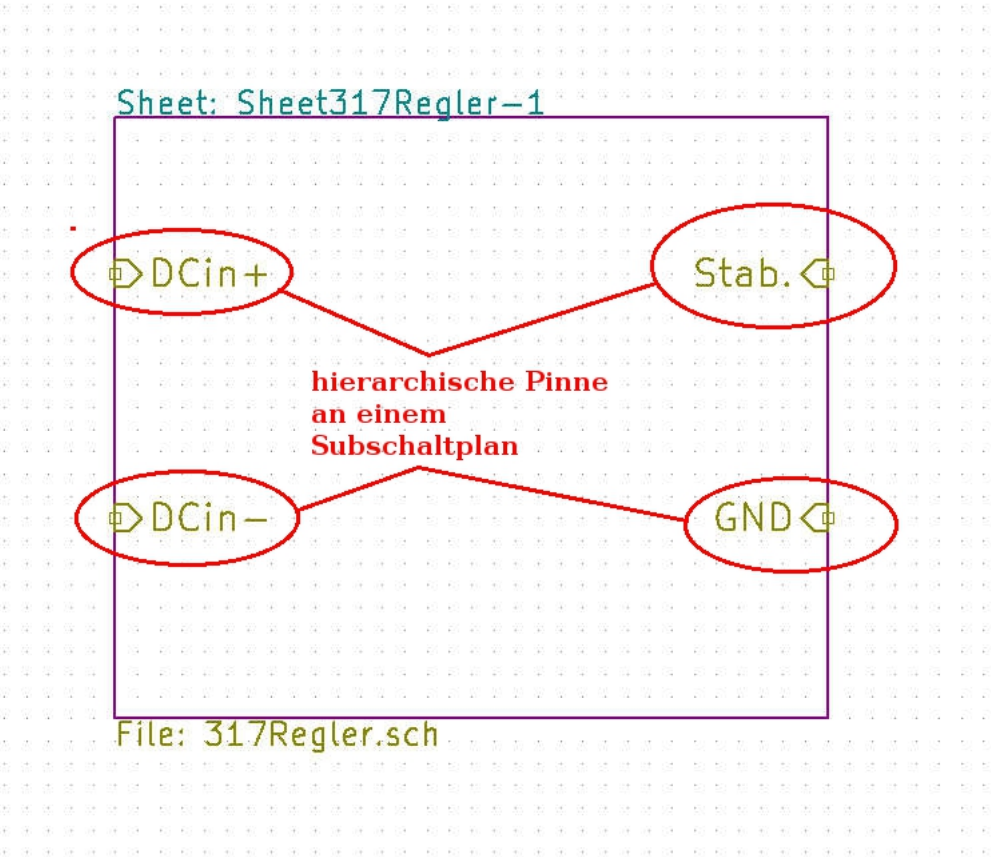

Figure 15: Subchematic with hierarchical pins

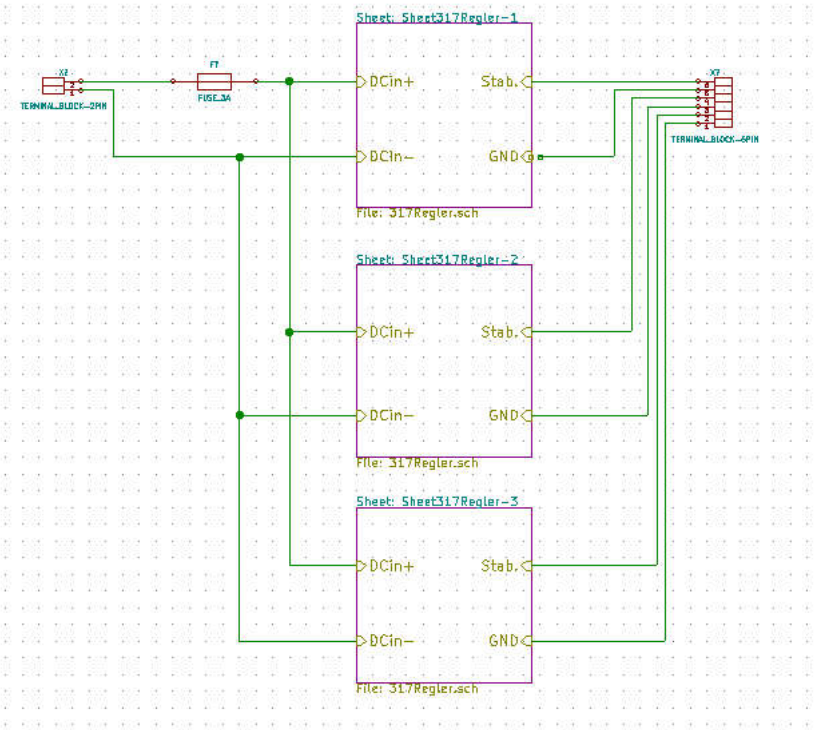

Figure 16: wired subschematic

with different references, are connected to only one value, despite they can be allocated to different footprints.

It depends on your personal style and "modus operandi" in handling values and footprints at the BOM ("Bill of materials") how to copy with this. in my opinion, different footprints with resulting different technology but same values (like all 100 Ohm, 100R) should get different "values" in the meaning, that the value should contain information about the footprint (like "100R, SMD 1206" or "100R, SMD 508") .

