

CvPcb

Table of Contents

CvPcb 简介	3
CvPcb 特性	3
手动或自动关闭	3
后跟 Cvpcb	3
CvPcb 命令	4
主界面	4
主界面工具栏	4
主界面快捷方式	5
CvPcb 配置	6
封装管理	7
重要提示:	7
封装表	7
使用封装列表添加方向	12
查看当前封装	16
封装命令	16
查看当前 3D 模型	19
使用 CvPcb 向元件分配封装	21
手动分配封装	21
封装列表	21
自关闭	25
Equivalence 文件	25
Equivalence 文件格式	25
自动元件分配封装	26

参考手册

版权

本文档版权所有 © 2010-2018，其贡献者如下所列。您可以根据 GNU 通用公共许可 (<https://www.gnu.org/licenses/gpl.html>)，版本 3 或更高版本，或知识共享署名许可 (<https://creativecommons.org/licenses/by/3.0/>)，版本 3.0 或更高版本的条款分发和/或修改它。

本文档中的所有商标都属于其合法所有者。

贡献者

Jean-Pierre Charras, Fabrizio Tappero, Wayne Stambaugh.

翻译人

Liu HanCheng <buaa_cnlhc@buaa.edu.cn>, 2018.

taotieren <admin@taotieren.com>, 2019, 2020, 2021.

Telegram 简体中文交流群: https://t.me/KiCad_zh_CN

反

将所有的 Bug, 建议以及新版本重定向于此:

- 关于 KiCad 文档: <https://gitlab.com/kicad/services/kicad-doc/issues>
- 关于 KiCad 软件: <https://gitlab.com/kicad/code/kicad/issues>
- 关于 KiCad 软件国际化: <https://gitlab.com/kicad/code/kicad-i18n/issues>

行日期及软件版本

布于 2015-05-22。

CvPcb 简介

CvPcb 能原理中的元器件与行 PCB 布局的封装分配关系。二者的关系将被添加入由原理构建程序 Eeschema 建的网列表文件中。

当在元器件的封装字段初始化后, 由 Eeschema 生成的网列表文件才会包含元器件 PCB 封装与原理端口的关系。

种情况下, 封装和原理之的关系是用在原理通置元件的封装字段建的。此外封装也可能被定义于原理符号中, 在用从中加元器件, 其封装会被自置。

CvPcb 提供了在建原理的过程中元器件分配 PCB 封装的便方法。它有封装列表, 封装以及 3D 模型功能。些功能旨在提高分配封装的准确率。

用手元器件分配的封装。通建 .equ 文件, 也可以封装的自分配。 .equ 文件包含了元器件和其封装的相关信息。

我认使用种交互式的封装分配方法, 比起直接在制原理的候行封装分配, 更加, 并且有更高的正确率。

使用 CvPcb, 你可以看到所有可能可用的封装列表。此外, 你能在窗口中看不同封装的真几何外形, 可以帮助你原理中的元器件正确的封装。

CvPcb 只能通 Eeschema 后, 其入口位于 Eeschema 的部工具。无 Eeschema 是通 Kicad 的目管理器后, 是作独立件独立, 都可以通其部工具按 CvPcb。

从通 Kicad 目管理器后的 Eeschema CvPcb 通常来是更好的, 是因:

- CvPcb 需要取目配置文件, 以确定哪些封装需要被加
- 当目文件和打开的原理文件于同一目, Cvpcb 可以初始化元器件的封装置字段。

从通 Kicad 目管理器后的 Eeschema CvPcb, 可以确保上述要求自得到足。

WARNING

尽管用确能 从独立后的 Eeschema 中 Cvpcb, 但是注意, 独打开的原理文件由于可能缺失相关的目文件, 而致缺失相关的文件, 会使 Cvpcb 的工作出。如果在独打开的原理文件所在的目中, 不包括其他 fp-lib-table 文件, 那么工程封装列表也是不可用的。

CvPcb 特性

手或自关

Cvpcb 同支持交互式的手封装分配和通 .equ 文件行的自封装分配。

后 Cvpcb

Cvpcb 能从原理制程序 Eeschema 中后

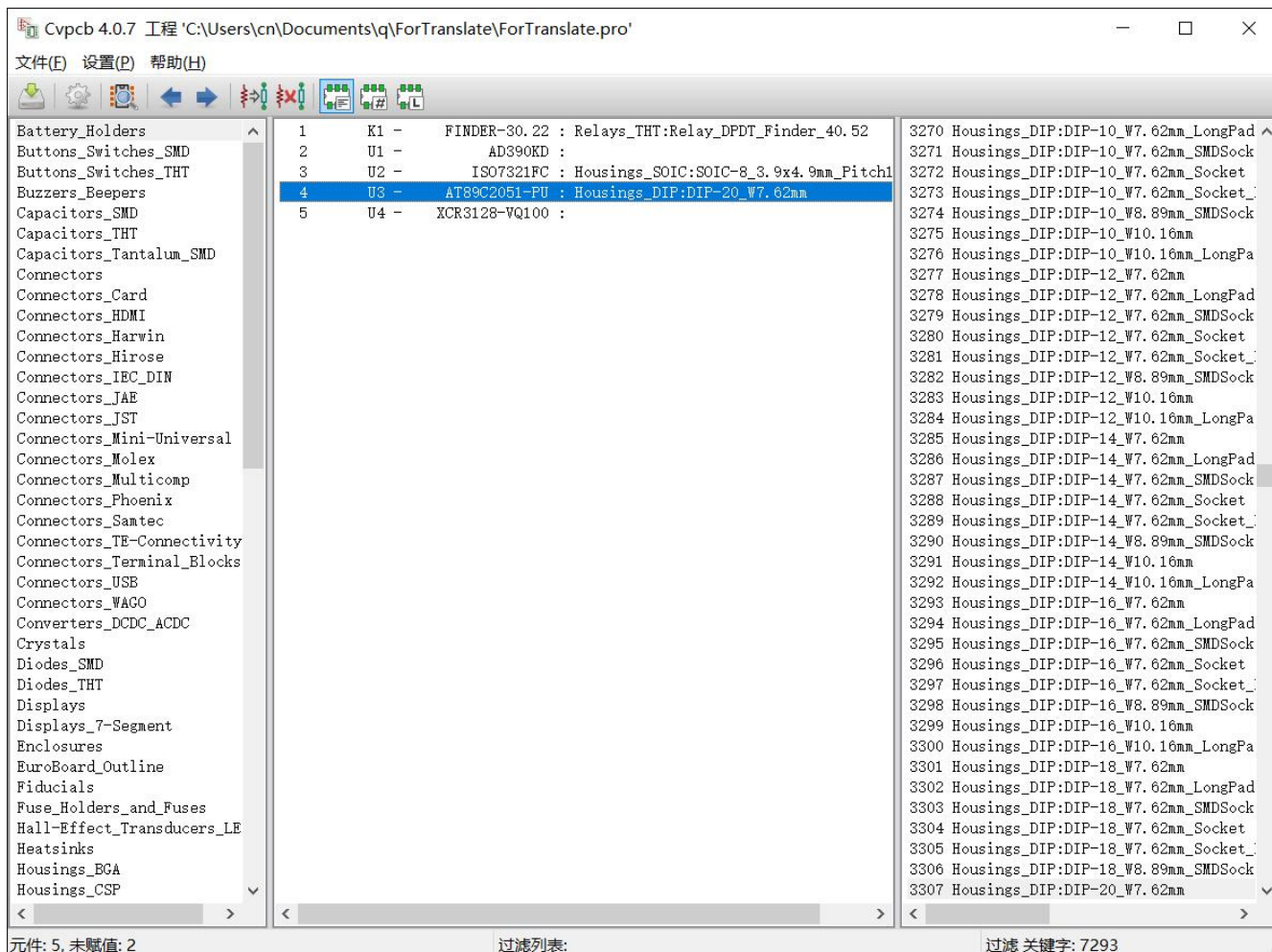


Eeschmea 会自动的将一些信息 (例如当前原理图中的元器件列表和可用的封装), 放入 CvPcb。在用 CvPcb 之前, 用唯一需要做的工作就是原理图中的各个元件号。

CvPcb 命令

主界面

下面的图片展示了 CvPcb 的主界面。



在主界面中, 左窗格包括了所有与项目相关的可用的封装文件名列表; 中间窗格包括了从网络文件中输入的所有元器件列表; 右窗格包括了所有从与项目相关的封装中加载的可用的封装列表。 如果没有加载网络文件, 那么元器件列表将是空的; 类似的, 如果没有找到可用的封装, 那么位于右窗格的封装列表也将是空的。

主界面工具



窗口的工具提供了下列命令的快速访问接口：

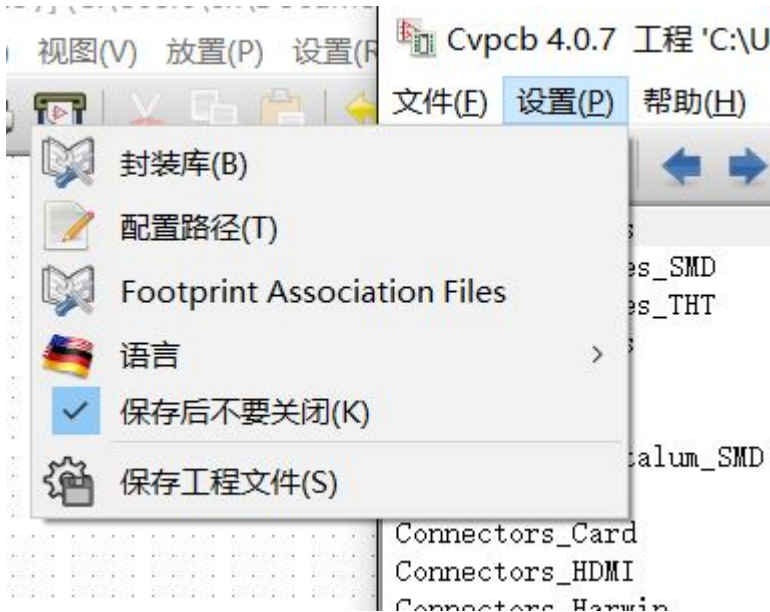
	将当前封装关移到 Eeschema是封装字段的内容）。
	用 CvPcb 配置菜
	示在封装中的元件的封装 窗口。
	在没有的情况下自列表中的上一个元件 封装
	自列表中的下一个元件而不占用封装
	使用等价文件自将封装与 元件相关
	除所有封装分配。
	使用默认打开定的封装文档 pdf 文件 pdf 看器。
	启用或禁用以限制封装列表 所元件的封装器。
	启用或禁用以限制使用的封装列表 所元件的引脚数。
	启用或禁用以使用。限制封装列表 定的

