



Pcbnew

November 6, 2020

Contents

1	Pcbnew 简介	1
1.1	描述	1
1.2	主要设计特色	1
1.3	一般建议	2
2	安装	3
2.1	安装软件	3
2.2	修改默认配置	3
2.3	管理封装库	3
2.3.1	全局封装库表	4
2.3.2	项目特定封装库表	4
2.3.3	初始配置	4
2.3.4	使用库管理器添加表条目	5
2.3.5	环境变量替代	6
2.3.6	使用库向导添加表条目	7
2.3.6.1	添加现有本地库	9
2.3.6.2	从 Github 添加库	10
2.3.7	使用 KiCad 插件	11
2.3.7.1	安装 KiCad 插件库	11
2.3.8	使用 GitHub 插件	11
2.3.8.1	写入时复制 (Copy-On-Write)	12
2.3.8.2	使用写入时复制 (Copy-On-Write) 共享封装	12
2.3.8.3	缓存 GitHub 请求	12
2.3.9	使用模式	13
2.3.9.1	修改 PCB 项目中的封装	13

3 一般操作	14
3.1 工具栏和命令	14
3.2 鼠标命令	15
3.2.1 基本命令	15
3.2.2 块上的操作	15
3.3 选择网格大小	15
3.4 调整缩放级别	16
3.5 显示光标坐标	16
3.6 键盘命令 - 热键	17
3.7 在块上操作	17
3.8 对话框中使用的单位	18
3.9 顶级菜单栏	18
3.9.1 “文件”菜单	18
3.9.2 编辑菜单	19
3.9.3 查看菜单	20
3.9.3.1 3D 查看器	21
3.9.4 设置菜单	22
3.9.5 放置菜单	23
3.9.6 布线菜单	23
3.9.7 检查菜单	23
3.9.8 工具菜单	24
3.9.9 设置菜单	24
3.9.10 帮助菜单	25
3.10 使用顶部工具栏上的图标	25
3.10.1 辅助工具栏	26
3.11 右侧工具栏	28
3.12 左侧工具栏	30
3.13 弹出窗口和快速编辑	30
3.14 可用模式	31
3.14.1 正常模式	31
3.14.2 封装模式	32
3.14.3 布线模式	36

4	原理图实施	38
4.1	将原理图链接到印刷电路板	38
4.2	创建印刷电路板的程序	38
4.3	更新印刷电路板的程序	39
4.4	读取网表文件 - 加载封装	39
4.4.1	对话框	39
4.4.2	可用选项	39
4.4.3	加载新的封装	40
5	层	43
5.1	简介	43
5.2	设置图层	43
5.3	层描述	44
5.3.1	铜层	44
5.3.2	配对技术层	45
5.3.3	独立技术层	46
5.3.4	一般用途的图层	46
5.4	选择活动层	46
5.4.1	使用图层管理器进行选择	46
5.4.2	使用上方工具栏选择	47
5.4.3	使用弹出窗口进行选择	48
5.5	选择过孔层	49
5.6	使用高对比度模式	50
5.6.1	高对比度模式下的铜层	50
5.6.2	技术层	51
6	创建和修改电路板	53
6.1	创建一个板	53
6.1.1	绘制板边框	53
6.1.2	使用 DXF 绘图作为电路板边框	54
6.1.2.1	准备 DXF 绘图以导入 KiCad	54
6.1.2.2	将 DXF 文件导入 KiCad	55
6.1.2.3	示例导入 DXF 形状	55

6.1.3	读取原理图生成的网表	56
6.2	校正一块板	58
6.2.1	要遵循的步骤	58
6.2.2	删除错误的布线	58
6.2.3	已删除的元件	59
6.2.4	修改后的封装	59
6.2.5	高级选项 - 使用时间戳选择	59
6.3	直接改变已经放置在板上的封装	60
7	封装放置	63
7.1	协助安置	63
7.2	手动放置	63
7.3	自动封装分布	65
7.4	自动放置封装	66
7.4.1	自动放置的特点	67
7.4.2	准备	67
7.4.3	互动式自动放置	67
7.4.4	附加说明	67
8	设置布线参数	68
8.1	当前设置	68
8.1.1	访问主对话框	68
8.1.2	当前设置	68
8.2	常规选项	68
8.3	网类	70
8.3.1	设置布线参数	70
8.3.2	网类编辑器	70
8.3.3	全局设计规则	71
8.3.4	过孔参数	72
8.3.5	布线参数	73
8.3.6	具体尺寸	73
8.4	示例和典型尺寸	73
8.4.1	布线宽度	73

8.4.2	绝缘（间隙）	73
8.5	例子	73
8.5.1	粗制	73
8.5.2	标准	74
8.6	手动布线	75
8.7	创建布线时的帮助	75
8.7.1	创建布线	75
8.7.2	移动和拖动布线	77
8.7.3	插入过孔	77
8.8	选择/编辑布线宽度和过孔尺寸	78
8.8.1	使用水平工具栏	78
8.8.2	使用弹出菜单	78
8.9	编辑和更改布线	79
8.9.1	更改布线	79
8.9.2	全局更改	80
9	交互式布线	83
9.1	配置	83
9.2	布线	85
9.3	设置布线宽度和通孔尺寸	86
9.4	拖动	86
9.5	选项	86
10	创建铜区	88
10.1	在铜层上创建区域	88
10.2	创建区域	88
10.2.1	创建区域的限制	88
10.2.2	优先级	90
10.2.3	填充区域	91
10.3	填充选项	93
10.3.1	填充模式	93
10.3.2	间隙和最小铜厚度	94
10.3.3	焊盘选项	94

10.3.4 热释放参数	95
10.3.5 选择参数	96
10.4 在区域内添加剪切区域	96
10.5 边框编辑	98
10.5.1 添加类似的区域	100
10.6 编辑区域参数	101
10.7 最后区域填充	101
10.8 更改区域网名	101
10.9 在技术层上创建区域	102
10.9.1 创建区域限制	102
10.10 创建禁止布线区域	104
11 用于电路制造的文件	106
11.1 最后的准备	106
11.2 最终的 DRC 测试	107
11.3 设置坐标原点	108
11.4 生成用于照片布线的文件	109
11.4.1 GERBER 格式	110
11.4.2 POSTSCRIPT 格式	111
11.4.3 绘图选项	111
11.4.4 其他格式	113
11.5 阻焊层和焊膏层的全局间隙设置	113
11.5.1 访问	114
11.5.2 阻焊层间隙	115
11.5.3 锡膏层间隙	115
11.6 生成钻孔文件	115
11.7 生成布线文档	116
11.8 生成用于自动元件插入的文件	117
11.9 高级布线选项	117

12 封装编辑器 - 管理库	120
12.1 封装编辑器概述	120
12.2 访问封装编辑器	121
12.3 封装编辑器用户界面	121
12.4 封装编辑器中的顶部工具栏	122
12.5 创建一个新库	123
12.6 在活动库中保存封装	123
12.7 将封装从一个库转移到另一个库	123
12.8 在活动库中保存电路板的所有封装	124
12.9 库封装的文档	124
12.10 记录库 - 推荐的做法	125
12.11 封装库管理	127
12.12 3D 形状库管理	127
13 封装编辑器 - 创建和编辑封装	128
13.1 封装编辑器概述	128
13.2 封装元素	128
13.2.1 焊盘	128
13.2.2 边框	129
13.2.3 字段	129
13.3 启动封装编辑器并选择要编辑的封装	129
13.4 封装编辑器工具栏	129
13.4.1 编辑工具栏 (右侧)	130
13.4.2 显示工具栏 (左侧)	130
13.5 上下文菜单	131
13.6 封装属性对话框	134
13.7 创建新的封装	135
13.8 添加和编辑焊盘	136
13.8.1 添加焊盘	136
13.8.2 设置焊盘属性	136
13.8.2.1 矩形焊盘	137
13.8.2.2 旋转焊盘	137
13.8.2.3 无镀通孔焊盘	137

13.8.2.4 偏移参数	138
13.8.2.5 Delta (希腊字母: 得尔塔) 参数 (梯形焊盘)	138
13.8.3 设置阻焊层和焊膏层的间隙	138
13.8.3.1 焊膏层设置	139
13.8.4 焊盘不在铜层上	139
13.9 字段属性	140
13.10 自动放置封装	140
13.11 属性	141
13.12 库封装的文档	141
13.13 三维可视化	143
13.13.1 3D 模型路径	144
13.14 将封装保存到活动库中	145
13.15 保存封装到电路板	145
14 高级的 PCB 编辑工具	146
14.1 复制项目	146
14.2 精确移动项目	146
14.3 阵列工具	147
14.3.1 激活阵列工具	147
14.3.2 网格阵列	148
14.3.2.1 几何选项	148
14.3.2.2 编号选项	150
14.3.3 圆形阵列	150
14.3.3.1 几何选项	151
14.3.3.2 编号选项	151
14.4 测量 (标尺) 工具	151
15 KiCad 脚本参考	153
15.1 KiCad 对象	153
15.2 基本 API 参考	153
15.3 加载和保存板 (Board)	154
15.4 列出和加载库	154
15.5 板 (BOARD)	155

- 15.6 例子 157
 - 15.6.1 更改元件引脚的焊膏层边距 157
- 15.7 封装向导 157
- 15.8 动作插件 161

参考手册

Copyright

本文档由以下列出的贡献者版权所有 (C)2010-2015。您可以根据 GNU 通用公共许可证(<http://www.gnu.org/licenses/gpl.htm>) 版本 3 或更高版本或知识共享署名许可的条款进行分发和/或修改。(<http://creativecommons.org/licenses/by/3.0/>) ,3.0 或更高版本。

本指南中的所有商标均属于其合法所有者。

贡献者

Jean-Pierre Charras, Fabrizio Tappero.