The straightest solution would be to create a copy of the subschematic with a different name, even if the subschemtaics differences only minor, despite the differences occur at the value (than this copy is absolute necessary, or if the difference at the footprint , and than reference to this copy. Than the subschematics differ not only in the names, but the schematic files, too, and in this different files can exist different values, too.
How this is done at the detail can be seen at the following example.

table of contents

10 The allocation of footprints and values - example

10.1 Annotation

We will use the autoannotation function on this set of schematics. . You can start it with the button shown at figure 17. You will get a menue, where you can do some adjustments. See figure 18.
"annotate" will start the autoannotation. If you get warnings about "multiple item" , like figure 19, you will have some references used double or multiple. At our case, this happend, because the used schematic "317Regler.sch" already had a existing annotation , who exists now multiple times, because we copied this schematic .

There are basically three methods to handle this problem.
A) Use a original schematic without annotation at all. But at our case, it is too late.
B) So could clear the annotation. . Look at figure 20.
C) You can allow Eeschema to replace annotations, if they are used multiple

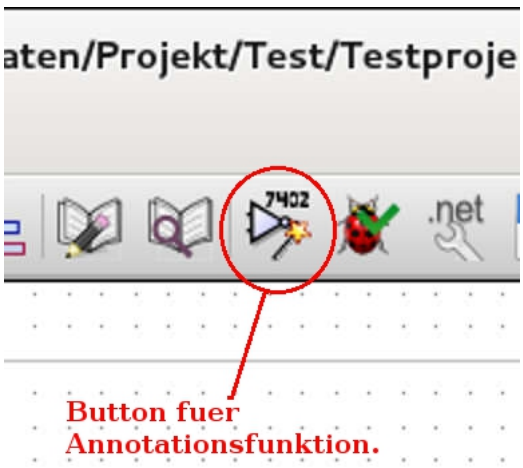

Figure 17: Button for auto annotation

times.. Look at figure 20 at the middle.

For this example case i would suggest method C. Method B and C are similar somehow, and at this case, it is the same result.
Method A is difficult to maintain straight. Theoretical you could store the libraries without annotation, but practical you need them, if you alter this schematic or if you show it to someone other. Probably you will often forget to clear the annotation after your work....

table of contents

10.2 Annotation - oriented to subschematics

If you use subschematics it will be useful to adapt the structure of the annotation to the subschematics, so the can guess from the number to the subschematic. KiCad can do this automatically, if you choose at "annotation choice": "Start to sheet number*100 and use first free number" or "Start to sheet number*1000 and use first free number". see figure 21.
The effect is as following:
All devices at the main schematic get one at the first digit of their reference number. All devices at the first subschematic get two at the first digit of their reference number. All devices at the second subschematic get three at the first digit of their reference number.