主界面快捷方式

下面的表格列出了 CvPcb 主窗口中的快捷方式：

右箭/卡	激活当前激活窗格右的下一个窗格。如果最后一个窗格当前已激活，行到第一个窗格。
左箭	激活当前激活的左的下一个窗格 窗格。如果第一个窗格当前已激活，行到最后一个窗格。
向上箭	当前所列表的上一个目。
向下箭	当前所列表的下一个目。
Page Up	当前所目的整 名
Page Down	当前所目的整 内容 名
Home	当前所列表的第一
End	当前所列表的最后一

CvPcb 配置



CvPcb 可以在保存封装关系后自动关闭或手动关闭。

点击菜单中的“设置”——“封装”将会打开封装配置对话框。

根据 CvPcb 版本的不同，存在两种封装管理方式：

- 旧的方式是使用 .mod 文件进行管理，用户可以看到封装文件列表。
- 新的管理方式使用“Pretty”格式。旧种管理方式将会使用一个文件列表，每个文件（文件名 *.pretty）就是一个新的管理方式允许用户使用来自 gEDA/gPCB 中的以及 Eagle 文件中 xml 格式的文件。

封装管理

重要提示:

本内容只与2013年12月之后行的KiCad 相关

封装表

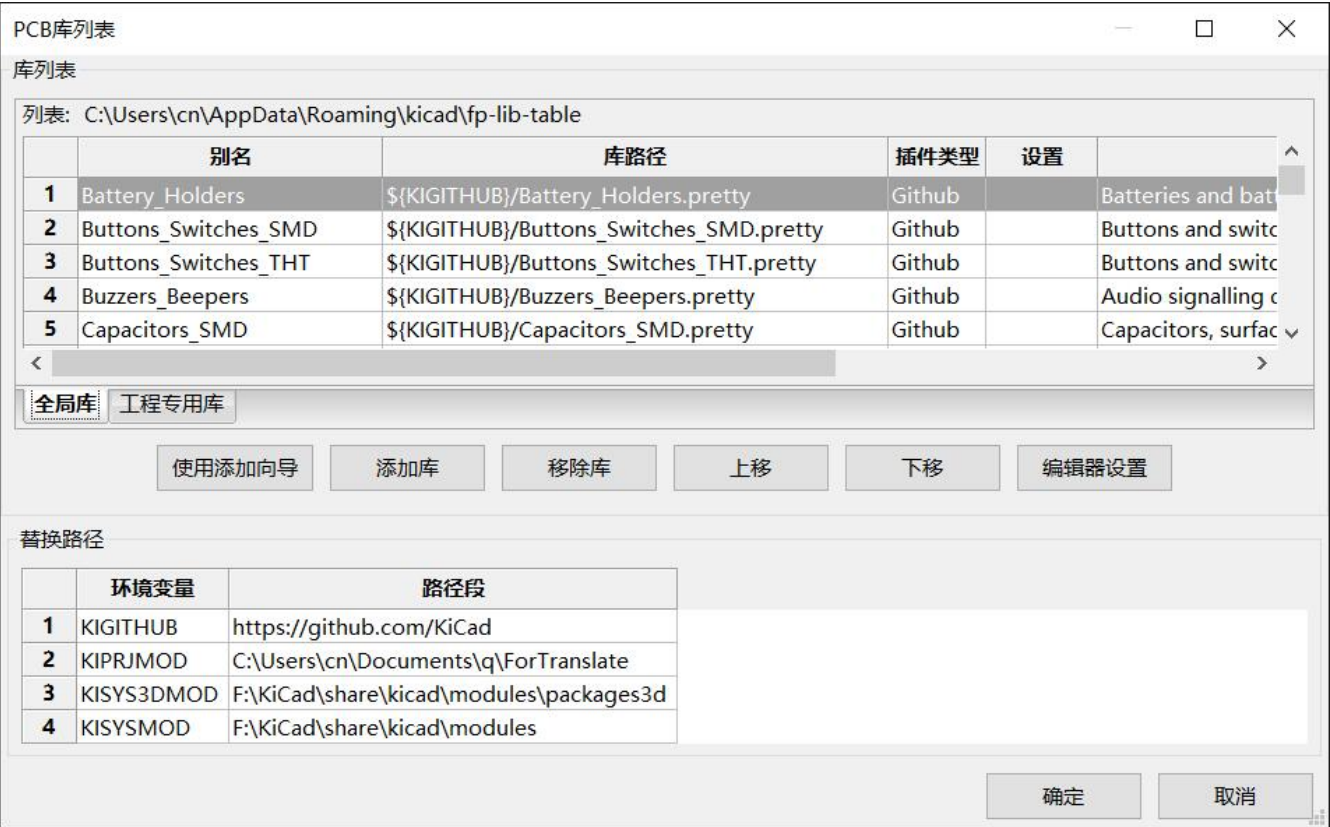
从2013年12月以后，Pcbnew 和 CvPcb 使用了新的基于 **封装列表** 的管理工具，新的管理工具允许用以下方式 **直接使用封装**

- KiCad 封装文件 (.mod 文件)
- KiCad 新式 .pretty 封装 (在用 的本地磁 中， 有 .pretty 展名并包含了 .kicad_mod 文件的文件)
- KiCad 新式 .pretty 封装 (托管在官方或第三方 的 Github 中)
- GEDA 封装 (包含了 .fp 文件的文件)
- Eagle 封装

NOTE

- 用 能改写位于本地磁 上的 Kicad .pretty 封装 (以及 些文件 中包括的 .kicad_mod 文件)。
- 其余的格式都是只 的。

下面的图片展示了封装列表对话框, 对话框可以通过菜单中的 “设置” - “封装” 打开。



封装列表的作用是 KiCad 支持的封装分配一个名称。可利用名称而不是之前基于封装路径找顺序的方法, 进行封装的查找。

该功能可使 Cvp pcb 控制不同封装的加载, 从而使得那些位于不同封装但是有相同名称的封装可以被正确加载。此外, 该功能能使 KiCad 来自其他 PCB 编辑器, 例如 Eagle 和 GEDA 的封装。

全局封装列表

全局封装列表包括了那些在任何项目中都可用的封装。表格的配置存在用主目下的 fp-lib-table 文件中。用主目的具体位置由用使用的操作系统决定。

项目封装列表

项目封装列表包括了在当前打开项目内能用的封装。项目封装列表能在项目的网络列表文件被加载后才能加载。如果当前没有打开任何项目, 或是在打开的项目中没有封装列表文件, 那么系统将建一个新的可供用的表格文件。

初始设置

首次运行 Pcbnew 或 Cvp pcb 如果在用主目下无法找到全局封装列表文件 **fp-lib-table**, 那么 Pcbnew 或 CvPcb 将把存在 KiCad 模板文件中的默认封装列表文件复制到用主目中。

如果默认的 fp-lib-table 文件无法被找到, 那么将会在用主目下建一个新的封装列表文件。种情况下, 用可以从复制 fp-lib-table 文件, 或是手动行封装配置。

默认的封装列表将作 kicad 的一部分而被安装, 其中包括了多标准的封装。

然, 用首先需要根据需求, 修改列表(添加/移除项目)。

(加多的封装会耗多)

添加列表

如果要使用一个封装，它必须先被添加到全局封装列表或工程封装列表中。当用当前有一个网列表文件的工程封装列表才是可用的。

封装列表中的项目名不可重复

“名”字段可以由用户自行决定，不必和封装的路径/封装文件名等相关。名中不可以出现冒号：。列表中的每一项需要有一个可用的路径。根据封装类型的不同，路径的具体表现形式可能不同。“路径”字段中的内容可以是绝对路径，相对路径，或者是环境变量(下文中会进一步讨论)

为了正确读取封装列表中每一项的“插件类型”字段必须被正确。目前 KiCad 支持的类型包括 KiCad legacy, KiCad Pretty, Eagle, 和 GEDA 封装。

列表中的“描述”字段，用于添加额外的注释信息。列表中的“置”字段在当前版本尚未使用，修改该字段没有任何效果。

- 注意，用无法在同一个封装列表中两项分配相同的名称。但是可以在全局封装列表和工程封装列表中使用相同的名称。
- 如果重名产生，工程封装列表中的名称将先被使用。定义在工程封装列表中的项目，将会被写入当前网列表所在目录下的 fp-lib-table 文件中。

环境变量替换

环境变量替换是封装列表的一大功能之一。它将允许用户使用环境变量定义自定义的封装路径。使用环境变量替换，需要在封装列表的“路径”字段中，遵守以下方法: `+${ENV_VAR_NAME}`

运行 KiCad 默认定义两个环境变量：

- **KIPRJMOD** 环境变量。该量指向当前项目的目录不可被更改
- **KISYSMOD** 环境变量。该量指向默认随 KiCad 安装的默认封装目录

用户可以通过在菜单中的“置” - “配置项目”中重新 KISYSMOD 的因此用户可以使用自定义的封装替换 Kicad 的默认封装。

如果当前项目的网列表文件已被加载，CvPcb 会将 KIPRJMOD 的置网列表文件的目录(即项目目录)。

在加载一个电路板文件 Pcbnew 也会置环境变量。

环境变量允许用户在不知道项目目录的情况下，将封装存储于项目目录下。

使用 GitHub 插件

GitHub 插件提供了只那些包含 Kicad pretty 封装文件的 GitHub 的接口。该插件也提供了 COW(“复制写入”)功能。该功能是可的它将允许从 GitHub 取的封装并且将它保存在本地。因而，“GitHub”插件用于只托管于 <https://github.com/> 的工程 pretty 封装。如果要向封装列表中添加 GitHub 其“路径”字段需要被置一个合法的 GitHub 地址。