翻译

taotieren <admin@taotieren.com>, 2019

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

反馈

请将任何错误报告、建议或新版本引导到此处:

- 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 i18n: <https://gitlab.com/kicad/code/kicad-i18n/issues>

出版日期和软件版本

2014 年 3 月 17 日。

Chapter 1

Pcbnew 简介

1.1 描述

Pcbnew 是一款功能强大的印刷电路板软件工具，适用于 Linux，Microsoft Windows 和 Apple OS X 操作系统。Pcbnew 与原理图捕获程序 Eeschema 结合使用，以创建印刷电路板。

Pcbnew 管理封装库。每个覆盖区都是物理元件的图形，包括其焊盘图案（电路板上焊盘的布局）。在读取网表期间会自动加载所需的封装。封装选择或注释的任何更改都可以在原理图中更改，并通过重新生成网表并再次在 pcbnew 中读取，在 pcbnew 中更新。

Pcbnew 提供了一种设计规则检查（DRC）工具，可防止布线和焊盘间隙问题，并防止网络/原理图中未连接的网络连接。使用交互式布线时，它会持续运行设计规则检查，并有助于自动布线各个布线。

Pcbnew 提供了一个飞线显示器，一条连接封装焊盘的飞线连接在原理图上。这些连接在布线和封装移动时动态移动。

Pcbnew 有一个简单但有效的自动布线器，可以帮助生产电路板。SPECCTRA dsn 格式的导出/导入允许使用更高级的自动布线器。

Pcbnew 提供专门用于生产超高频微波电路的选项（例如梯形和复杂形式的焊盘，印刷电路上线圈的自动布局等）。

1.2 主要设计特色

Pcbnew 中最小的单位是 1 纳米。所有尺寸都存储为整数纳米。

Pcbnew 可生成多达 32 层铜，14 层技术层（丝印层，阻焊层，元件粘合剂层，焊膏层和边缘切割层）以及 4 个辅助层（图纸和注释），并实时管理飞线指示（飞线）丢失的布线。

PCB 元素（布线，焊盘，文本，图纸……）的显示可自定义：

- 完整或边框。
- 有无布线间隙。

对于复杂电路，可以选择性地隐藏层，区域和元件的显示以便在屏幕上清晰显示。可以高亮显示网的布线以提供高对比度。

封装可以旋转到任何角度，分辨率为 0.1 度。

Pcbnew 包含一个封装编辑器，可以编辑 PCB 上的单个封装或编辑库中的封装。

封装编辑器提供了许多省时工具，例如：

- 只需按照您希望编号的顺序将鼠标拖到焊盘上即可快速填充焊盘编号。
- 轻松生成用于 LGA/BGA 或圆形封装的矩形和圆形焊盘阵列。
- 半自动对齐行或列的焊盘。

封装焊盘具有可调节的各种属性。焊盘可以是圆形，矩形，椭圆形或梯形。对于通孔部件，钻头可以在焊盘内部偏移并且是圆形或槽。单个焊盘也可以旋转并具有独特的阻焊，网或焊膏间隙。焊盘还可以具有牢固的连接或热释放连接，以便于制造。可以在封装内放置任何独特焊盘的组合。

Pcbnew 可轻松生成生产所需的所有文件：

- 制造输出：
 - 用于 GERBER RS274X 格式的 Photoplotters 的文件。
 - 用于以 EXCELLON 格式钻孔的文件。
- 以 HPGL, SVG 和 DXF 格式绘制文件。
- 以 POSTSCRIPT 格式绘制和钻取地图。
- 本地打印输出。

1.3 一般建议

由于必要的控制程度，强烈建议使用带有 pcbnew 的 3 键鼠标。平移和缩放等许多功能都需要 3 键鼠标。

在 KiCad 的新版本中，pcbnew 已经从 CERN 的开发人员那里看到了广泛的变化。这包括诸如新渲染器（OpenGL 和 Cairo 视图模式），交互式推送布线器，差分 and 曲折布线和调整，重新设计的封装编辑器以及许多其他功能等功能。请注意，大多数这些新功能 **仅**存在于新的 OpenGL 和 Cairo 视图模式中。

Chapter 2

安装

2.1 安装软件

KiCad 文档中描述了安装过程。

2.2 修改默认配置

在 “kicad/share/template” 中提供了默认配置文件 “kicad.pro”。此文件用作所有新项目的初始配置。

可以修改此配置文件以更改要加载的库。

去做这个：

- 使用 KiCad 或直接启动 Pcbnew。在 Windows 上它位于 `C:\kicad\bin\pcbnew.exe` 中，在 Linux 上你可以运行 `/usr/local/kicad/bin/kicad` 或 `/usr/local/kicad/bin/pcbnew` 二进制文件位于 `/usr/local/kicad/bin` 中。
- 选择首选项 - 库 (Libs) 和目录 (Dir)。
- 根据需要编辑。
- 将修改后的配置 (Save Cfg) 保存到 “kicad/share/template/kicad.pro”。

2.3 管理封装库

从 4.0 版本开始，Pcbnew 使用名为“封装库表”的文件组织封装库。封装库表包含一些单独的封装库的描述，以及每个库的“昵称”，用于在引用封装时引用该库。

Pcbnew 支持几种库，每种都由“插件”支持：

- KiCad - 以 *.pretty* 格式存储在本地文件系统上的本机 KiCad 封装库（包含 *.kicad_mod* 文件的文件夹）
 - Github - *.pretty* 格式的原生 KiCad 封装库，作为 Github 存储库在线存储
-

- 旧版 - 旧式 KiCad 封装库 (*.mod* 文件)
- Eagle - Eagle 封装库 (包含 *.fp* 文件的文件夹)
- Geda-PCB - Geda PCB 库

Note

- 您只能在本地磁盘上写入 KiCad *.pretty* 封装库文件夹 (以及这些文件夹中的 *.kicad_mod* 文件)。
 - 所有其他格式都是只读的。
-

允许在不同的库中具有相同名称的封装。封装将存储为库 和封装名称的组合，以确保从相应的库加载正确的封装。有两个封装库表：全局表和项目表。

2.3.1 全局封装库表

全局封装库表包含始终可用的库列表，与当前加载的项目文件无关。该表保存在用户主文件夹中的文件 “fp-lib-table” 中。此文件夹的位置取决于操作系统。

2.3.2 项目特定封装库表

项目特定的封装库表包含专门用于当前加载的项目文件的库列表。项目特定的封装库表只能在与项目板文件一起加载时进行编辑。如果没有加载项目文件或项目路径中没有封装库表文件，则会创建一个空表，可以对其进行编辑，然后将其与板文件一起保存。

当在项目特定表中定义条目时，包含条目的 “fp-lib-table 文件” 将被写入当前打开的 PCB 的文件夹中。

2.3.3 初始配置

第一次运行 CvPcb 或 Pcbnew 并且在用户的主文件夹中找不到全局封装表文件 “fp-lib-table”，Pcbnew 将尝试复制存储在系统的 KiCad 模板文件夹中的默认封装表文件 “fp_global_table” 到用户主文件夹中的 “fp-lib-table” 文件。如果找不到 “fp_global_table”，将在用户的主文件夹中创建一个空的封装库表。如果发生这种情况，用户可以手动复制 “fp_global_table” 或手动配置表。

默认的封装库表包括作为 KiCad 的一部分安装的所有标准封装库。

Tip

在官方 [KiCad 库存储库](#) 中也有示例 “fp-lib-table” 文件，您可以将它们用作您自己的起点：

- 通过 Github 的所有 KiCad 库： [fp-lib-table.for-github](#)
 - 所有 KiCad 库，假设它们已经在您的磁盘上 (如果您还没有它们，则需要下载它们)： [fp-lib-table.for-pretty](#)
 - 标准 Eagle 库 (适用于 Eagle 6.4.0) [fp-lib-table.for-eagle-6.4.0](#)
-

配置 KiCad 时要做的第一件事就是根据您的工作和项目所需的库来修改此表（添加/删除条目）。

Tip

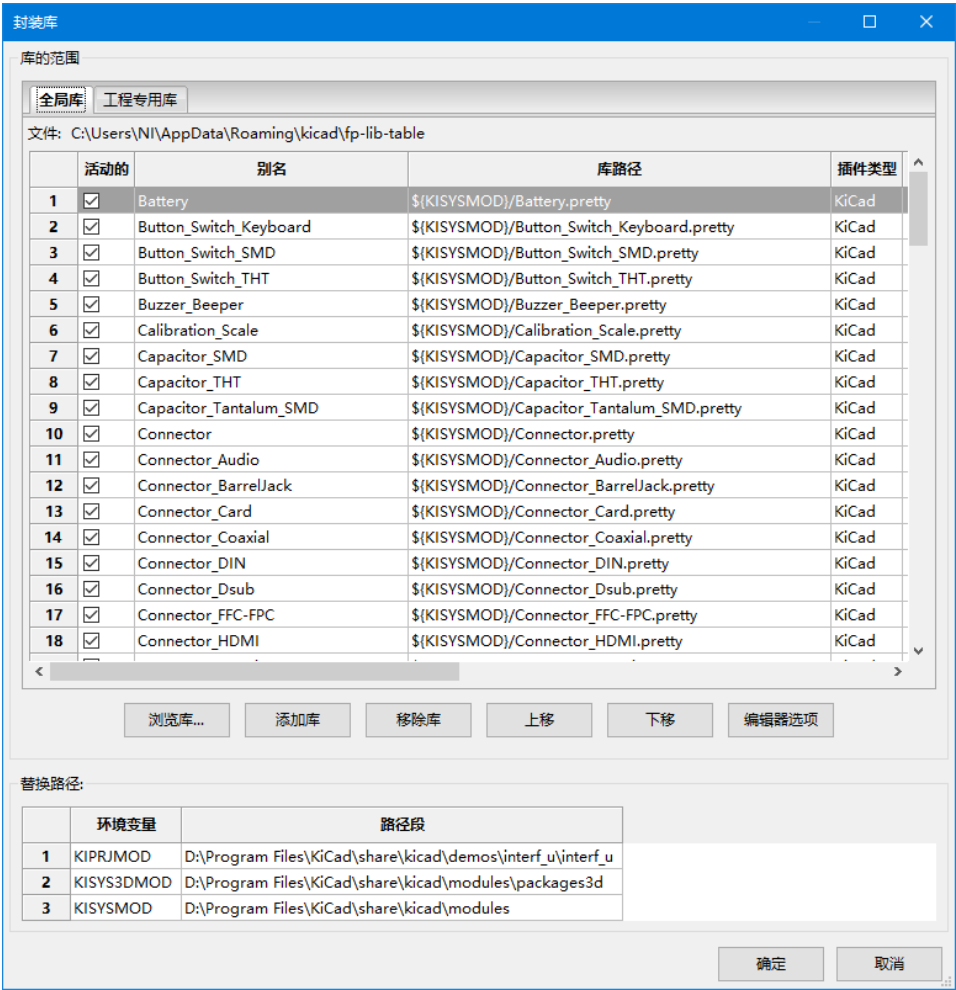
拥有许多库可能非常耗时，特别是如果它们只在网上找到（例如 Github 库）。如果您发现库加载缓慢，请尝试删除不需要的库。