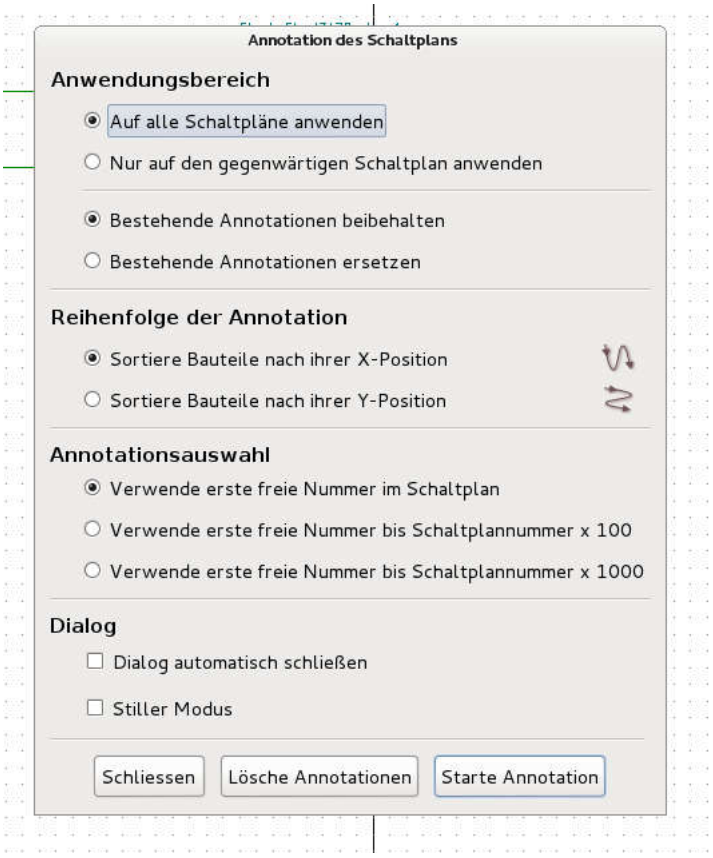

Figure 18: Menue for auto annotation

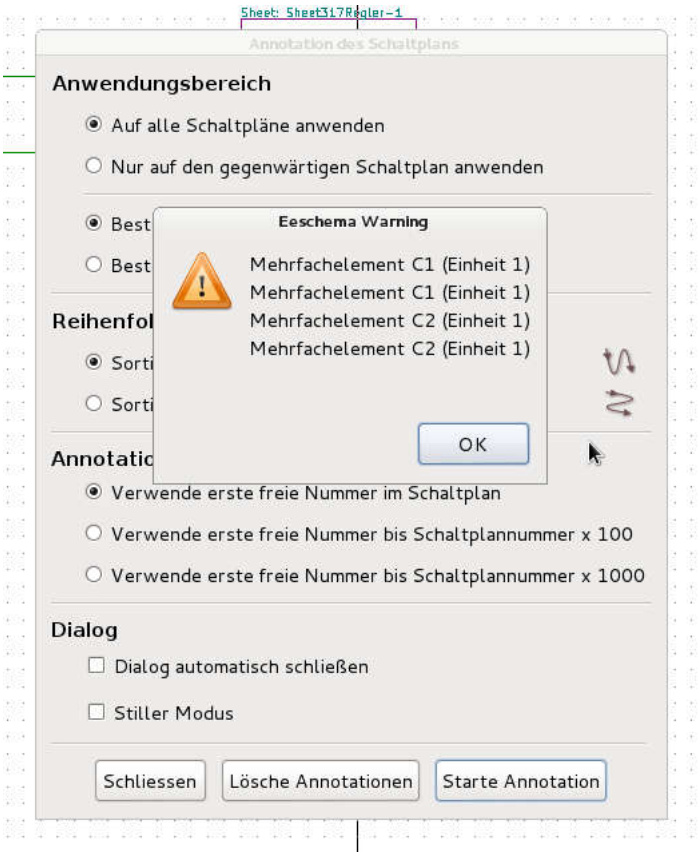

Figure 19: Warning at multiple used references

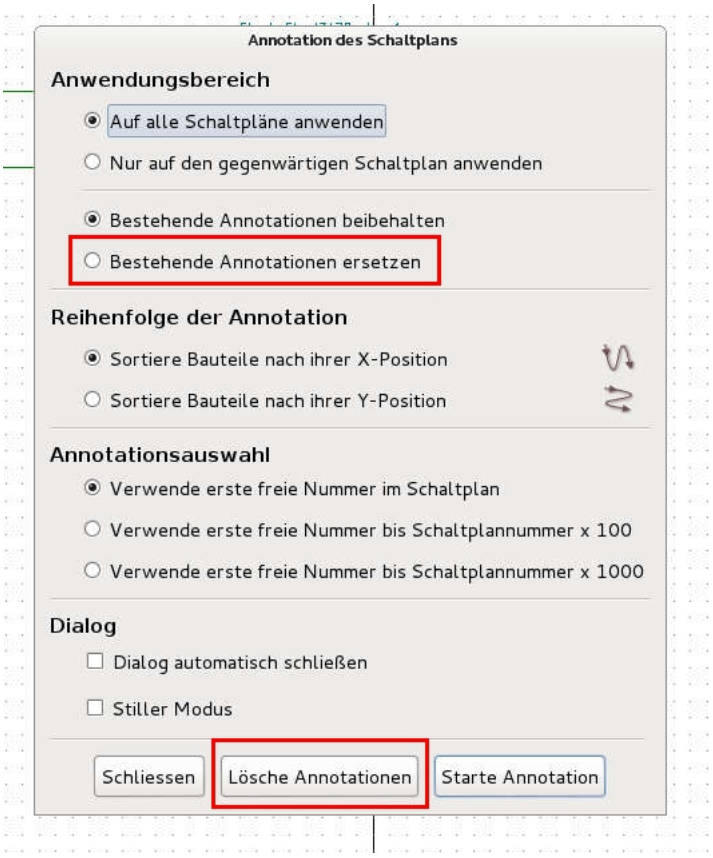

Figure 20: Annotation replacement or clearing

And so on...
KiCad does this differented to different prefixes. All resistors at the first subschematic get the reference numbers R201 to R299, all capacitors get C201 to C299. Dis means, for each prefix and subschematic 99 devices can get annitated in a structurated way, if you choosed "Start to sheet number*100 and use first free number". For whom this is not enough, there can be choosen "Start to sheet number*1000 and use first free number". So you get 999 free Numbers per prefix .
If you violate this constraint, because a prefix is used more than 99 or 999 times, KiCad reacts flexible. For this prefix the counting goes further on like the natural order of numbers, but the other prefixes will do as usual.
If the counting of the prefixes can cope again with the number of used prefixes, because there might be subschematics, which contains this prefix not or less often, the scheme will be used again. .
At this example the method "Start to sheet number*100 and use first free number" is choosen. So you can later guess the subschematic by the reference number of the devices. "Sheet317Regler-1" gets the numbers with two hundred, "Sheet317Regler-2" the numbers with three hundred and "Sheet317Regler-3" the four hundred numbers. The one hundred numbers references to the main, or root, schematic.

table of contents

10.3 Allocation of values

You can allocate values to the first subschematic (Sheet317Regler-1) by open it, click right to the values and choose "edit". You can see the effect at figure 22. If you look at the appropriate places at this subschematics (Sheet317Regler-2 and Sheet317Regler-3), you will see the same values at corresponding devices. Figure 23 and figure 24. Only the reference numbers are different. But this is not astonishing, because all reference to the same schematic file"317Regler.sch".

table of contents

10.4 Creating a netlist and allocate footprints

Now you have to create a netlist. . You can do this by the button like figure 25. Use the default values (figure 26). Then open CVpcb (figure 27) , and