例如:

https://github.com/liftoff-sr/pretty_footprints

或

<https://github.com/KiCad>

典型的 GitHub URL 有以下形式:

https://github.com/user_name/repo_name

“插件”型” 字段必被置“GitHub”。如果要使用 COW 功能,必向置字段内添加 允写个目的。用于置 GitHub 上封装的修改副本的存路径。存目中的封装将和 GitHub 上的只部分共同构成封装。如果没有声明,那么 GitHub 封装就完全是只的。如果声明了,那么“混合”封装的修改,将存到本地的 .pretty 文件中。注意“混合”封装中位于 GitHub 中的那一部分始是只的,意味着你无法直接除或修改特定 GitHub 中的内容。一步的讨认混合仍然属于“GitHub”型,只是它包括了本地的/写部分及程的只部分。

下面的表格展示了没有 允写个目 置的封装列表:

昵称	路径	插件型		描述。
github	https://github.com/liftoff-sr/pretty_footprints	Github		Liftoff's GH footprints

下面的表格展示了开启 COW 功能的封装列表。注意 境量 \${HOME} 做例用。github.pretty 目位于 \${HOME}/pretty/。如果用使用了 允写个目 提前手建上述目并以 .pretty 尾。

昵称	路径	插件型		描述。
github	https://github.com/liftoff-sr/pretty_footprints	Github	allow_pretty_writing_to_this_dir=\${HOME}/pretty/github.pretty	Liftoff's GH footprints

于置了 允写个目的 列表将首先加位于本地的封装。一旦用使用封装器封装行修改并保存在了 COW 本地文件中,那么 GitHub 中,和用修改并保存的的封装同名的任何封装的更新将不会被看。

始每一个 GitHub 封装使用不同的本地 .pretty 目。不要通多次引用同一目的方式来合两个不同的 GitHub 封装。

也不要不同的封装列表中使用相同的 COW (*.pretty) 目。将来不少。

置字段中, 允写个目的 置将会通使用 \${} 表示的境量来建路径, 就和在路径字段中使用境量一。

那 COW 的目的究竟是什么? 其它是为了更好的促封装的分享。

如果用周期性的将通 COW 修改的 pretty 元件反到 GitHub 的者, 那么用可以帮助更新 GitHub 中的封装。用可以将 COW 文件中的 *.kicad_mod 文件送到的者。当用得知他的修改被同步到了 GitHub 他就可以除本地的 COW 文件, 然后使用 GitHub 插件提供的只功能。用尽量向位于 <https://github.com> 的主多多提交, 让自己的 COW 文件尽可能的小。

使用模式

封装既可以在全局定义, 也可以在当前工程的范内定义。全局定义的封装将被存在用主目下的 fp-lib-table 文件中, 它将在所有的目中使用。

全局封装始终可以工作，即使当前没有加载任何模块。

工程封装只能在当前打开的网络列表文件中使用。

工程封装列表存储在模块文件内的 `fp-lib-table` 文件中。你可以自由地在哪个列表中定义封装

两种方法各有优劣。你可以在全局封装列表中定义所有你可能会用到的封装，可以随用随取。做的缺点是你必须在非常多的封装中找到你需要的封装。你也可以根据模块的需求定义你的封装

做的好在于你只需要定义你在某个模块中需要用到的封装，将大大减小搜索的复杂度。

做的缺点是，每新建一个模块，你都得重新手动定义每一个模块中需要用到的封装。你也可以同时在全局范和工程范内定义你的封装

一种使用模式是，将所有常用的封装定义在全局封装列表中，将一些只在特定模块中使用的封装定义在工程封装列表中。而言之，用完全可以自主决定封装的管理方式。

使用封装库列表添加向导

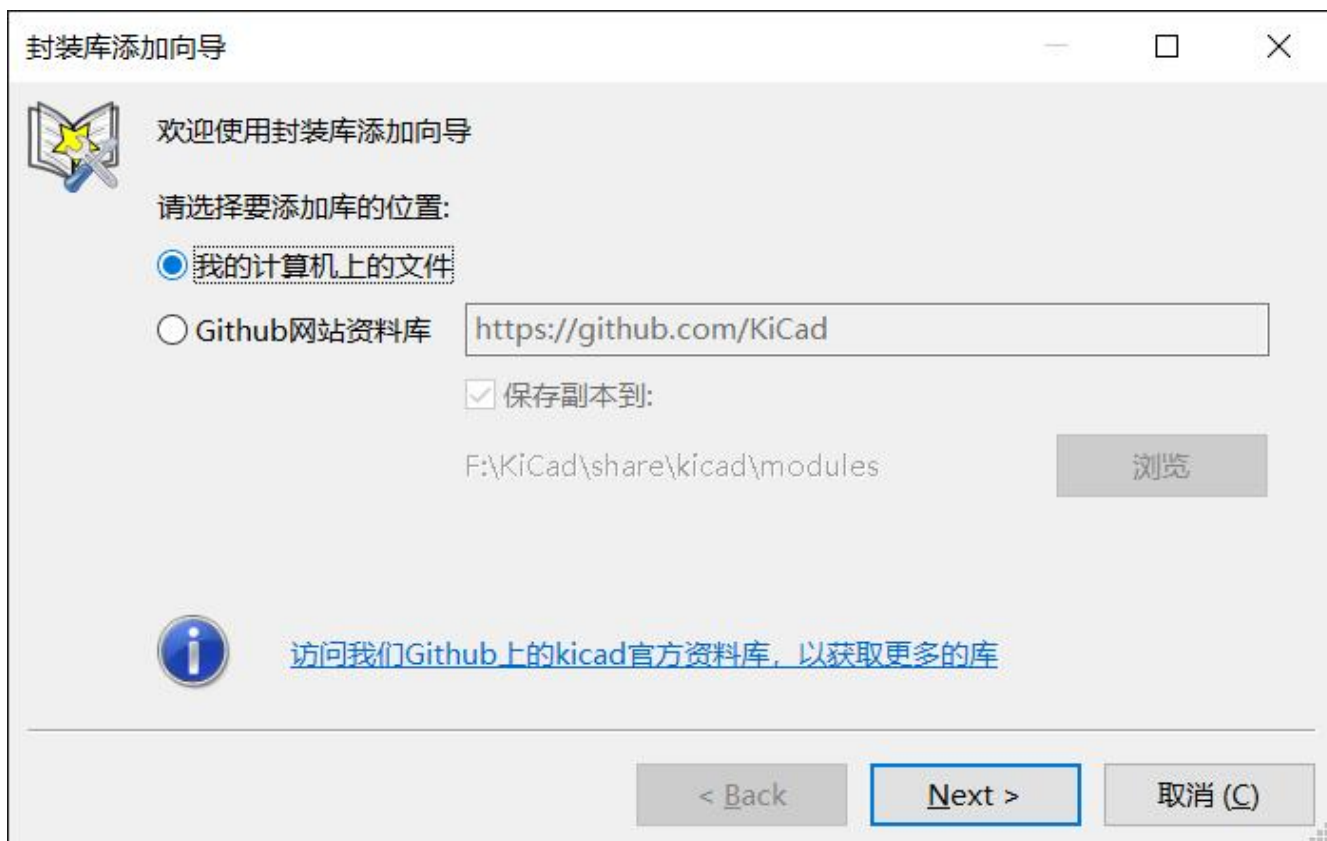
添加向导用于向封装库列表中添加封装库可以在 封装库列表 框中打开添加向导

待添加的封装库可以是任何 Kicad 支持的库型。

它可以是本地的封装库或是 GitHub 上的封装库

当库位于 GitHub 库中时它可以被添加工程也可以 将它下载到本地作本地添加。

此本地封装库默认被选中。



封装库添加向导

欢迎使用封装库添加向导


请选择要添加库的位置:

☒ 我的计算机上的文件

☐ Github网站资料库

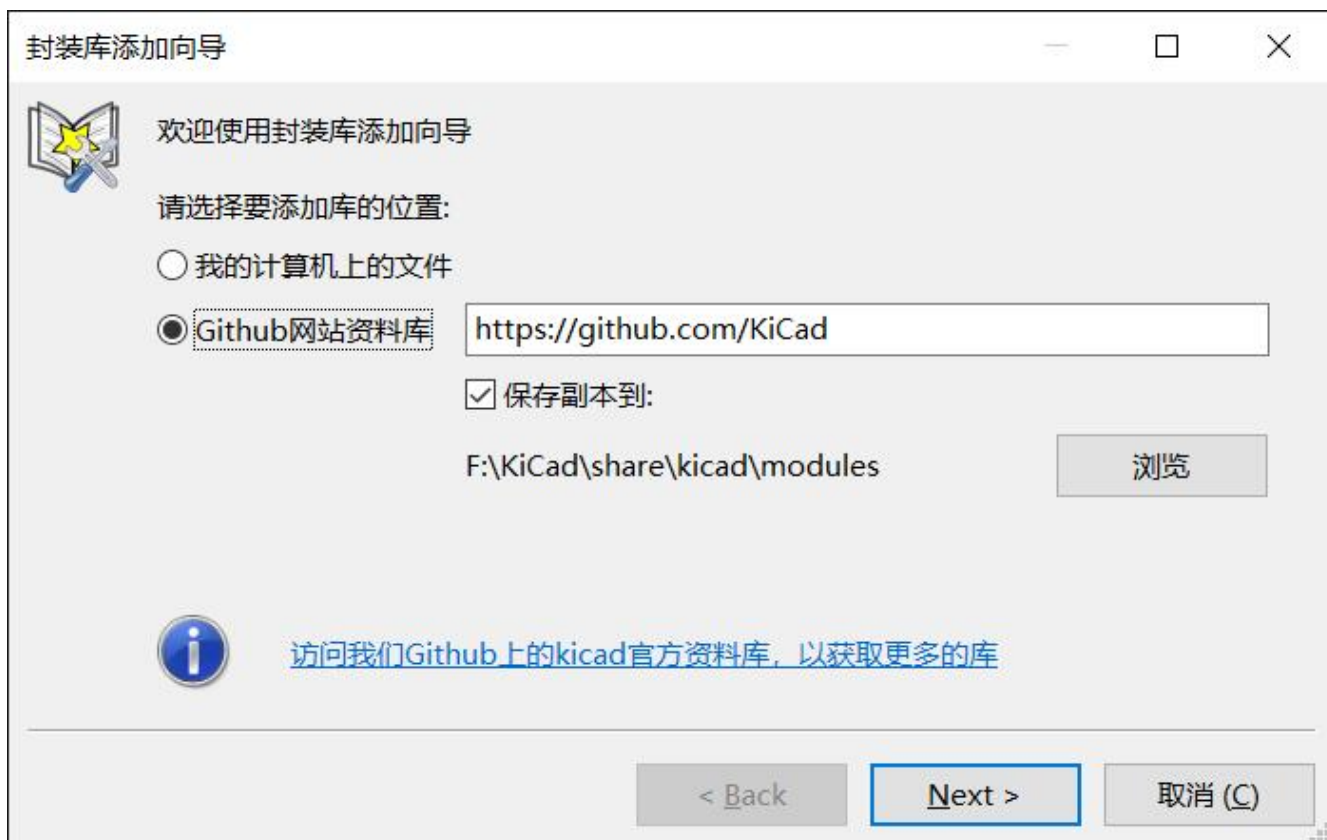
☒ 保存副本到:

F:\KiCad\share\kicad\modules

 [访问我们Github上的kicad官方资料库，以获取更多的库](#)

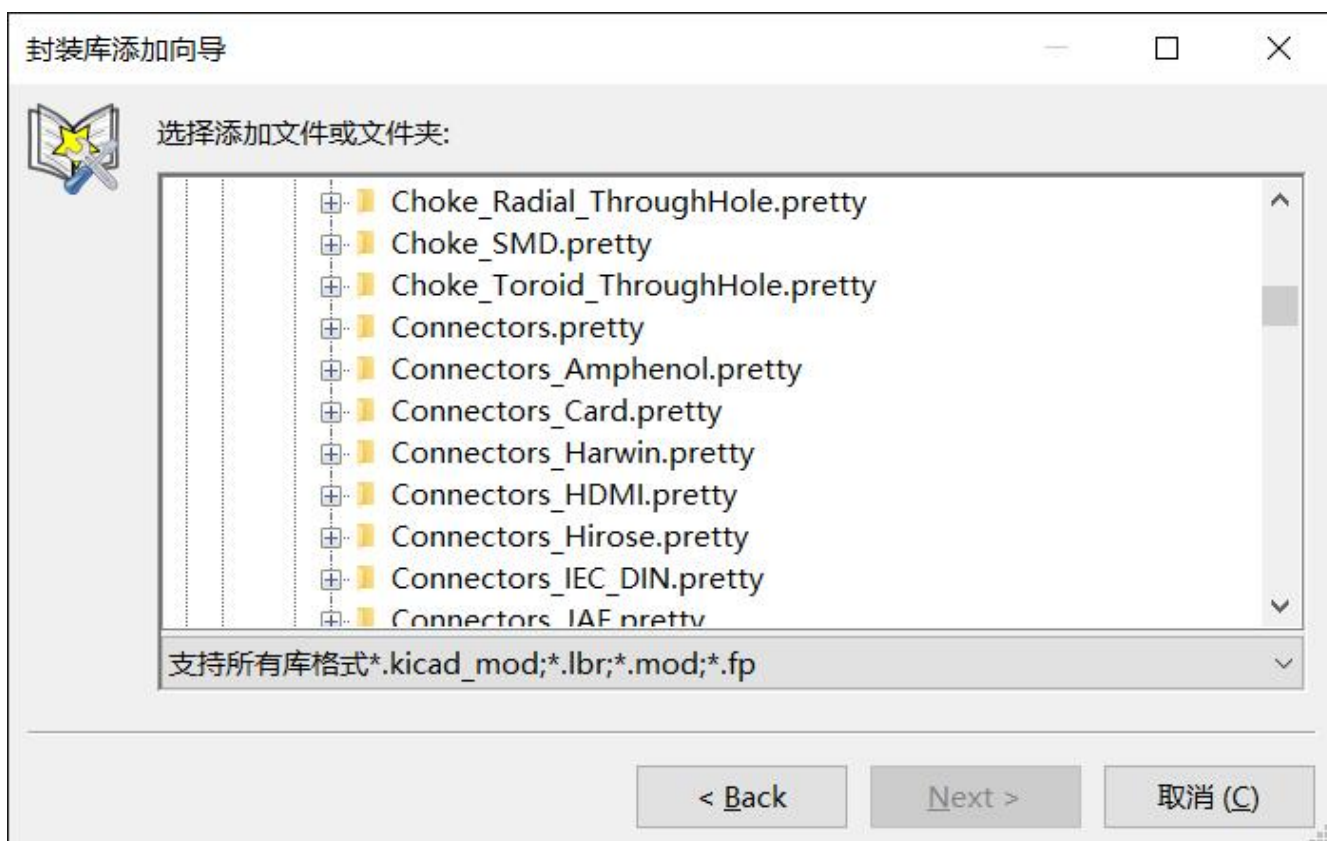
< Back Next > 取消 (C)

此工程封装库被选中。

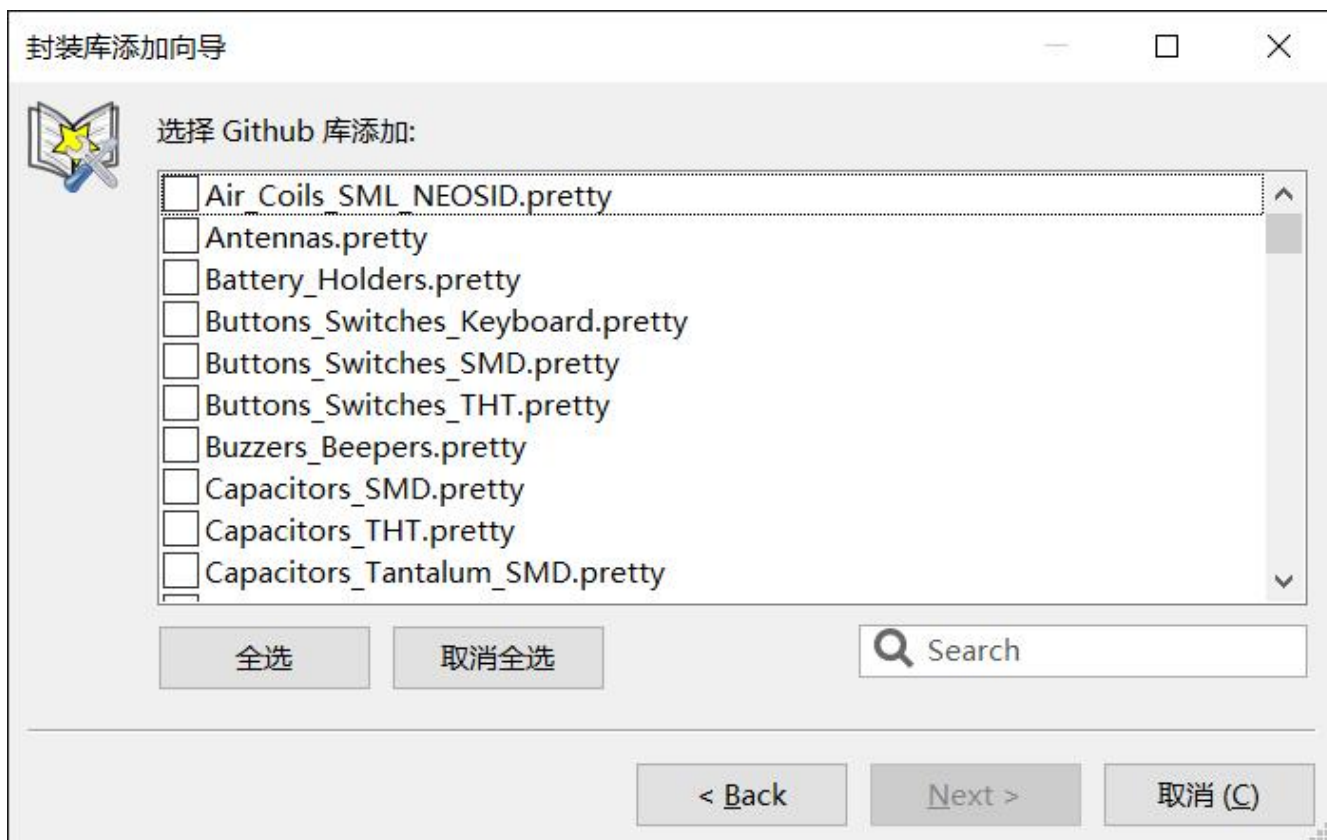


根据的不同, 将会显示些面中的一个,用于让用要添加的封装列表:

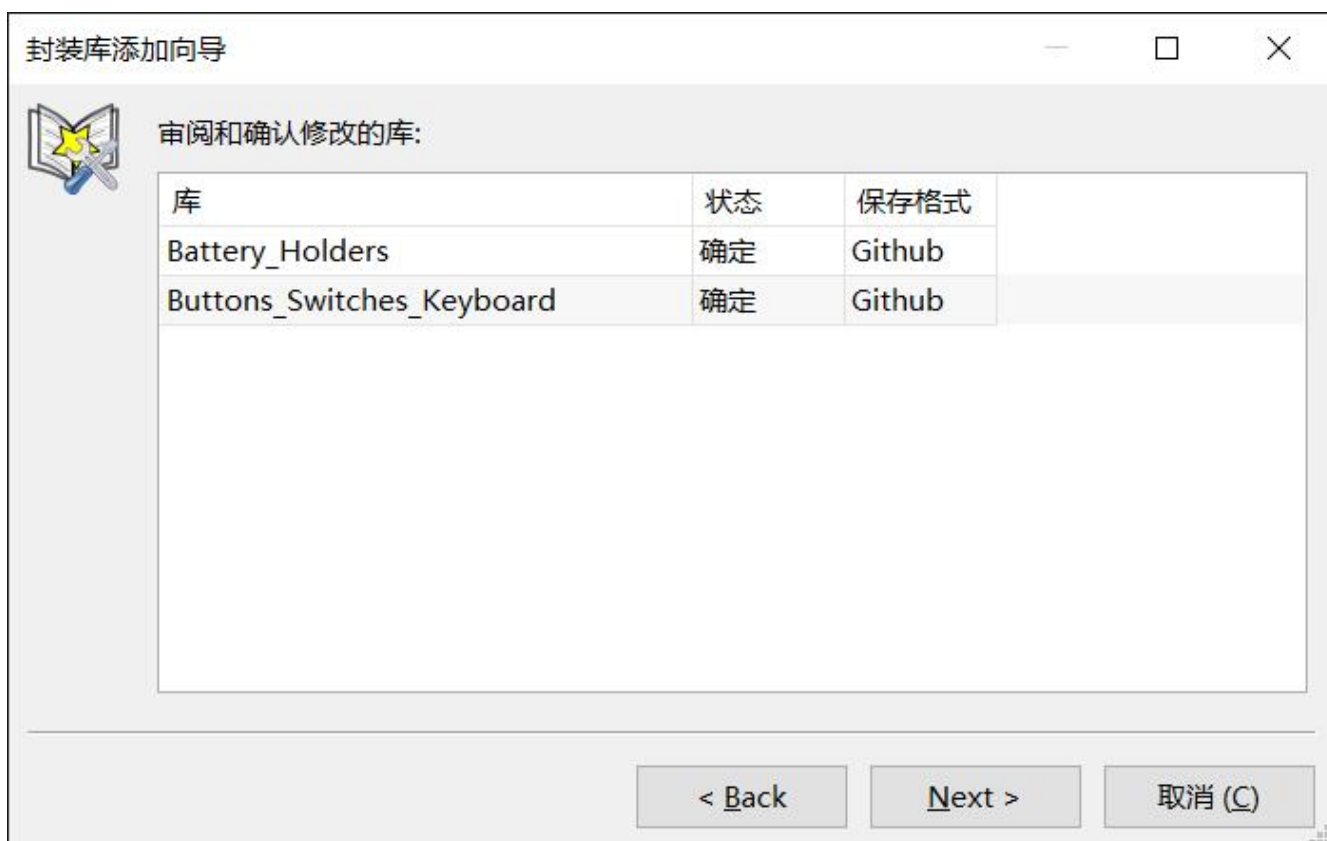
此本地被中.



此,程被中。



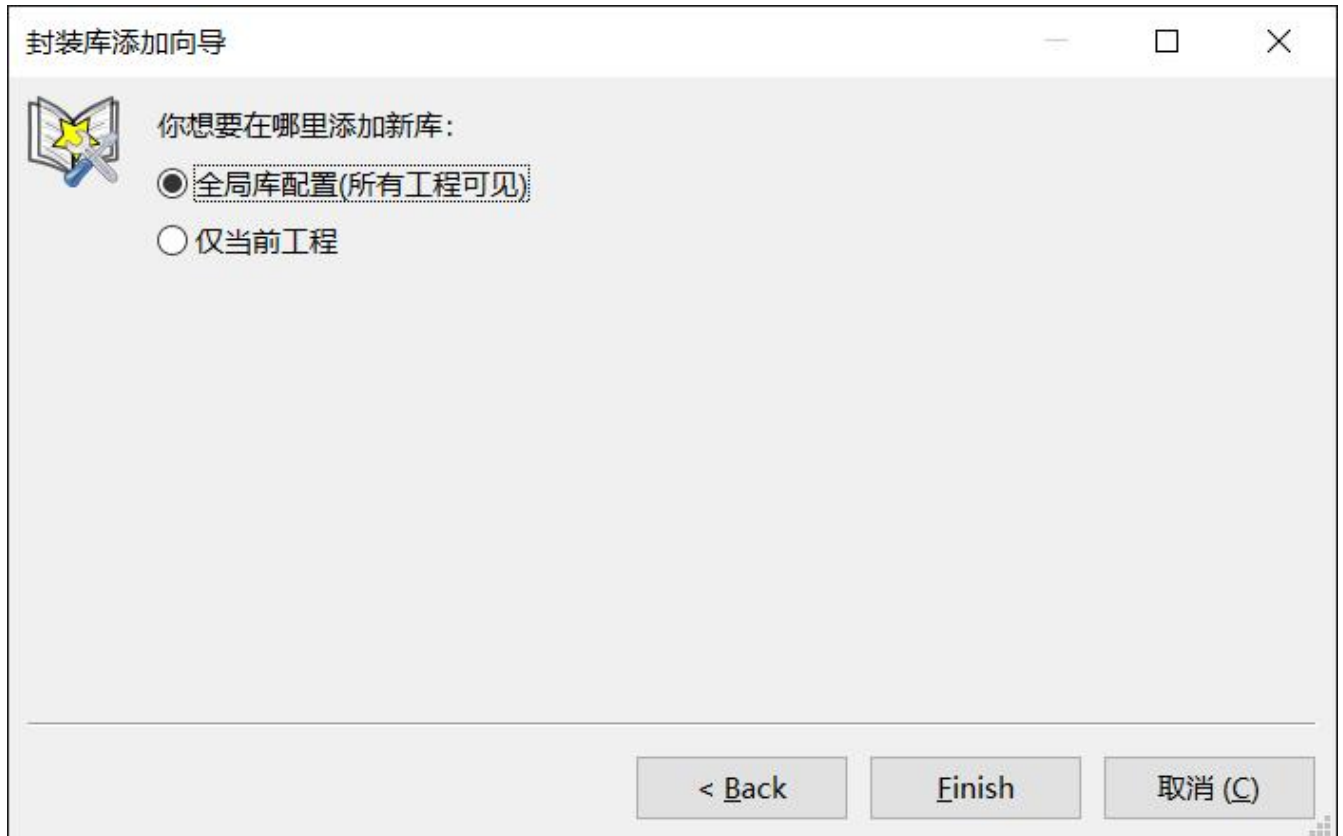
当一系列封装被选中后,下一个界面将显示:



如果一些库中的库不正确(不支持,不是封装库,...),它将被标记为“不可用”。

最后一个界面是需要输出的封装列表:

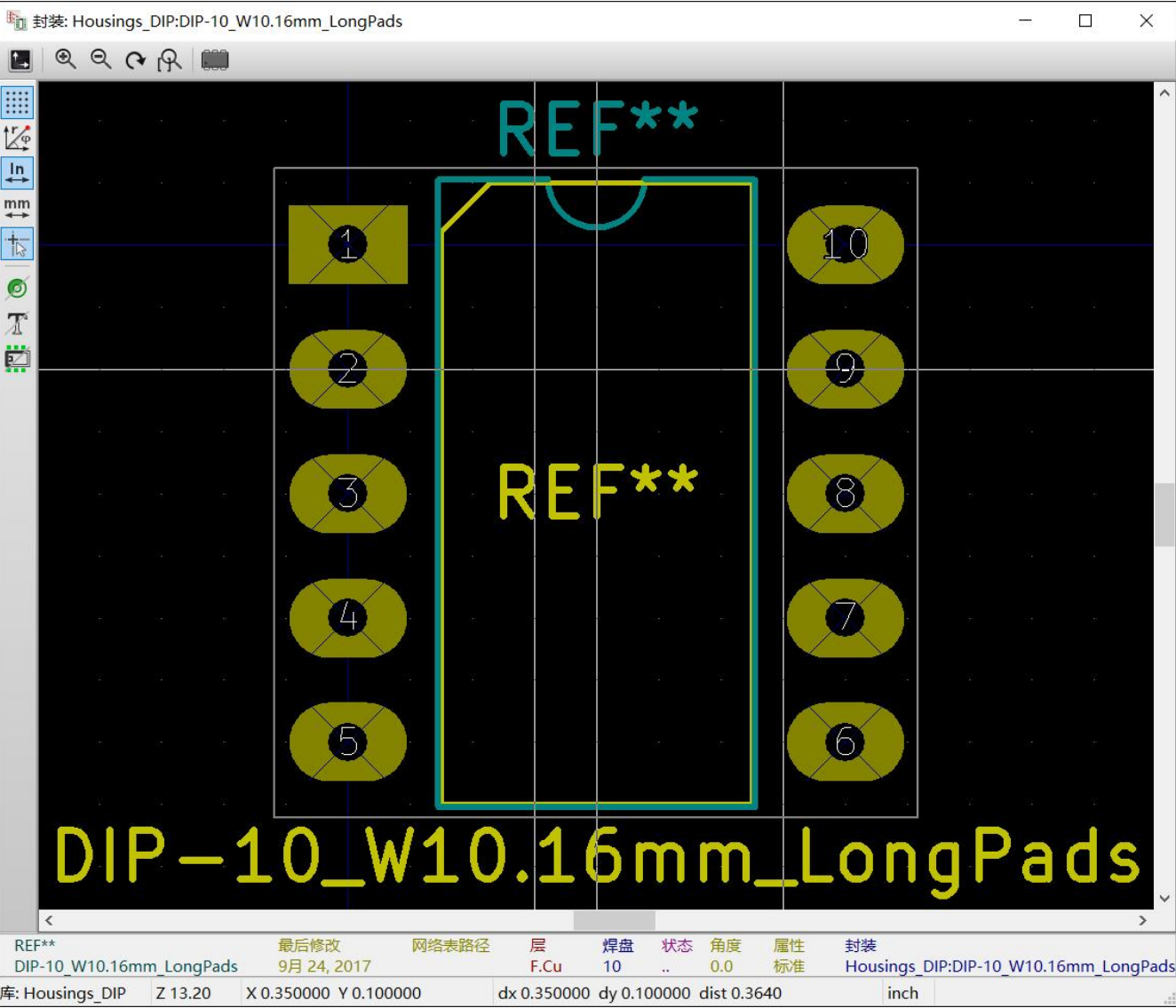
- 全局配置



看当前封装

封装命令

封装命令会将当前中的封装示在 封装 窗口中. 如果封装已被分配了一个 3D 模型, 那么三模型也会在窗口中示。下面的片展示了封装看器窗口。



状态信息

状态位于 CvPcb 主窗口的底部, 它用提供了多有用的信息。下面的表格列出了状态中不同窗格的定义。

左	元件计: 数量, 未分配封装的数量
中	中元件的列表
右	器状以及可用封装的数量

快捷

F1	放大
F2	缩小
F3	刷新显示
F4	将光标移到窗体中心
Home	是封装大小自适应于当前窗体
空格	置相于当前光标的相坐
右箭头	将光标向右移一个网格位
左箭头	将光标向左移一个网格位
上箭头	将光标向上移一个网格位
下箭头	将光标向下移一个网格位