2.3.4 使用库管理器添加表条目

可以通过以下方式访问库表管理器：



下图显示了封装库表编辑对话框，可以通过从“首选项”菜单调用“封装库管理器”条目来打开该对话框。



要使用封装库，必须先将其添加到全局表或项目特定表中。项目特定表仅适用于打开板文件的情况。

每个库表条目都有一个昵称。这个 必须 (*must*) 在该表中是唯一的。昵称不必以任何方式与实际库文件名或路径相关。

有效的库表条目有一些规则：

- 冒号 “:” 字符不能在昵称中的任何位置使用。
- 每个库条目必须具有有效的路径和/或文件名，具体取决于库的类型。路径可以定义为绝对，相对或环境变量替换（见下文）
- 必须选择适当的插件类型才能使库成为必需的。必须选择适当的插件类型才能正确读取库。阅读。

还有一个描述字段用于添加库条目的描述。选项字段包含特定于插件的特殊选项，通常为空白。

虽然在同一个表中不能有重复的库昵称，但是在全局和项目特定的封装库表中都可以有重复的库昵称。当出现重复的名称时，项目特定的表条目将优先于全局表条目。

2.3.5 环境变量替代

封装库表最强大的功能之一是环境变量替换。这允许您定义库存储在环境变量中的自定义路径。

KiCad 定义了一些默认变量：

- “\$KISYSMOD”: 这指向与 KiCad 一起安装的默认封装库所在的位置。您可以通过自己定义来覆盖 “\$KISYSMOD”，它允许您替换自己的库来代替默认的 KiCad 封装库。
- 加载板文件时，使用该板的路径定义 “\$KPRJMOD”。这允许您引用项目路径中的库，而无需在项目特定的封装库表中重复库的绝对路径。