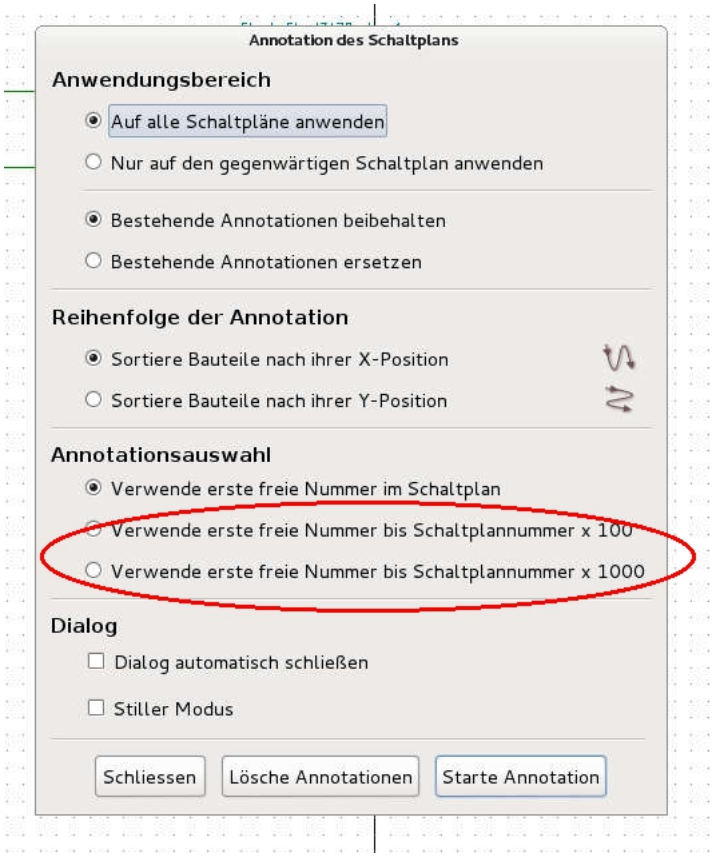

Figure 21: Structurated auto annotation

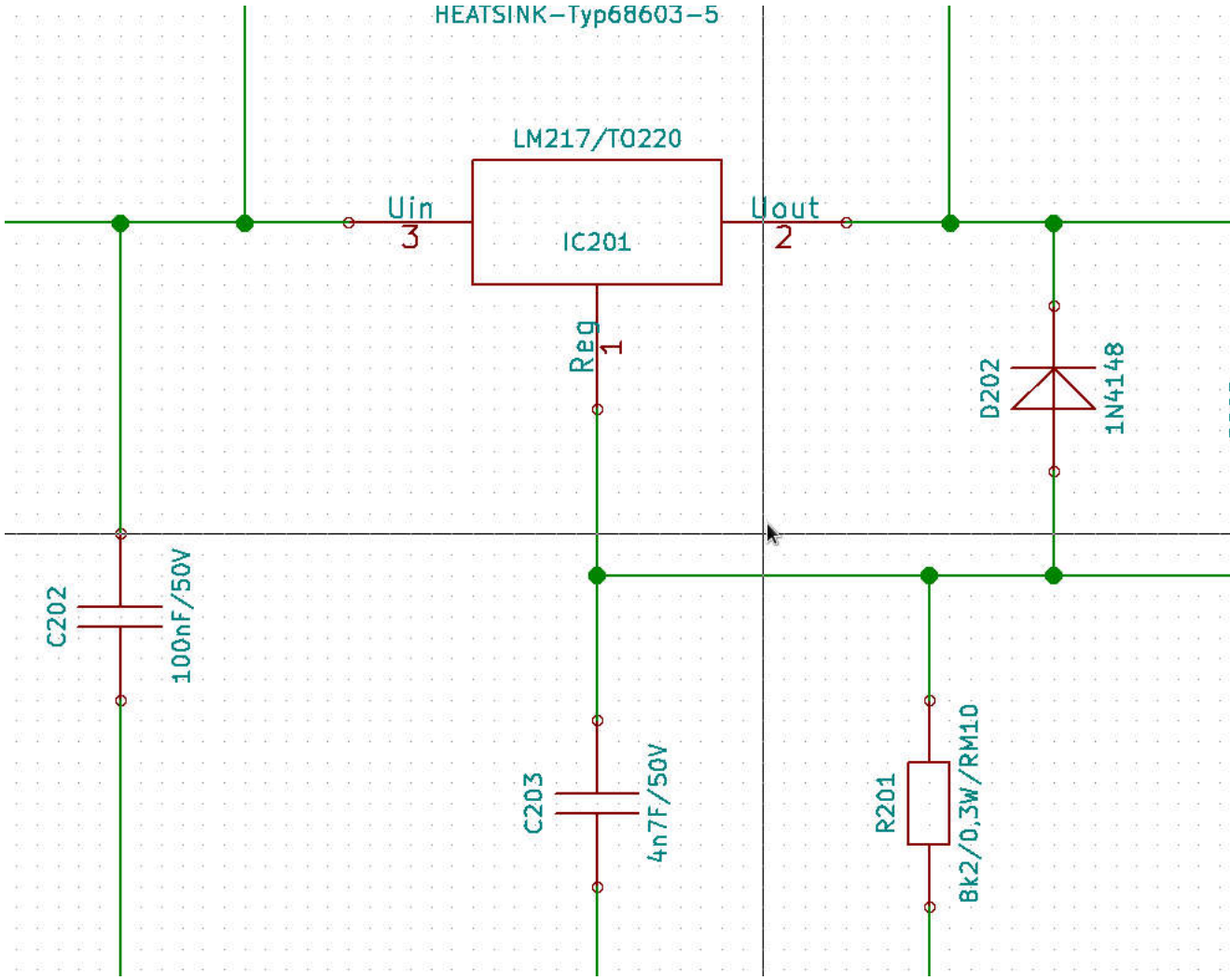

Figure 22: Device references at Sheet317Regler-1.