鼠标操作

鼠标滚轮	在当前游标位置放大或缩小
Ctrl + 鼠标滚轮	水平方向平移
Shift + 鼠标滚轮	垂直方向平移
鼠标右键	打开右键菜单

右键菜单

通过点击鼠标右键打开右键菜单



□放□□	□□当前□放的倍数
网格□□	□□网格大小\

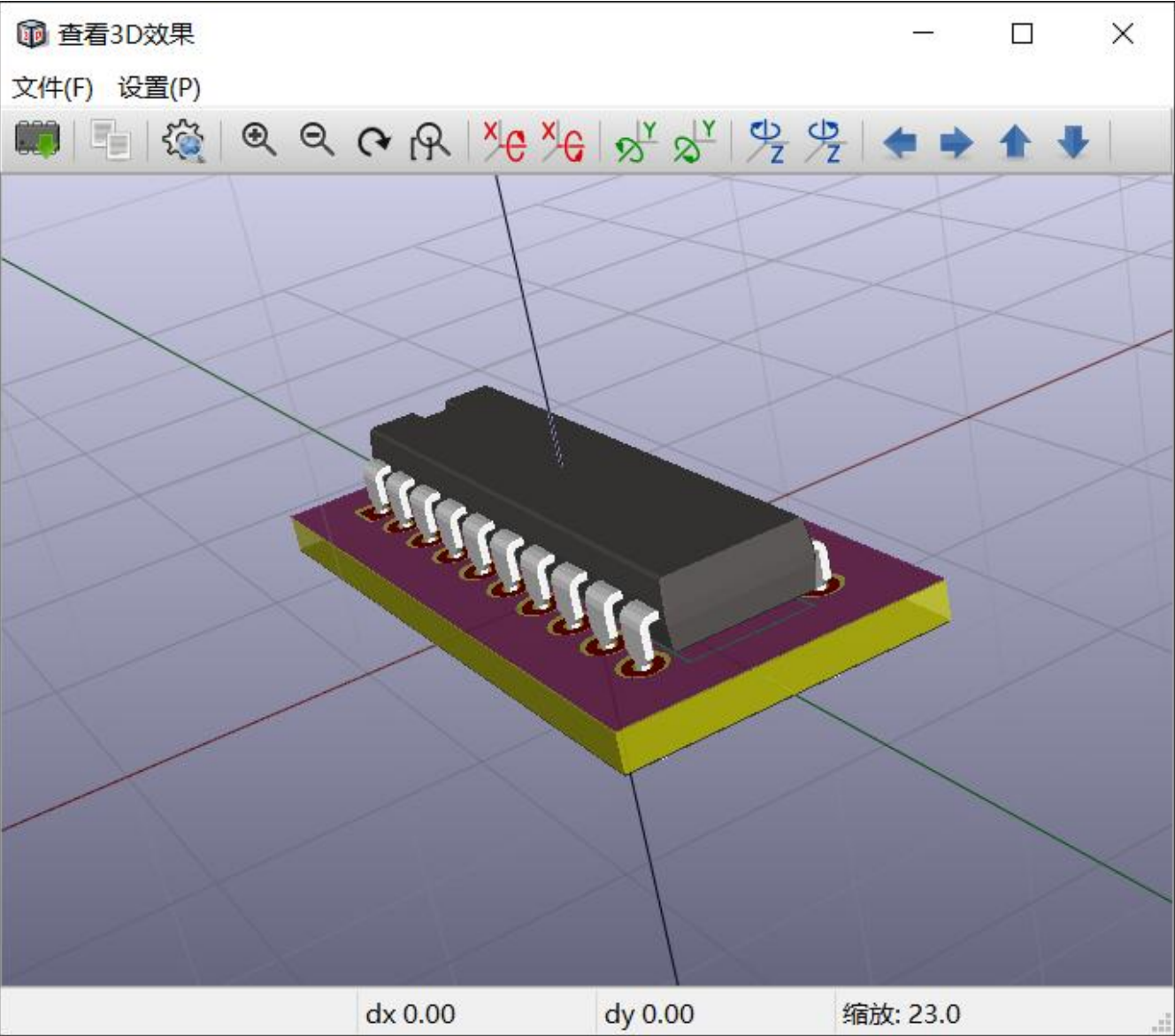
水平菜单□

	□示□示□□□□框
	放大
	□小
	重□
	在□示区域适合□□
	打开 3D 模型□看器

垂直工具□

	□示或□藏网格
	以极坐□或矩形表示法□示坐□
	以英寸□示坐□
	□示以毫米□□位的坐□
	切□指□□式
	在草□或正常模式下在□□板之□切□
	在草□或普通模式下□制文本之□切□
	在草□或正常模式下□制□□之□切□

看当前 3D 模型



鼠标操作

鼠标左键	在视图中放大或缩小
Ctrl+ 鼠标左键	水平方向平移
Shift + 鼠标左键	垂直方向平移

水平菜单

	重新加载 3D 模型
	将 3D 图像复制到剪贴板
	设置 3D 查看器
	放大
	缩小
	重置
	在显示区域适合
	沿 X 轴向后旋转
	沿 X 轴向前旋转
	沿 Y 轴向后旋转
	沿 Y 轴向前旋转
	沿 Z 轴向后旋转
	沿 Z 轴向前旋转
	向左
	向右
	向上
	向下
	打开和关闭正交投影模式

使用 CvPcb 向元件分配封装

手动分配封装

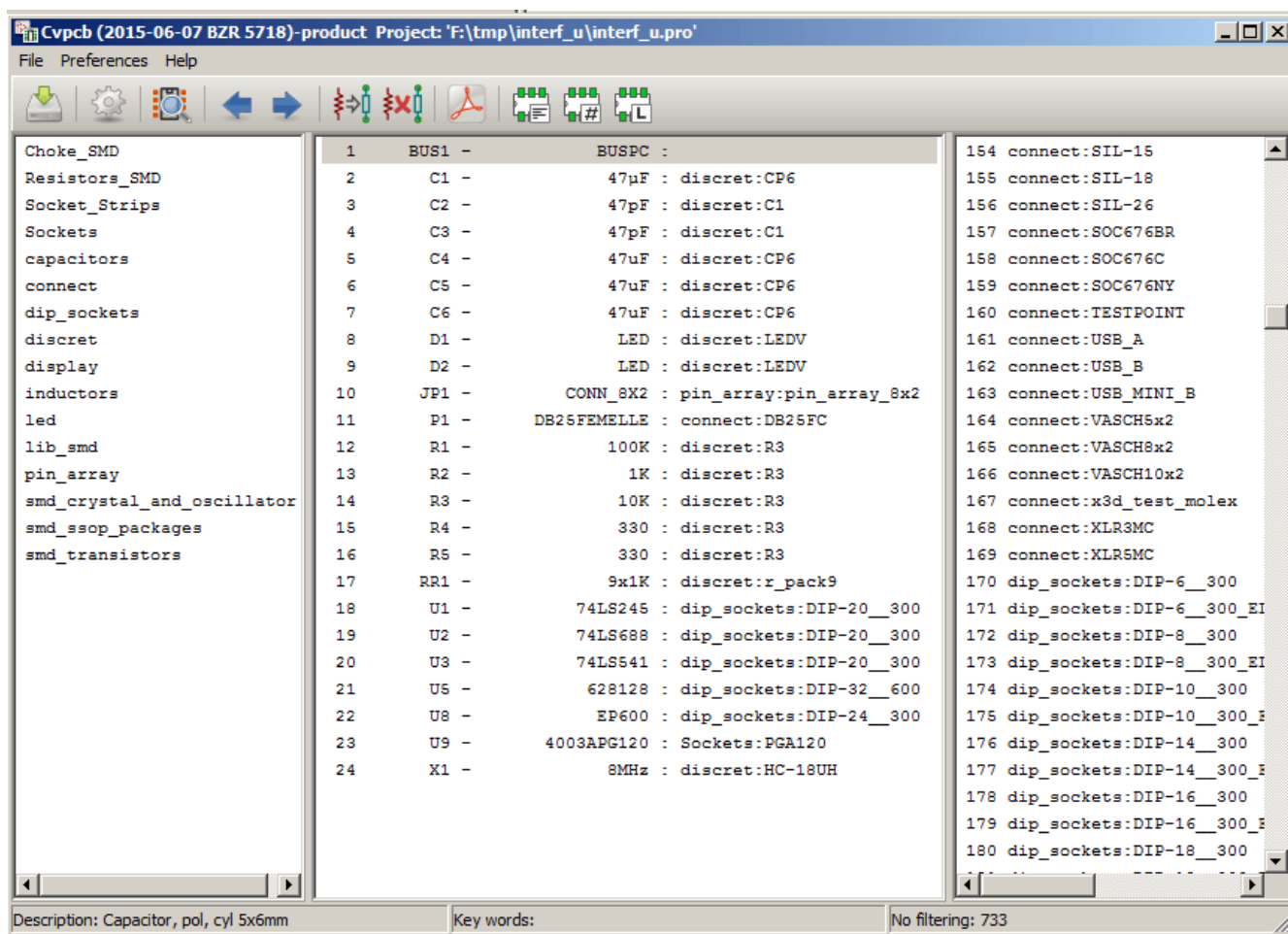
如果要手动分配封装, 首先在需要位于窗口中部的元件窗格中选择一个元器件. 然后在右侧的封装窗格中, 鼠标左键双击想要分配的封装. 然后封装就会被分配到该元器件. 此外, 分配完成后, 下一个未分配封装的元件将会被自动选中. 更改元件的封装, 操作类似.