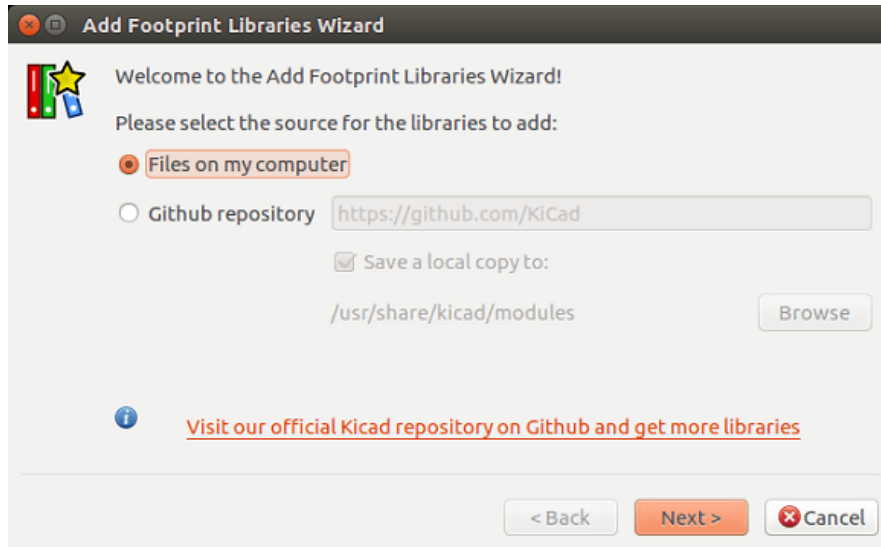
2.3.6 使用库向导添加表条目

有一个交互式向导，可以帮助您将库添加到库表。可从菜单访问：

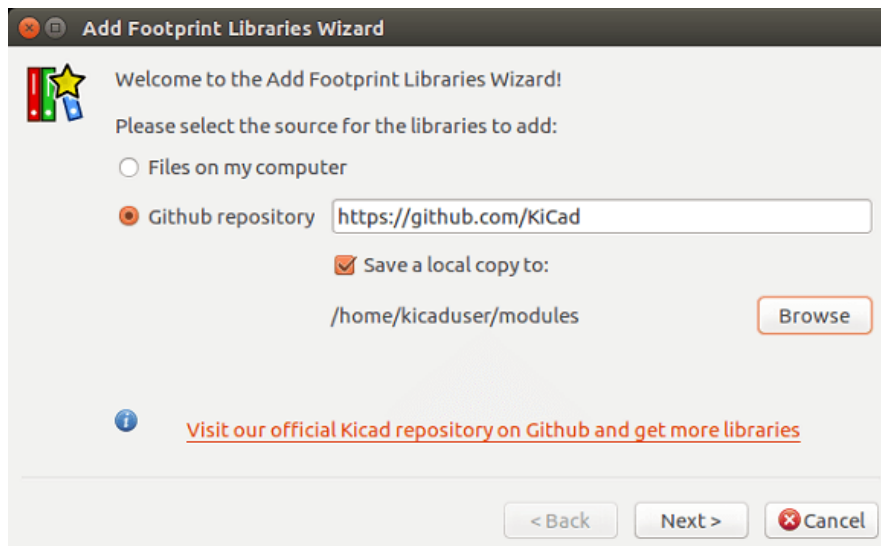


它也可以使用“附加向导”按钮从库管理器启动。

在这里，选择了本地库选项。

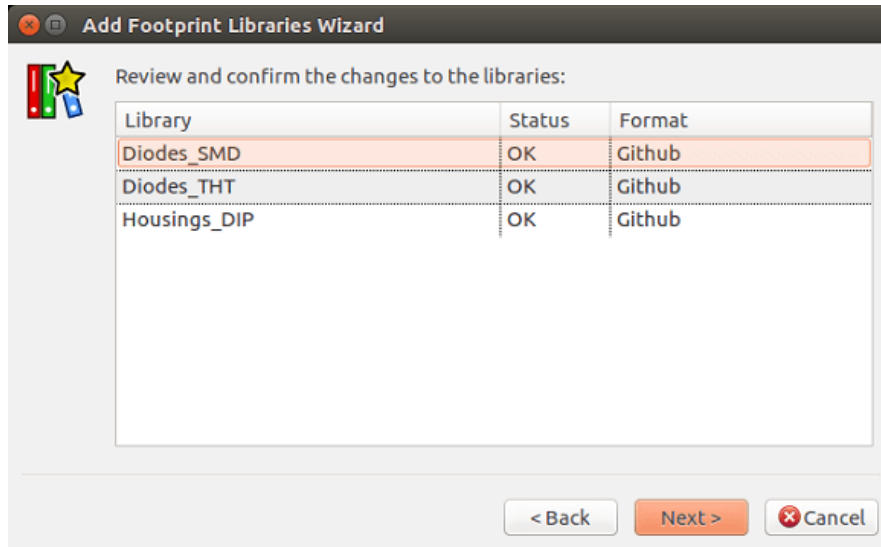


此处，选择了远程库选项。



然后，向导将引导您完成添加库的步骤，这将取决于您要添加的库的类型。下面将解释每种类型的过程。

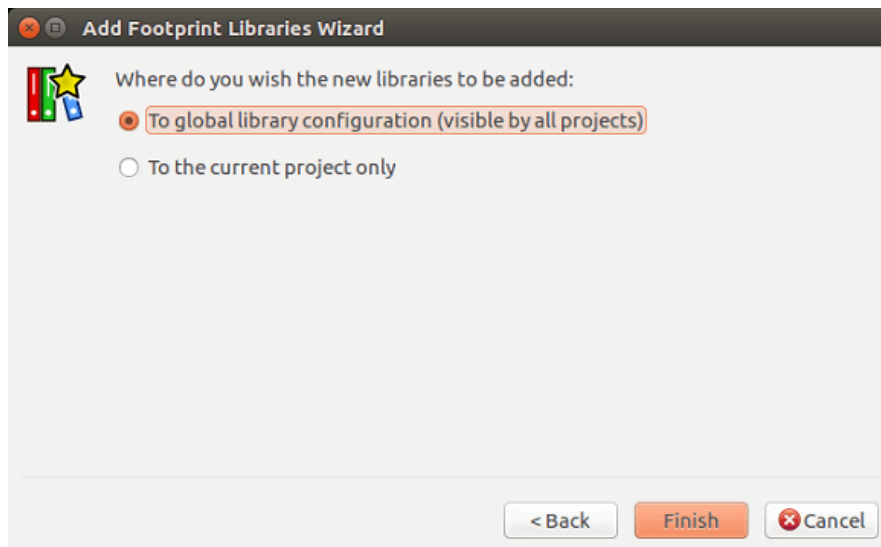
选择一组库后，下一页将验证选项：



如果某些选定的库不正确（不支持，而不是封装库……），它们将被标记为“无效”。

最后一个选择是要填充的封装库表：

- 全局表，或
- 项目特定表

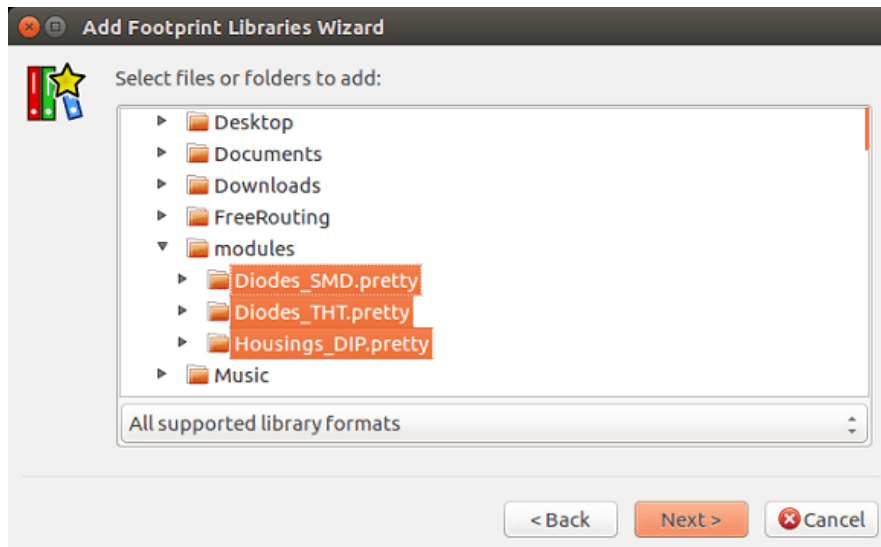


2.3.6.1 添加现有本地库

您的计算机上可能已有本地库。例如：

- 以前下载的 KiCad 漂亮的目录
- 来自旧安装的旧版 KiCad *.mod* 文件
- Geda 或 Eagle 库

这些可以通过“我的计算机上的文件”选项添加。系统将要求您提供要添加的库的目录以及格式：



如果您不选择格式，向导将尝试猜测正确的格式。

2.3.6.2 从 Github 添加库

该向导还可以使用“Github 存储库”选项从 Github 添加库。

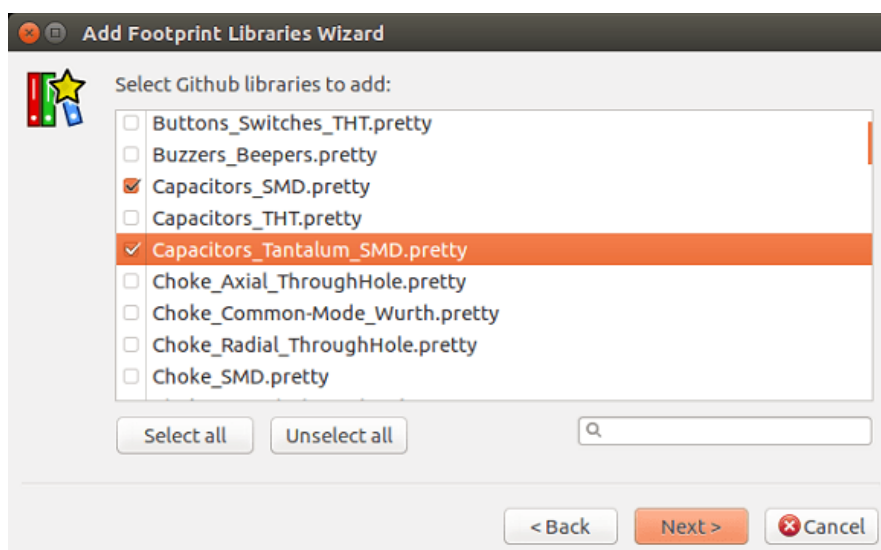
您需要指定包含要添加的存储库的 Github 帐户。

Tip

官方 KiCad 库 Github 帐户是 <https://github.com/KiCad>

您可以选择保存本地副本。如果您执行 `_not_` 保存本地副本，则该库将是 `_Github_` 库，并将在每次重新加载库时重新同步。如果您 `do` 保存本地副本，该库将是 *KiCad*（漂亮）库，并且将来不会自动更新。

下一页将加载在该 Github 帐户上找到的 `.pretty` 存储库列表。您可以选择要添加到库中的任何数字。



确认后，如果您选择保存副本，脚印将立即下载到指定的本地位置。如果您使用的是 Github 插件（没有本地副本），则在需要时会从 Github 加载封装。

2.3.7 使用 KiCad 插件

KiCad 插件处理您计算机上存在的本机 KiCad 库（或某些可访问的文件系统）。

它用于与 KiCad 一起安装的预安装库，以及其他 KiCad 库，可以是官方的 KiCad 库集，第三方库或您自己的策划库。

2.3.7.1 安装 KiCad 插件库

Footprint Library Wizard 可以帮助您安装磁盘或 Github 上的库。但是，对于磁盘上的库，您需要首先将它们放在那里。

KiCad 库是包含一些 `.kicad_mod` 文件的目录。

这通常通过解压缩归档文件，从其他位置复制目录或克隆版本控制的存储库来完成。

KiCad 插件没有指定任何类型的版本控制，但 Git 非常常用于跟踪库的更改，这对于确保库数据的安全记录和备份至关重要。

跟踪变化很容易，并为官方的 KiCad Github 库做出贡献。这是使用 Git 版本控制软件完成的。如果你想回馈，你必须在 Github 上分配回购，这样你就可以发送拉请求。如果您只是想在需要时更新库，则不需要这样做，您可以直接克隆官方 KiCad 库并根据需要进行拉取。

Note

通过 Github 发送拉取请求将允许自动库标准检查器验证您提议的更改。有关库约定的详细信息，请参阅 [KiCad Library Conventions](#)。

2.3.8 使用 GitHub 插件

GitHub 插件是一个特殊的插件，它提供了一个接口，用于对包含 `.pretty` 封装的远程 GitHub 存储库进行只读访问，并可选择提供“写时复制”（COW）支持，用于编辑从 GitHub 存储库读取的封装和在本地保存它们。



Important

- “GitHub” 插件用于 [上的远程漂亮封装库](https://github.com) 的只读访问 <https://github.com>。
 - 自上次使用它以来，您不会被告知远程存储库是否已更改。直接从 Github 使用封装时要小心。
-

要将封装库表中的 GitHub 条目添加到封装库表条目中的“库路径”，必须将其设置为有效的 GitHub URL。

例如：

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

通常 GitHub URL 采用以下形式：

```
https://github.com/user_name/repo_name
```

“插件类型”必须设置为“Github”。

下表显示了具有默认选项（无 COW 支持）的封装库表条目：

昵称	库路径	插件类型	选项	描述
GitHub 上	https://github.com/-liftoff-sr/-pretty_footprints	GitHub 上		Liftoff 的 GH 封装

2.3.8.1 写入时复制 (Copy-On-Write)

要启用“写入时复制”功能，必须将“allow_pretty_writing_to_this_dir”选项添加到封装库表条目的“选项”设置中。此选项是“库路径”，用于本地存储从 GitHub 存储库读取的修改过的封装副本。保存到此路径的占用空间与 GitHub 存储库的只读部分组合以创建封装库。如果缺少此选项，则 GitHub 库是只读的。如果 GitHub 库存在该选项，则对该混合库的任何写入都将转到本地“*.pretty”目录。

此混合 COW 库的 github.com 常驻部分始终是只读的，这意味着您无法删除任何内容或直接修改指定 GitHub 存储库中的任何封装。在所有进一步的讨论中，聚合库类型仍然是“Github”，但它包括本地读/写部分和远程只读部分。

昵称	库路径	插件类型	选项	描述
GitHub 上	https://github.com/-liftoff-sr/-pretty_footprints	GitHub 上		Liftoff 的 GH 封装

封装载荷将始终优先于选项“allow_pretty_writing_to_this_dir”给出的路径中找到的局部封装。通过在封装编辑器中执行封装保存将封装保存到 COW 库的本地目录后，在加载与您本地保存的封装相同的封装时，不会看到 GitHub 更新。

始终为每个 GitHub 库保留一个单独的本地“*.pretty”目录，永远不要通过多次引用同一目录来组合它们。另外，不要在封装库表条目中使用相同的 COW (“*.pretty”) 目录。这可能会造成混乱。选项“allow_pretty_writing_to_this_dir”的值将使用“\${}”符号扩展任何环境变量，以与“库路径”设置相同的方式创建路径。

2.3.8.2 使用写入时复制 (Copy-On-Write) 共享封装

COW 有什么意义？如果您定期通过电子邮件将您的 COW 相当封装修改发送给 GitHub 存储库维护者，您可以帮助更新 GitHub 副本。只需将您在 COW 目录中找到的单个“*.kicad_mod”文件通过电子邮件发送给 GitHub 存储库的维护者。在收到确认已提交更改后，您可以安全地删除您的 COW 文件，并且 GitHub 库的只读部分中更新的封装将向下流动。您的目标应该是通过 <https://github.com> 上的共享主副本频繁提供 COW 文件设置尽可能小。

Tip

您还可以使用 KiCad 插件使用相关库的本地 Git 克隆为库开发做出贡献，并向库维护者提交拉取请求。

2.3.8.3 缓存 GitHub 请求

GitHub 插件可能很慢，因为它必须先从 Internet 下载所有库才能使用它们。

Nginx 可以用作 GitHub 服务器的缓存，以加快封装的加载速度。它可以安装在本地或网络服务器上。在“pcbnew/github/nginx.conf”的 KiCad 源代码中有一个示例配置。最直接的方法就是使用这个和“export KIGHUB=http://my_server:54321/KiCad”覆盖默认的 nginx.conf，其中“my_server”是运行 Nginx 的机器的 IP 或域名。

2.3.9 使用模式

封装库可以全局定义，也可以专门定义到当前加载的项目。用户全局表中定义的封装库始终可用，并存储在用户主文件夹的“fp-lib-table”文件中。即使没有打开项目网表文件，也可以随时访问全局封装库。项目特定的封装表仅对当前打开的网络列表文件有效。项目特定的封装库表保存在当前打开的板文件路径中的文件“fp-lib-table”中。您可以在任一表中自由定义库。

每种方法都有优点和缺点：

- 您可以在全局表中定义所有库，这意味着它们将在您需要时始终可用。
 - 这样做的缺点是您可能需要搜索大量的库来查找您正在寻找的足迹。
- 您可以根据项目特定定义所有库。
 - 这样做的好处在于你仅需要定义你在某个项目中需要用到的封装，这将大大减小搜索的复杂度。
 - 缺点是您始终需要记住为每个项目添加所需的每个封装库。
- 您还可以在全局和项目中专门定义封装库。