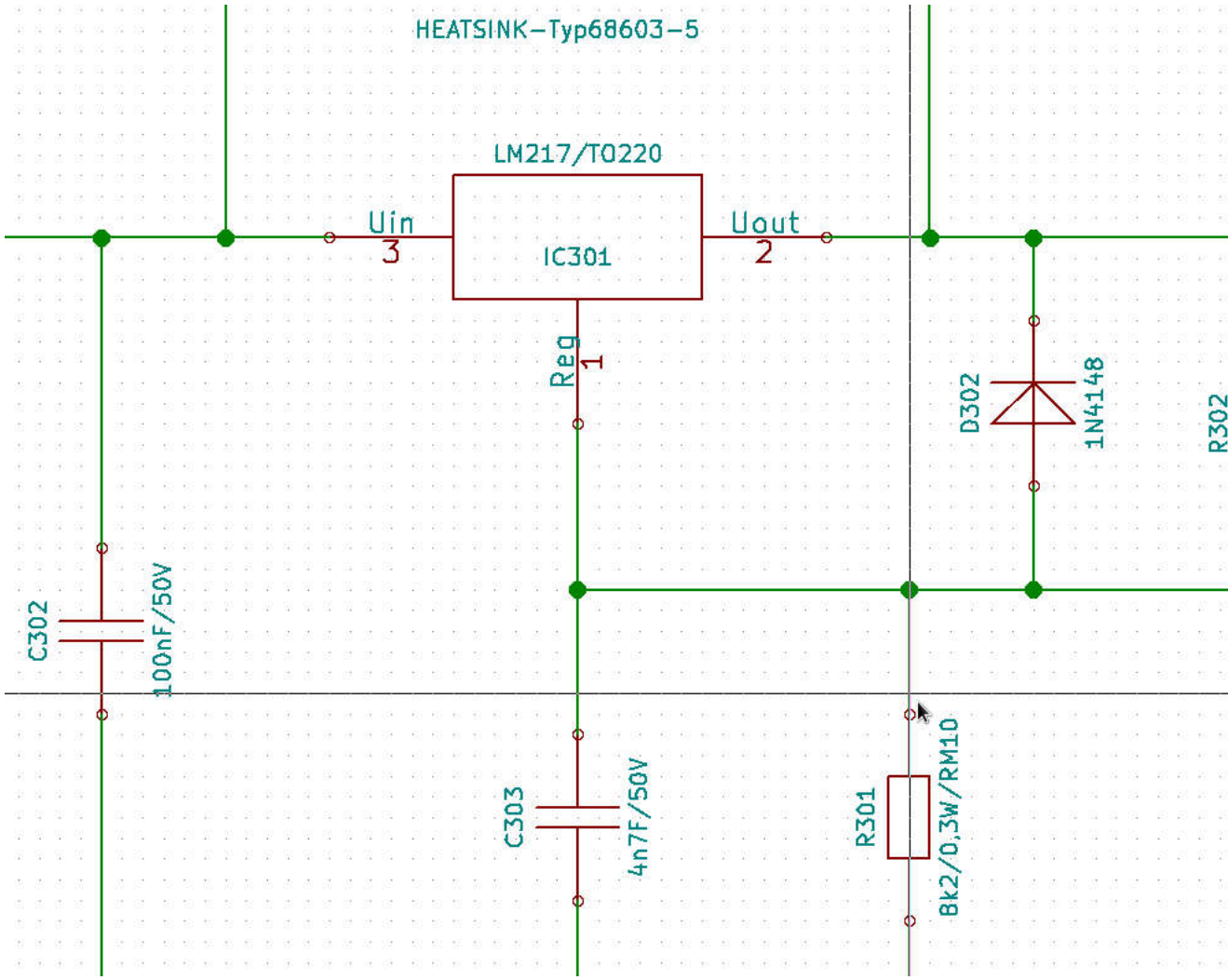

Figure 23: Device references at Sheet317Regler-2.