封装列表

如果高亮表中某元器件/封装, 有一个或多个过滤器已打开, 那么, 右侧的封装窗格将自动显示该后的封装列表.

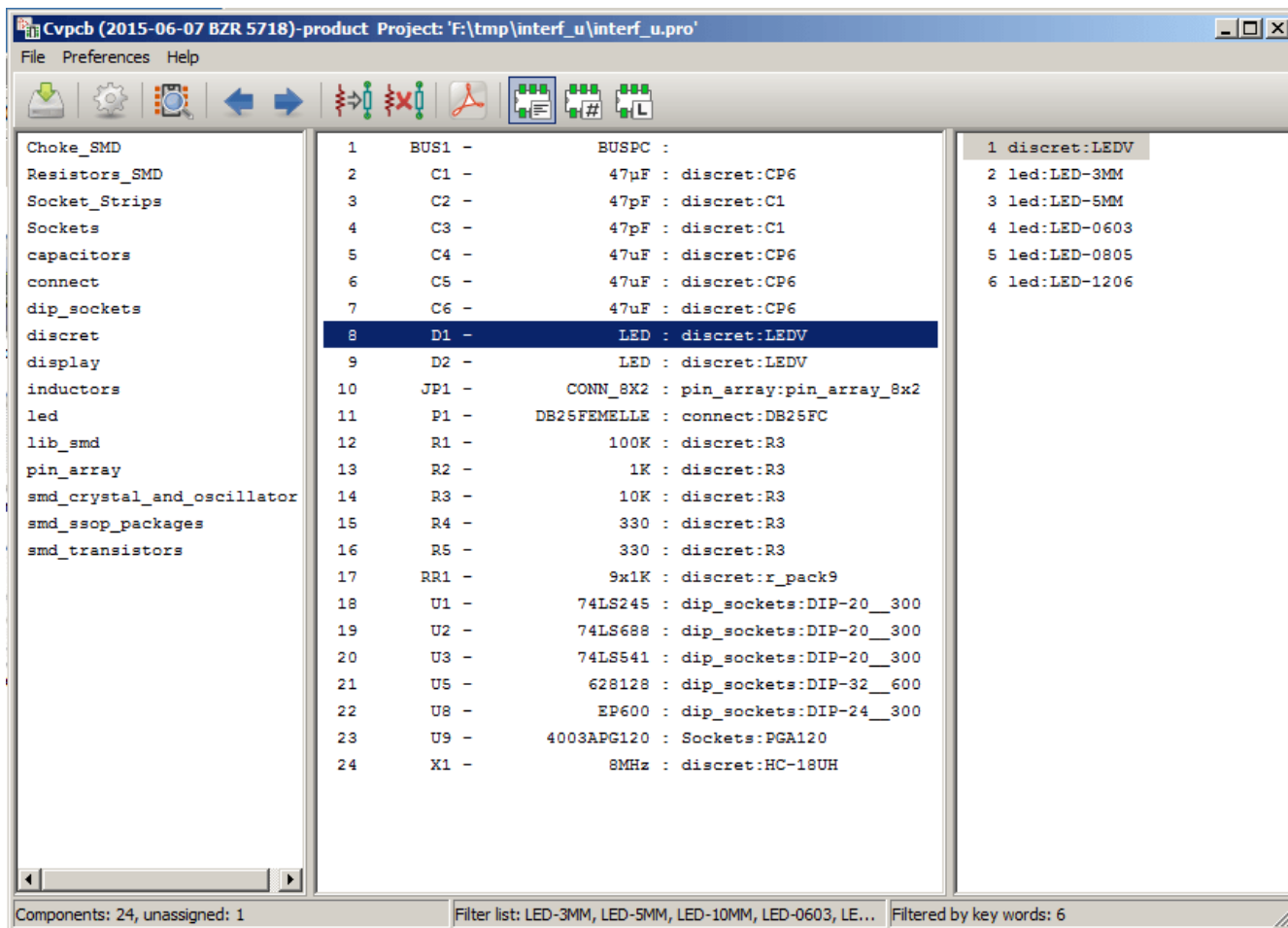
点击某些按钮将打开或关闭过滤器. 当所有的过滤器都处于关闭状态, 将会在右侧显示所有可用的封装.

关闭过滤器:

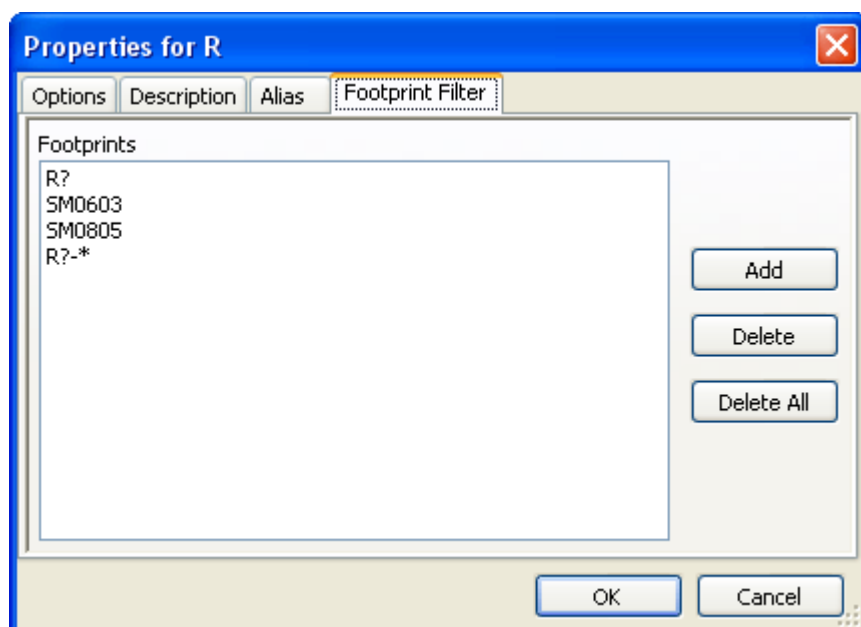


根据用于定义元件的过滤器封装列表行, 定义元件中启用的过滤器显示于主窗口底部的状态中部.