一种用法模式是全局定义最常用的库，而库只需要项目特定库表中的项目。您如何定义库没有限制。

2.3.9.1 修改 PCB 项目中的封装

将占位面积添加到 PCB 时，整个封装将复制到 PCB 文件中（*.kicad_pcb*）。这意味着库中的封装的更改不会自动影响 PCB。

这也意味着您可以单独编辑 PCB 上的封装，而不会影响相同封装的其他实例（在同一 PCB 或其他 PCB 上）。

但是，如果修改库占用空间，则下次放置实例时，它将与现有的同名封装不匹配。

Tip

通常的做法是将您使用的所有占用空间复制到单独的版本控制位置，以便此项目不会因系统或用户库的更改而意外受到影响。此外，它确保用于 PCB 的所有封装资源都可以轻松地与 PCB 文件一起分发。

Chapter 3

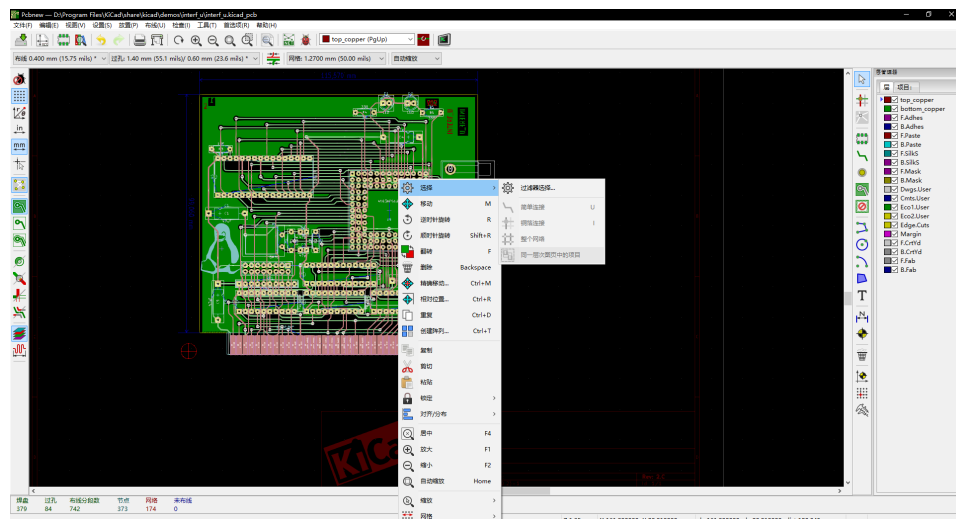
一般操作

3.1 工具栏和命令

在 Pcbnew 中，可以使用各种方法执行命令：

- 主窗口顶部的基于文本的菜单。
- 顶部工具栏菜单。
- 右侧工具栏菜单。
- 左侧工具栏菜单。
- 鼠标按钮 (菜单选项)。特别：
 - 鼠标右键显示一个弹出菜单，其内容取决于鼠标箭头下的元素。
- 键盘 (功能键 “F1”，“F2”，“F3”，“F4”，“Shift”，“Delete”，“+”，“-”，“Page Up”，“Page Down” 和 “空格键”)。“Esc” 键通常会取消正在进行的操作。

下面的屏幕截图说明了对这些操作的一些可能访问：



3.2 鼠标命令

3.2.1 基本命令

- 左键
 - 单击可在下方状态栏中显示光标下的覆盖区或文本的特征。
 - 双击显示光标下元素的编辑器（如果元素可编辑）。
- 中心按钮/滚轮
 - 快速缩放和图层管理器中的一些命令。
 - 按住中心按钮并绘制一个矩形以缩放到所描述的区域。旋转鼠标滚轮将允许您放大和缩小。
- 右键
 - 显示弹出菜单

3.2.2 块上的操作

通过弹出菜单可以进行移动，反转（镜像），复制，旋转和删除块的操作。此外，视图可以缩放到块描述的区域。

通过在按住鼠标左键的同时移动鼠标来跟踪块的框架。释放按钮时执行操作。

通过按住其中一个热键“Shift”或“Ctrl”，或同时按住“Shift”和“Ctrl”两个键，在绘制块时，操作反转，旋转或删除将自动选择，如下表所示：

活动	影响
按住鼠标左键	布线移动块的框架
按住“Shift”+ 鼠标左键	反转块的布线框架
按住“Ctrl”键 + 鼠标左键	布线框架旋转块 90°
按住 Shift 键 + “Ctrl” + 鼠标左键	布线框架删除块
按下中心鼠标按钮	布线框架缩放到阻止

移动块时：

- 将块移动到新位置并操作鼠标左键以放置元素。
- 要取消操作，请使用鼠标右键并从菜单中选择“取消阻止”（或按“Esc”键）。

或者，如果在绘制块时未按任何键，请使用鼠标右键显示弹出菜单并选择所需的操作。

对于每个块操作，选择窗口使操作仅限于某些元素。

3.3 选择网格大小

在元素布局期间，光标在网格上移动。可以使用左侧工具栏上的图标打开或关闭网格。

可以使用弹出窗口或屏幕顶部工具栏上的下拉选择器来选择任何预定义的网格大小或用户定义的网格。使用菜单栏选项“尺寸” -> “用户网格大小”设置“用户定义”网格的大小。

3.4 调整缩放级别

可以使用以下任一方法更改缩放级别：

- 打开弹出窗口（使用鼠标右键），然后选择所需的缩放。
- 使用以下功能键：
 - ‘F1’：放大（放大）
 - ‘F2’：缩小（缩小）
 - ‘F3’：重绘显示
 - ‘F4’：当前光标位置的中心视图
- 旋转鼠标滚轮。
- 按住鼠标中键并绘制一个矩形以缩放到所描述的区域。

3.5 显示光标坐标

光标坐标以英寸或毫米显示，使用左侧工具栏上的“in”或“mm”图标进行选择。

无论选择哪个单位，Pcbnew 的精度始终为 1/10,000 英寸。

屏幕底部的状态栏给出：

- 当前的缩放设置。
- 光标的绝对位置。
- 光标的相对位置。注意，通过按空格键可以在任何位置将相对坐标（x，y）设置为（0,0）。然后相对于该新数据显示光标位置。

此外，可以使用其极坐标（光线 + 角度）显示光标的相对位置。可以使用左侧工具栏中的图标打开和关闭此功能。

Z 12.05	X 143.510000 Y 125.730000	dx 143.510000 dy 125.730000 dist 190.796	mm
---------	---------------------------	--	----

3.6 键盘命令 - 热键

可以使用键盘直接访问许多命令。选择可以是大写或小写。大多数热键都显示在菜单中。一些没有出现的热键是：

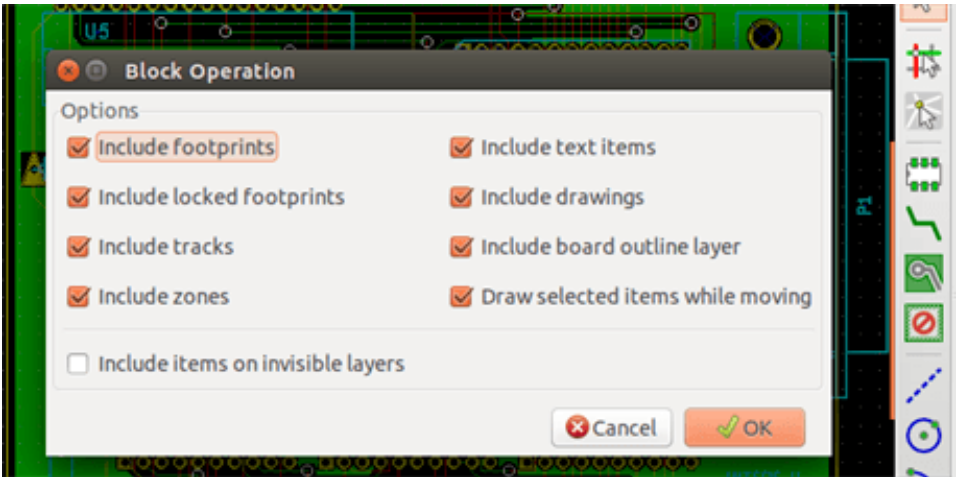
- ‘删除’：删除封装或布线。（仅在封装模式或布线模式处于活动状态时可用）
- ‘V’：如果布线正在进行中，如果布线工具处于活动状态，则切换工作层或放置过孔。
- ‘+’ 和 ‘-’：选择下一层或上一层。
- ‘Ctrl+F1’：显示所有热键的列表。
- ‘Space’：重置相对坐标。

3.7 在块上操作

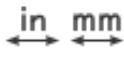
弹出菜单中提供了移动，反转（镜像），复制，旋转和删除块的操作。此外，视图可以缩放到块描述的视图。
通过在按住鼠标左键的同时移动鼠标来跟踪块的框架。释放按钮时执行操作。
通过按住“Shift”或“Ctrl”键中的一个键，同时将“Shift”和“Ctrl”键合在一起，或者“Alt”，在绘制块时，将自动选择反转，旋转，删除或复制操作，如图所示下表：

活动	影响
按住鼠标左键	移动块
按住 “Shift” + 鼠标左键	反转（镜像）块
按住 “Ctrl” 键 + 鼠标左键	旋转块 90°
按住 Shift 键 + “Ctrl” + 鼠标左键	删除块
按住 ‘Alt’ + 鼠标左键	复制块

执行块命令时，将显示一个对话框窗口，可以选择此命令中涉及的项目。
可以通过相同的弹出菜单或按 Esc 键（“Esc”）取消上述任何命令。



3.8 对话框中使用的单位

用于显示尺寸值的单位是英寸和毫米。可以通过按左侧工具栏中的图标来选择所需的单位： 但是在输入新值时，可以输入用于定义值的单位,。

可接受的单位是：

1 in	1 英尺
1 ”	1 英寸
25 th	25 thou
25 mi	25 mils, 和 thou 一样
6 mm	6 毫米

规则是:

- 数字和单位之间的空格被接受。
- 只有前两个字母很重要。
- 在使用替代小数分隔符而不是句点的国家/地区，也可以使用句点（‘.’’）。因此 ‘1,5’ 和 ‘1.5’ 在法语中是相同的。

3.9 顶级菜单栏

顶部菜单栏提供对文件的访问（加载和保存），配置选项，打印，绘图和帮助文件。

文件(F) 编辑(E) 视图(V) 设置(S) 放置(P) 布线(U) 检查(I) 工具(T) 首选项(R) 帮助(H)