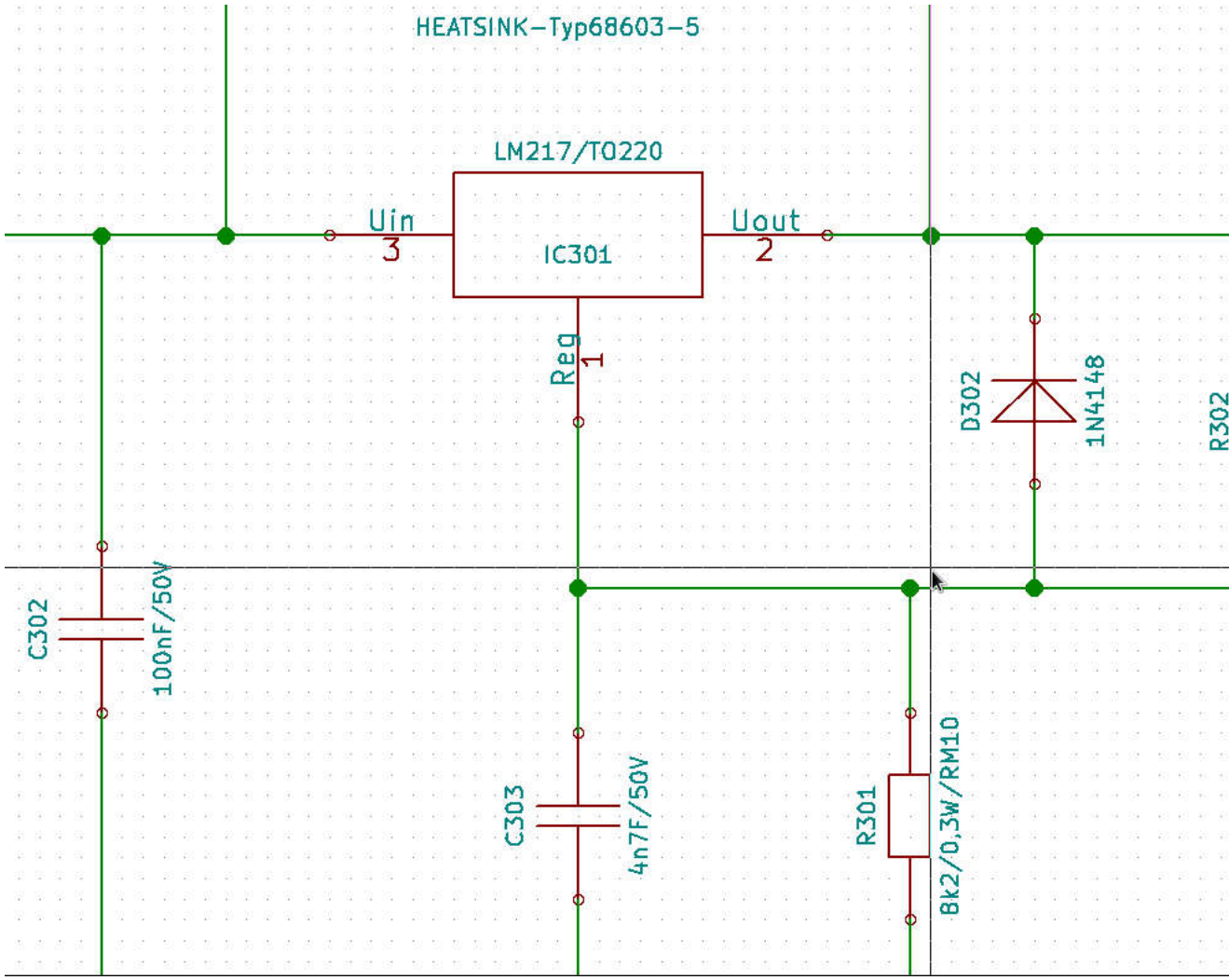

Figure 24: Device references at Sheet317Regler-3.

Figure 25: Button for creating a netlist

allocate footprints to the single device references. . At figure 28 you can see, that despite the forced same values different footprints can be allocated. This is, because the footprints does not reference to the schematic, but to the netlist. And for the netlist, different device reference numbers are different items.
Example: D201 and D301 are forced (because both reference to the same schematic "317Regler.sch") to the same value "1N4001", but due to the separation by the netlist, you can allocate different fottprints (THT and SMD) to this diodes.

table of contents

10.5 The effect at PCBnew

If you open PCBnew now, (figure 29) , you have to read the netlist first. . You can use the button like figure 30 . The default adjustments for reading the netlist into a new, empty boardfile shows figure 31. After reading the netlist, the footprints have to be drawn apart and placed. to get an overlook. You can do this by typing the "t" key. A window opens, where you can type the referencenumber of the device (as an example "R201"). After pressing <Enter>, you get the footprint at the mousepointer and can be placed with a left mouseclick. At figure 32 you see some of such devices.

Figure 26: Default adjustments for creating a netlist

Figure 27: Button for opening CVpcb

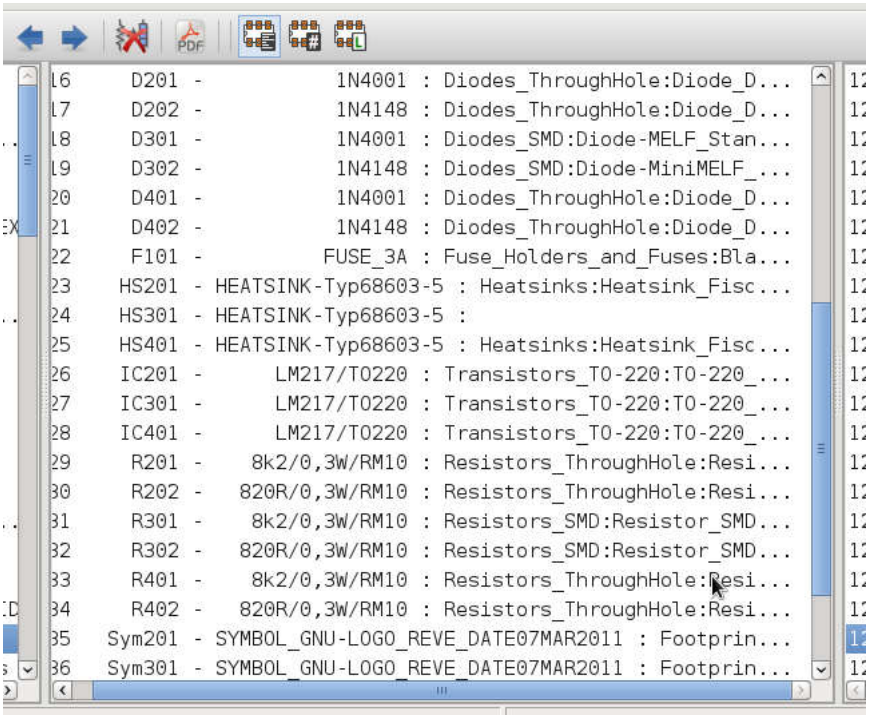

Figure 28: Different Footprints at CVpcb.