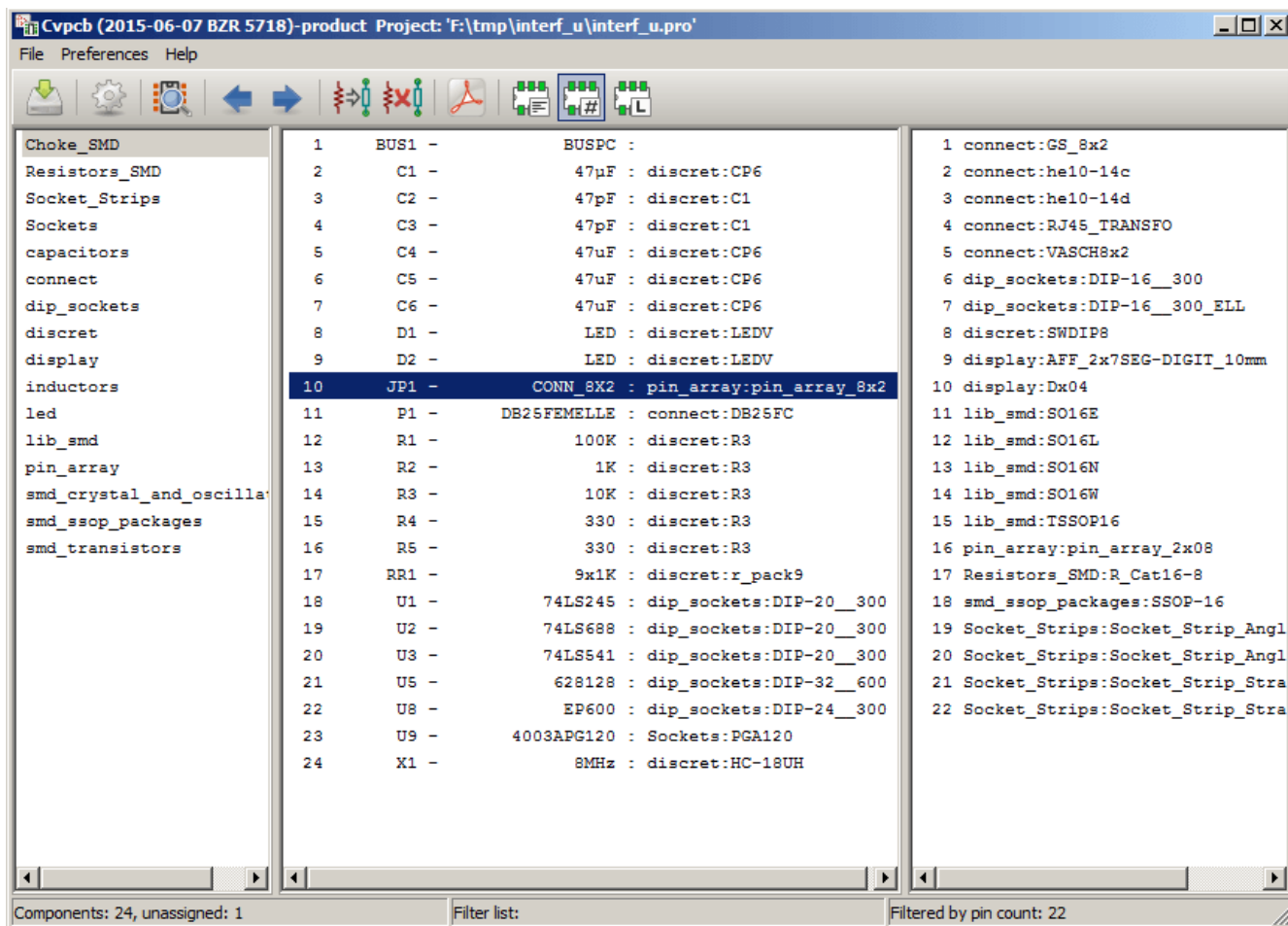
根据用于定义元件的过滤器封装列表行



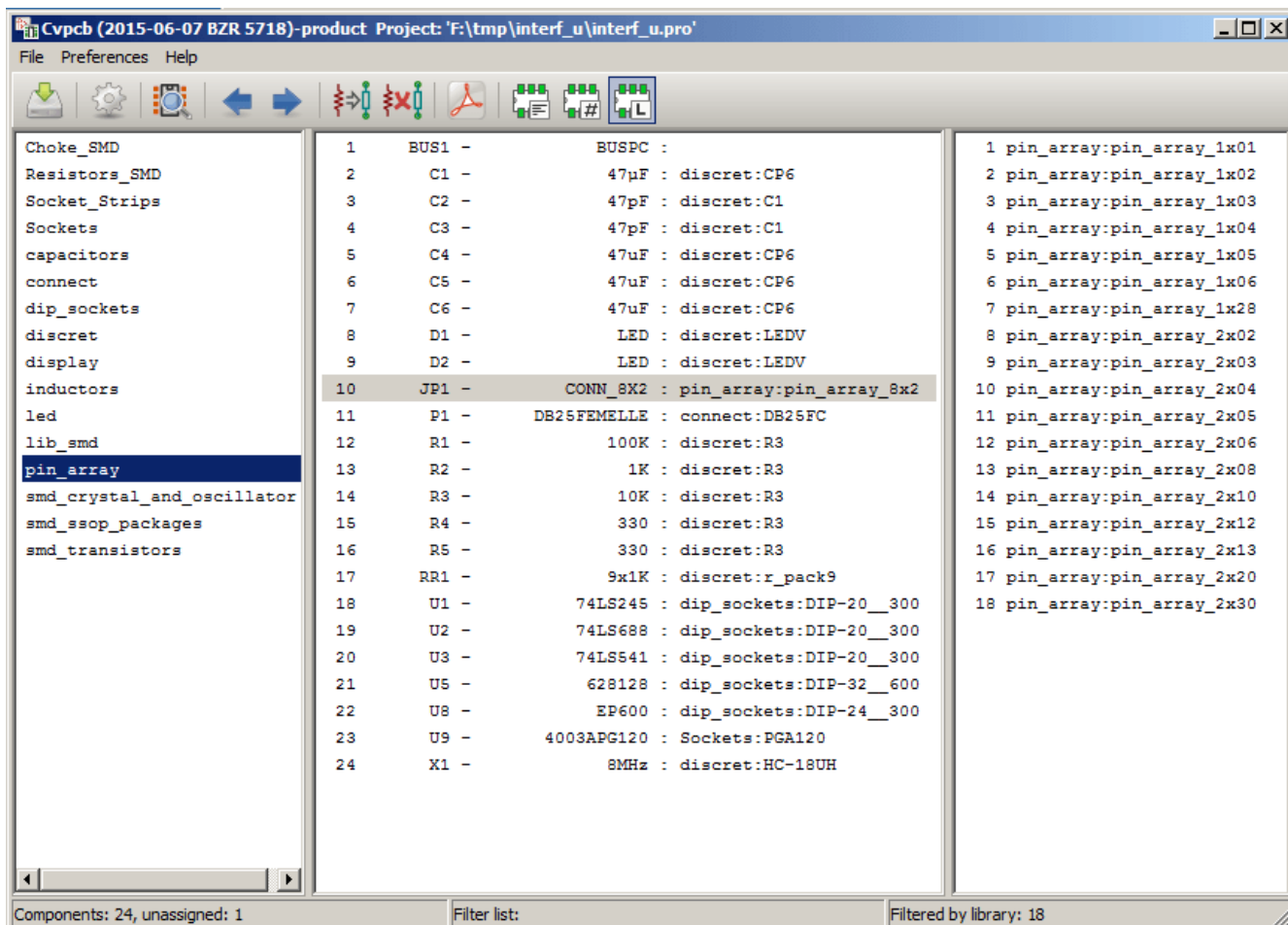
在 Eeschema 的元件编辑器中, 用可以在元件属性框的“封装”卡中设置元件的可用封装列表元件属性框如下所示。



根据中的元件的引脚数目行:

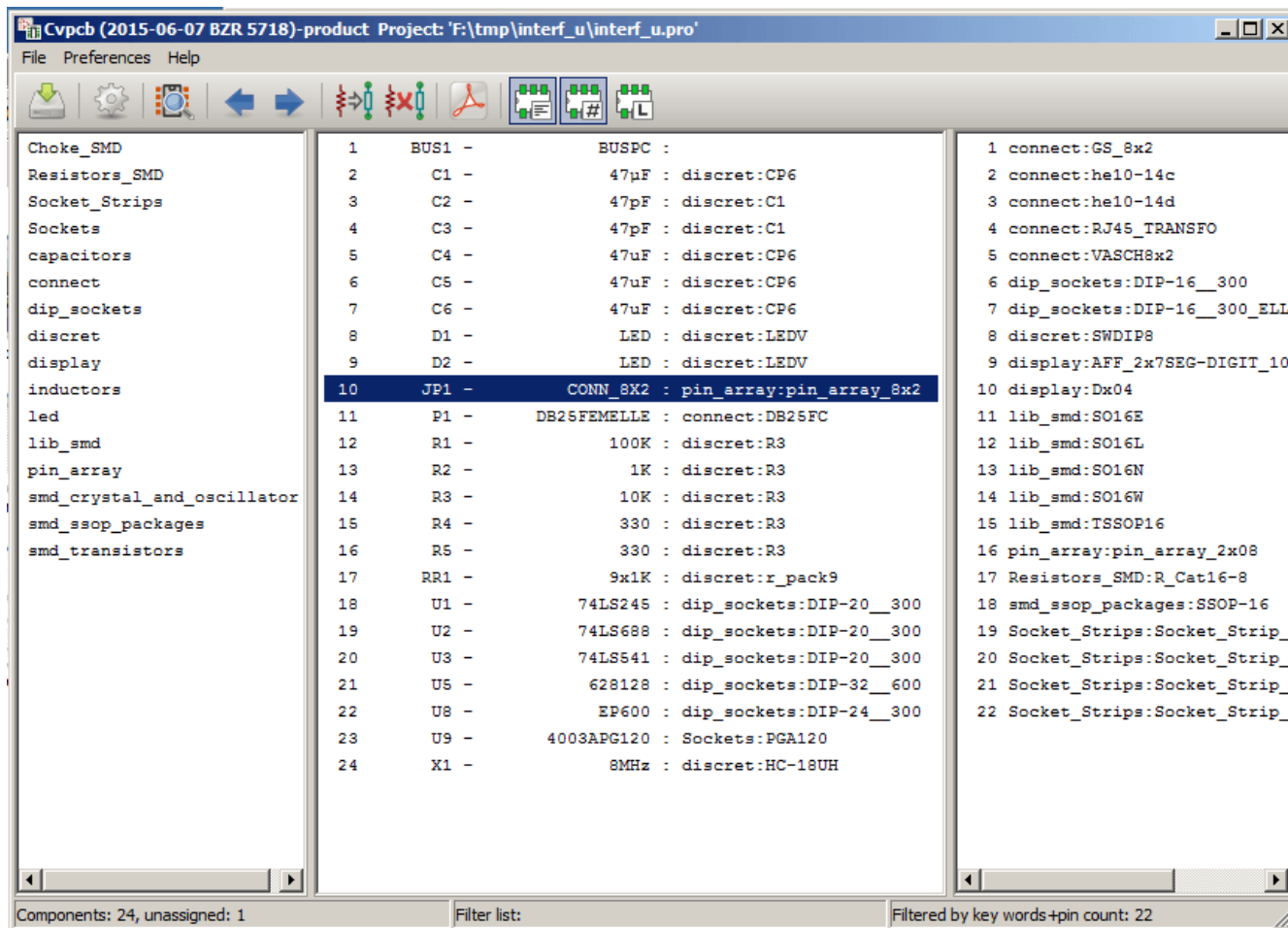


根据00的封装00行000



不同的器件可以叠加作用, 重复的器件需求可以帮助减少右侧窗格中的封装数目, 方便。

根据定义元器件的引脚和元件器件行:



自 关

Equivalence 文件

Equivalence 文件可以帮助用 自 元件分配封装.

它会根据元件的名称属性(*value field*)列出与之 的封装. Equivalence 文件的文件 展名 **.equ** 。

Equivalence 文件格式

equ 文件中每一行 一个元件. 每行的格式如下:

“元器件” “封装名”

每个名称都 被 引号括起, 不同的封装名 由一个或者多个空格隔开。

例子:

如果 U3 是 14011, 它的封装是 14DIP300,那么其 行 写:

“14011” “14DIP300”

开 的行 注 行.

Equivalence 文件示例:

```
#Integrierte Schaltkreise (SMD):
```

```
'74LV14' 'S014E'
```

```
'74HCT541M' 'S020L'
```

```
'EL7242C' 'S08E'
```

```
'DS1302N' 'S08E'
```

```
'XRC3064' 'VQFP44'
```

```
'LM324N' 'S014E'
```

```
'LT3430' 'SSOP17'
```

```
'LM358' 'S08E'
```

```
'LTC1878' 'MSOP8'
```

```
'24LC512I/SM' 'S08E'
```

```
'LM2903M' 'S08E'
```

```
'LT1129_S08' 'S08E'
```

```
'LT1129CS8-3.3' 'S08E'
```

```
'LT1129CS8' 'S08E'
```

```
'LM358M' 'S08E'
```

```
'TL7702BID' 'S08E'
```

```
'TL7702BCD' 'S08E'
```

```
'U2270B' 'S016E'
```

```
#Xilinx
```

```
'XC3S400PQ208' 'PQFP208'
```

```
'XCR3128-VQ100' 'VQFP100'
```

```
'XCF08P' 'BGA48'
```

```
#upro
```

```
'MCF5213-LQFP100' 'VQFP100'
```

```
#Spannungsregler
```

```
'LP2985LV' 'SOT23-5'
```

自□□元件分配封装

点□位于□部工具□的自□分配按□以解析 Equivalence 文件.

所有在 .equ 文件中能找到相关□□的元件,将会被自□分配封装。