3.9.1 “文件” 菜单



“文件” 菜单允许加载和保存印刷电路文件，以及打印和绘制电路板。它可以导出电路的导出（格式为 GenCAD 1.4），以便与自动测试仪配合使用。

3.9.2 编辑菜单

允许一些全局编辑操作：



3.9.3 查看菜单

允许：

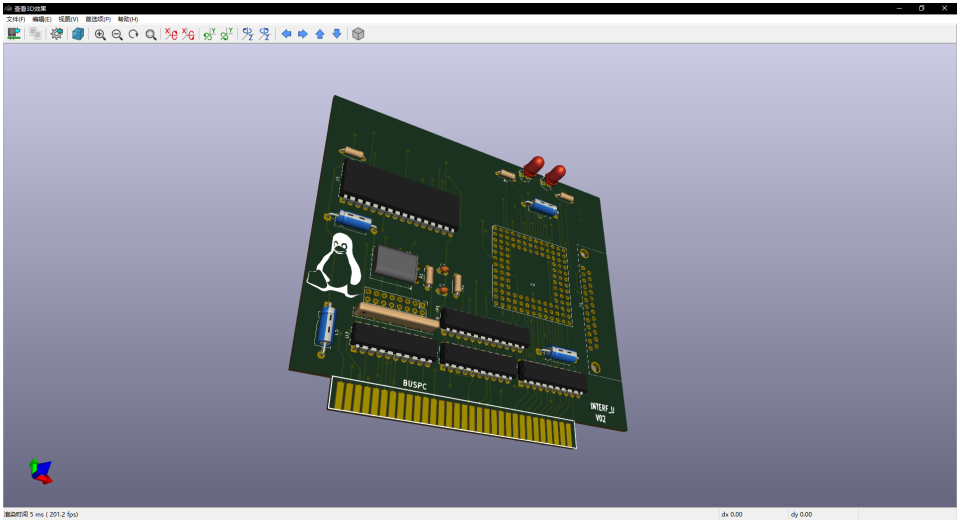
- 隐藏/显示图层管理器（用于显示图层和其他元素的颜色选择。还可以打开和关闭元素的显示）。
- 隐藏/显示微波工具栏。
- 显示库浏览器和 3D 查看器。
- 缩放功能
- 设置网格和单位
- 选择绘图模式和对比度模式



缩放功能和 3D 板显示。

3.9.3.1 3D 查看器

打开 3D 查看器。这是一个示例：



3.9.4 设置菜单

提供对 2 个对话框的访问:

- 设置图层（数字，已启用和图层名称）
- 设置设计规则（布线和过孔尺寸，间隙）。

一个重要的菜单。允许调整：

- 文本大小和图纸的线宽。
- 焊盘的尺寸和特性。
- 设置阻焊层和焊膏层的全局值



3.9.5 放置菜单

与右侧工具栏功能相同。



3.9.6 布线菜单

布线功能。



3.9.7 检查菜单

允许：

- 列出网

- 测量功能
- 设计规则检查器



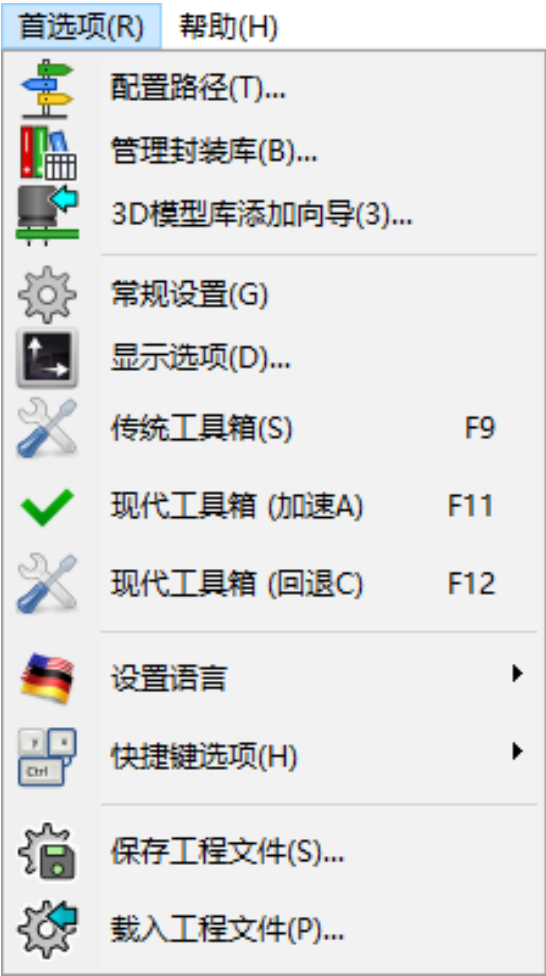
3.9.8 工具菜单

允许：

- 显示加载网表对话框
- 从原理图更新 PCB
- 从库中更新封装
- FreeRoute 协作
- Python 脚本控制台
- 外部插件



3.9.9 设置菜单



允许：

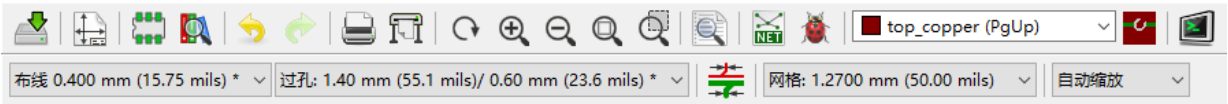
- 选择封装库。
- 一般选择的管理（单位等）。
- 管理其他显示选项。
- 创建，编辑（和重新读取）热键文件。

3.9.10 帮助菜单

提供对用户手册和版本信息菜单（Pcbnew About）的访问。

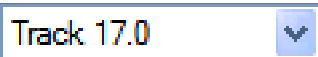

3.10 使用顶部工具栏上的图标


此工具栏可以访问 Pcbnew 的主要功能。



	创建一个新的印刷电路。
	打开旧印刷电路。.
	保存印刷电路。
	选择页面大小和修改文件属性。
	打开封装编辑器以编辑库或 PCB 封装。
	打开封装查看器以显示库或 PCB 封装。
	撤消/重做最后命令 (10 级)
	显示打印菜单。
	显示绘图菜单。
	放大和缩小 (相对于屏幕中心)。
	重绘屏幕
	适合页面
	查找封装或文字。
	网表操作 (选择, 阅读, 测试和编译)。
	DRC (设计规则检查): 自动检查布线。
	Soudure (PgDn) ▼
	选择工作层。
	选择层对 (用于过孔)
	封装模式: 当激活时, 启用封装选项弹出窗口。
	布线模式: 激活时启用布线选项弹出窗口
	直接访问布线 Freerouter
	显示/隐藏 Python 脚本控制台

3.10.1 辅助工具栏

	选择已经使用的布线厚度。
	选择已经使用的过孔尺寸。

	自动布线宽度：如果在创建新布线时启用， 线的宽度 设置为现有布线的宽度。
<div>Grid 50.0</div>	选择网格大小。
<div>Zoom 128</div>	选择缩放。

3.11 右侧工具栏

通过此工具栏可以访问编辑工具以更改 Pcbnew 中显示的 PCB。

		选择标准鼠标模式
		通过单击布线或焊盘高亮显示选定的网络
		显示局部飞线（焊盘或封装）
		从库中添加封装
		布线和过孔的放置
		区域（铜平面）的放置
		禁止布线区域的放置（在铜层上）
		在技术层上绘制线（即不是铜层）
		在技术层（即不是铜层）上绘制圆圈
		在技术层（即不是铜层）上绘制弧
		放置文字
		在技术层（即不是铜层）上绘制尺寸
		绘制对齐标记（出现在所有图层上）
		删除光标指向的元素 Note: 删除时，如果有多个叠加元素指出，优先考虑最小（在减少优先级布线，文本，封装库）。功能“取消删除”上部工具栏允许取消最后一项删除
		偏移调整钻孔和放置文件

	网格原点。（网格偏移）。主要用于编辑和放置封装。也可以在“尺寸/网格”菜单中进行设置
---	--

- 放置封装，轨道，铜区，文本等。
 - 网络高亮显示。
 - 创建笔记，图形元素等。
 - 删除元素。
-

3.12 左侧工具栏

左侧工具栏提供影响 Pcbnew 界面的显示和控制选项。

		打开/关闭 DRC（设计规则检查）。 注意： DRC 时关闭可以进行不正确的连接
		打开/关闭网格显示 * 注意：* 可能不会显示小网格除非放大到足够远
		极性显示状态栏上的相对坐标开/关
		以英寸或毫米为单位显示/输入坐标或尺寸
		更改光标显示形状
		显示一般飞线（封装之间不完整的连接）
		显示封装飞线在移动时动态嵌套
		重新绘制布线时启用/禁用自动删除布线
		显示区域中的填充区域
		不要在区域中显示填充区域
		仅显示区域中填充区域的边框
		打开/关闭边框模式中的焊盘
		在边框模式中打开/关闭过孔
		打开/关闭边框模式中的布线显示。
		高亮显示模式开/关。在此模式下激活图层正常显示，显示所有其他图层灰色的适用于多层电路
		隐藏/显示图层管理器
		使用微波工具。正在开发中