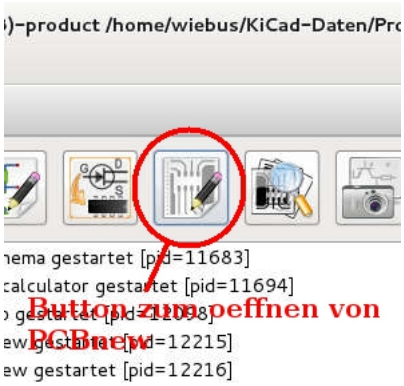

Figure 29: Button for opening PCBnew

Figure 30: Button for reading netlists

You see, that there are really different footprints. But the values are equal, because of the reasons i told you before. This values can be edited by clicking right to the value of a footprint. Here at "D301" the value is "1N4001" (Figure 32 at the lower right corner). A window pops up, where you can edit the value . As an example to "P600M". Se the effect at figure 33.
"D201" stays fix to "1N4001". But this method has severe drawbacks and should not be used. the drawback is the inconsistence between schematic and board (at the schematic, the value is yet "1N4007". So it is recommendet to change this value at the schematic, create a new netlist and reread this netlist into PCBnew, to get the new changed value into PCBnew.

table of contents

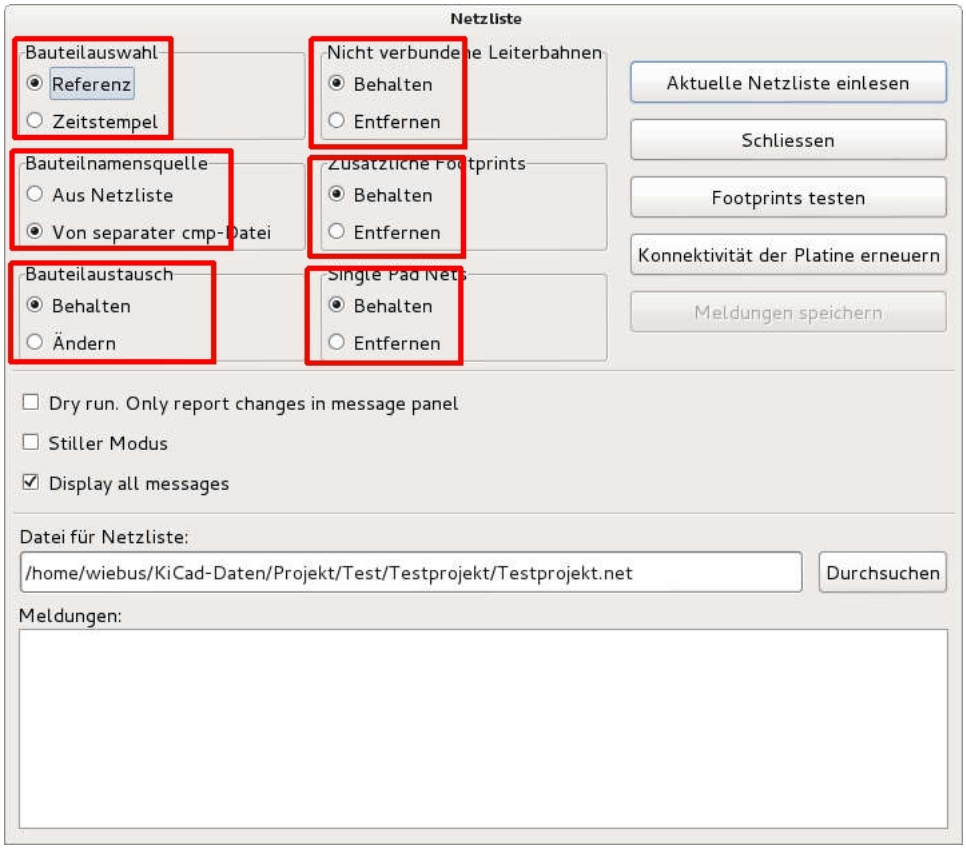

Figure 31: Default adjustments for reading a netlist into an empty board

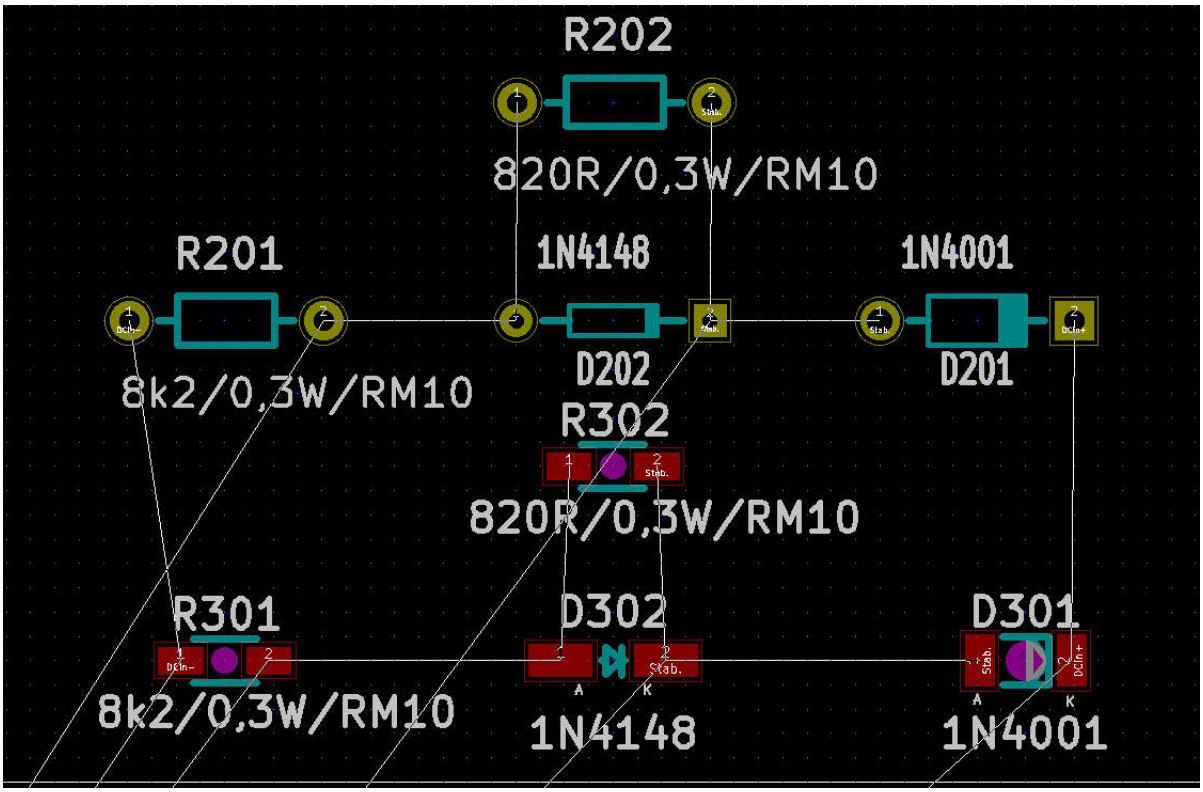

Figure 32: Different footprints with the same values

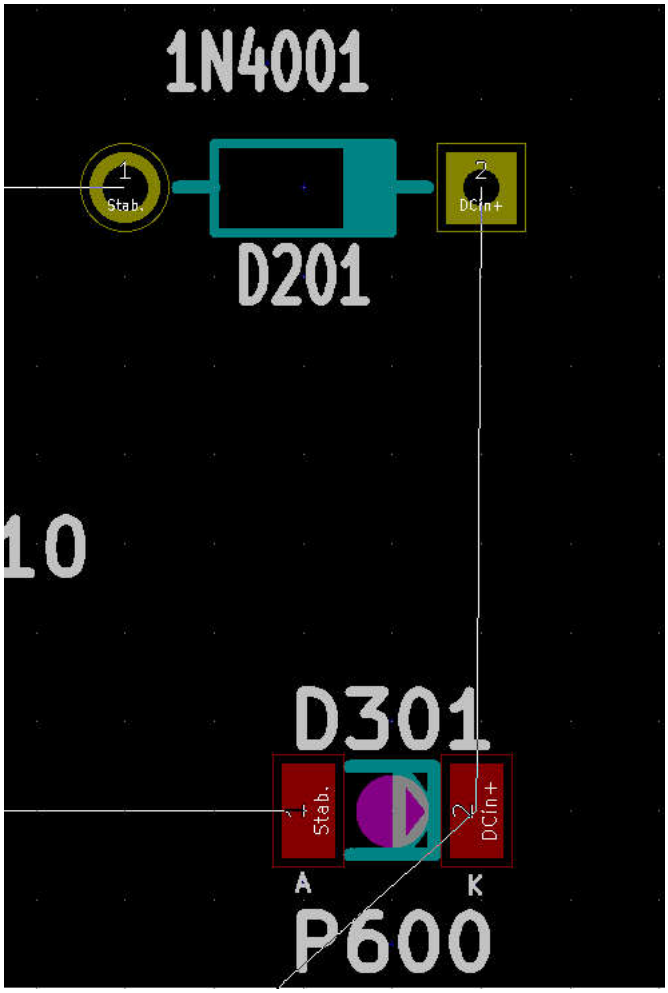

Figure 33: Changed values at D301

10.6 solving dependencies between the subschematics

For changing the values at a subschematic free, without altering and effecting the values at other subschematics, who references to the same schematic file, you have to create a separated subschematic. This can be done by copying and renaming the original subschematic. As an Example "317Regler.sch" to "317Regler-2.sch". After this, you can allocate "317Regler-2.sch" to the subschematic named ""Sheet317Regler-2". for this, make a right mouseclick to "Sheet317Regler-2" and choose editing. Than change the schematic file to "317Regler-2.sch".

For ensuring that the changes are well done, save the whole schematic project, close Eeschema and reopen the schematic again.
Open the subschematic "Sheet317Regler-2". Now you can edit the values of "R301" to "10k/0,3W/RM10" or "R302" to "3k3/0,3W/RM10". If you now open the subschematic "Sheet317Regler-1", you will see, that the before also dependend altered values of "R201" and "R202" are unchanged.
The subschematic "Sheet317Regler-1" references to the still original schematic file "317Regler.sch", which coexists adjacent to its copy "317Regler-2.sch". The mutual dependency between the subschematics is gone, they can be edited separately.
But because "Sheet317Regler-3" references further on to "317Regler.sch", the mutual dependency between "Sheet317Regler-1" and "Sheet314Regler-3" exists further on. but it could solved with the same method.

table of contents

11 Perspective - Using the method of the building blocks at PCBnew, too

Basically you can use this methods at PCBnew, too. You could use "appendboard" to quick compose new boards from prefabricated boards as buildingblocks. . But at moment, there is much more to do to achieve this, because the annotation has to matched manually, which is very time consuming.
Also because of the stronger dependency to the reality, a concrete layout has to be altered more at the detail. . As example slight turning and moving of footprints and pads and connecting them to the rest of the board.
I hope, to write about this topic in former revisions of this tutorial.

table of contents

12 Suggestions for improvements?

If you have any suggestions for improvements, so you are invited to tell me this. Please write me an e-mail to mailto:bernd.wiebus@gmx.de mitzuteilen.

13 Legal notice

Autor:
Dipl. Ing. Bernd Wiebus
Weezer Str. 5
47589 Uedem / Germany
Tel. +49-02825-9399977
Tel. +49-0162-6157950 (mobil)
e-mail: bernd.wiebus@gmx.de

table of contents

Index

table of contents