3.13 弹出窗口和快速编辑





右键单击鼠标可打开弹出窗口。其内容取决于光标指向的元素。
这样可以立即访问：

- 更改显示（光标中央显示，放大或缩小或选择缩放）。
- 设置网格大小。
- 此外，右键单击元素可以编辑最常修改的元素参数。

下面的屏幕截图显示了弹出窗口的外观。

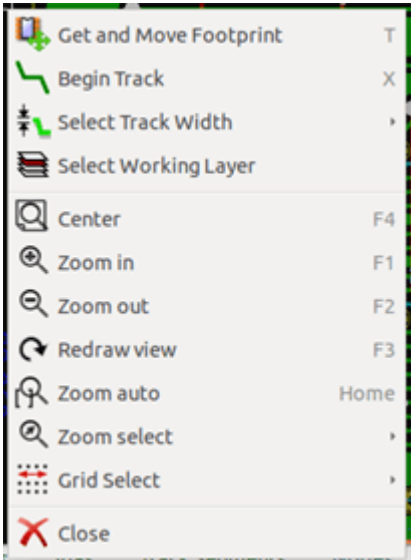
3.14 可用模式

使用弹出菜单时有 3 种模式。在弹出菜单中，这些模式添加或删除一些特定命令。

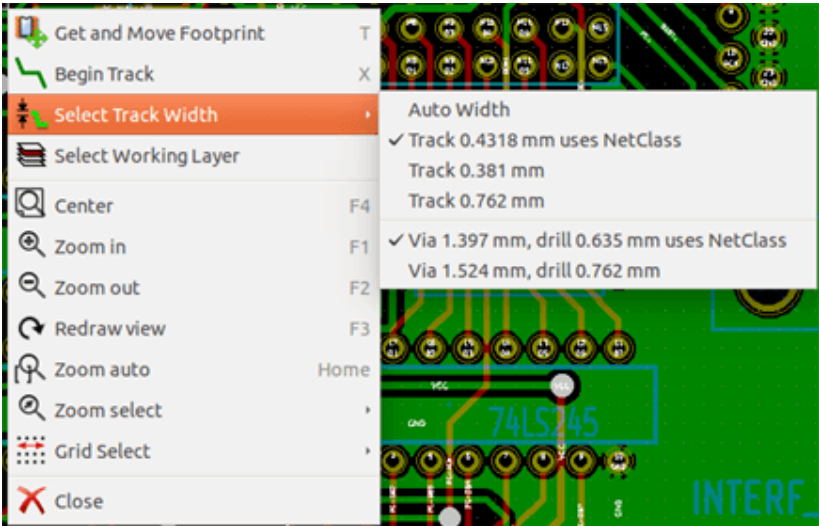
 and  disabled	正常模式
 enabled	封装模式
 enabled	布线模式

3.14.1 正常模式

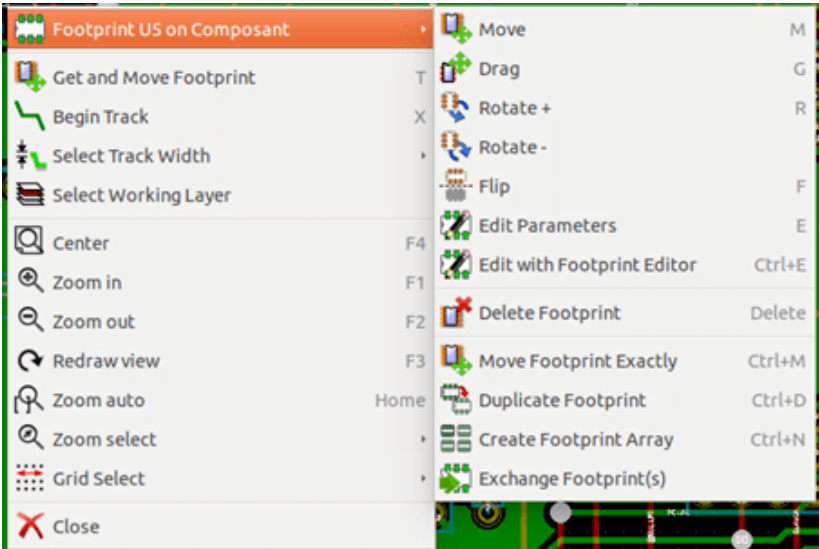
- 没有选择的弹出菜单：




- 选择了布线的弹出菜单：



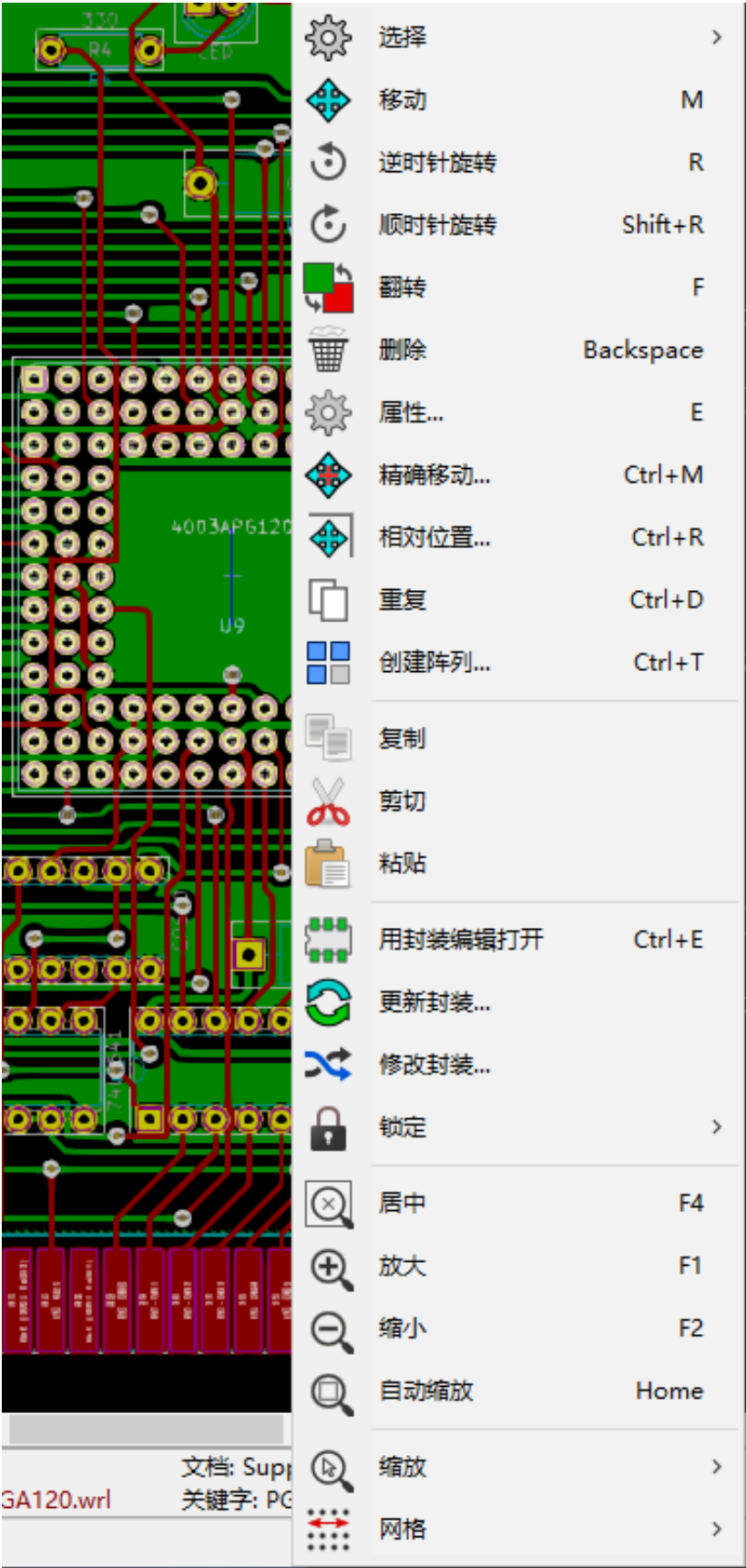
- 选择封装的弹出菜单：



3.14.2 封装模式

封装模式中的相同情况 ( 已启用)

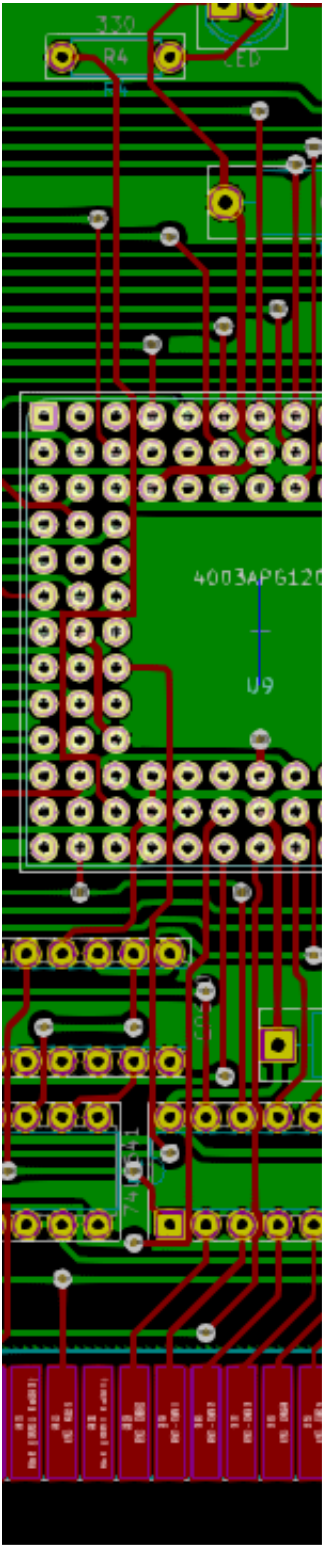
- 没有选择的弹出菜单：




- 选择了布线的弹出菜单:





- 选择封装的弹出菜单:





文档: Sup
关键字: PC
GA120.wrl


 选择 >


 移动 M


 逆时针旋转 R


 顺时针旋转 Shift+R


 翻转 F


 删除 Backspace


 属性... E


 精确移动... Ctrl+M


 相对位置... Ctrl+R


 重复 Ctrl+D


 创建阵列... Ctrl+T


 复制


 剪切


 粘贴


 用封装编辑打开 Ctrl+E


 更新封装...


 修改封装...


 锁定 >


 居中 F4

 放大 F1


 缩小 F2

 自动缩放 Home

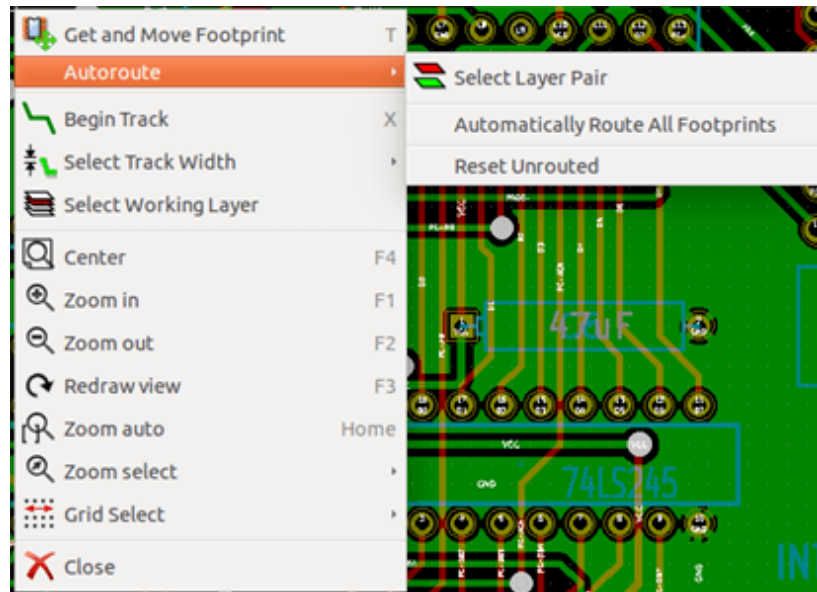
 缩放 >

 网格 >

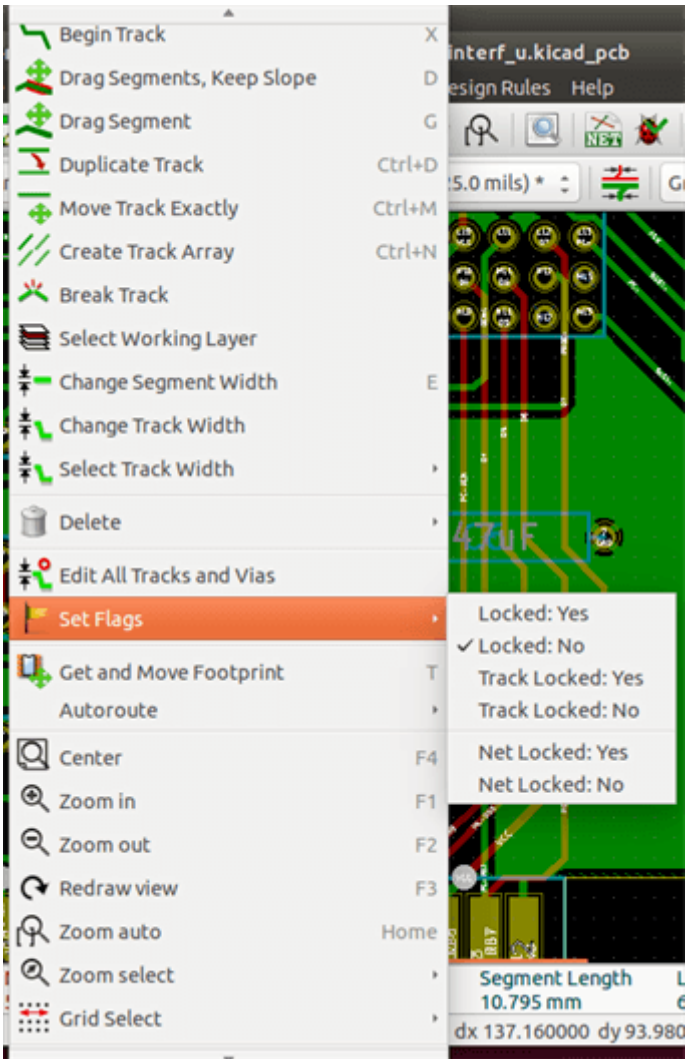
3.14.3 布线模式

布线模式中的相同情况（ 已启用）

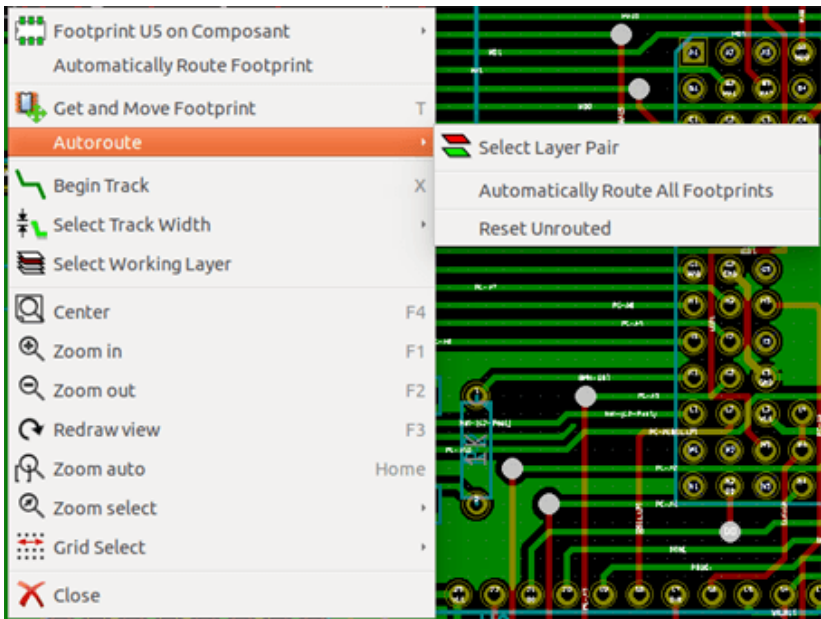
- 没有选择的弹出菜单：



- 选择了布线的弹出菜单：



- 选择封装的弹出菜单:



Chapter 4

原理图实施

4.1 将原理图链接到印刷电路板

一般而言，原理图表通过网表文件链接到其印刷电路板，网表文件通常由用于制作原理图的原理图编辑器生成。Pcbnew 接受使用 Eeschema 或 Orcad PCB 2 制作的网表文件。从原理图生成的网表文件通常缺少与各种元件对应的封装。因此，中间阶段是必要的。在该中间过程期间，执行具有封装的元件的关联。在 KiCad 中，CvPcb 用于创建此关联，并生成名为 '*.cmp' 的文件。CvPcb 还使用此信息更新网表文件。

CvPcb 还可以输出一个“填充文件” '*.stf'，它可以作为每个元件的 F2 字段返回注释到原理图文件中，从而节省了在每个原理图编辑通道中重新分配封装的任务。在 Eeschema 中，复制元件还将复制足迹分配并将参考标志设置为未分配以用于以后的自动增量注释。

Pcbnew 读取修改后的网表文件 '*.net'，如果存在，则读取 '*.cmp' 文件。如果在 Pcbnew 中直接更改了封装，则会自动更新 '*.cmp' 文件，从而无需再次运行 CvPcb。

请参阅 *KiCad* 工作流程部分中的“KiCad 入门”手册中的图，该部分说明了 KiCad 的工作流程以及如何由构成 KiCad 的不同软件工具获取和使用中间文件。

4.2 创建印刷电路板的程序

在 Eeschema 中创建原理图后：

- 使用 Eeschema 生成网表。
- 将网表文件中的每个元件分配给使用 Cvpcb 在印刷电路上使用的相应焊盘图案（通常称为封装）。
- 启动 Pcbnew 并阅读修改后的网表。这也将读取带封装选择的文件。

然后，Pcbnew 将自动加载所有必要的封装。现在可以手动或自动将封装放置在电路板上，并且可以布线。

4.3 更新印刷电路板的程序

如果修改了原理图（在生成印刷电路板之后），则必须重复以下步骤：

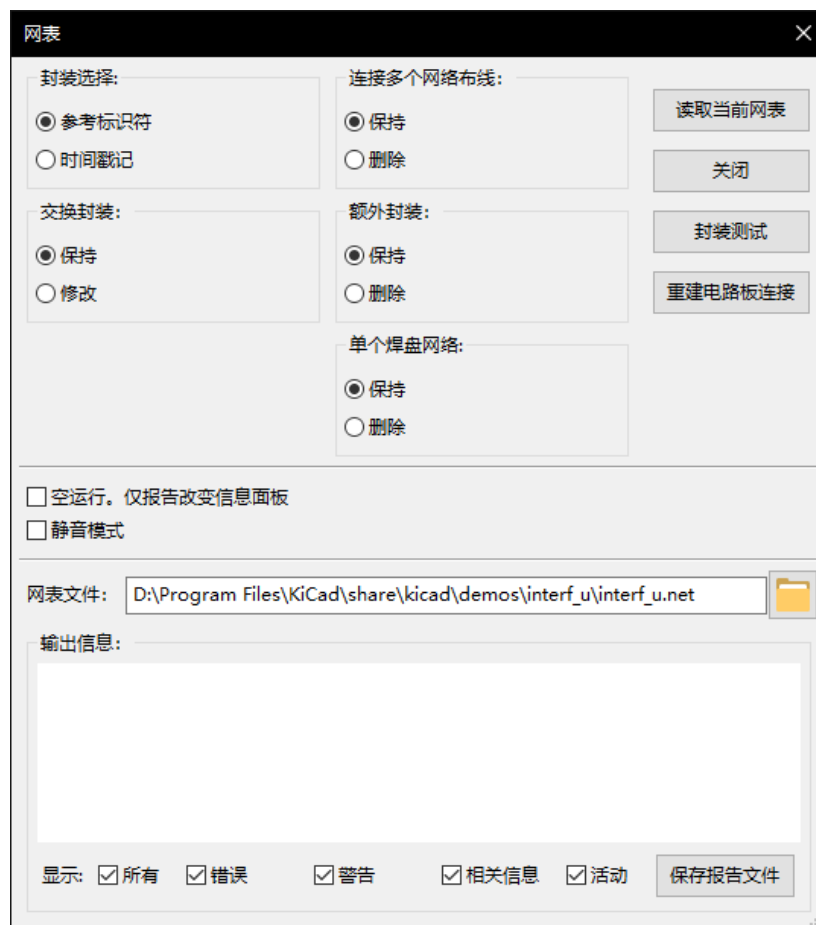
- 使用 Eeschema 生成新的网表文件。
- 如果对原理图的更改涉及新元件，则必须使用 Cvpcb 分配相应的封装。
- 启动 Pcbnew 并重新读取修改后的网表（这也将重新读取带有封装选择的文件）。

然后，Pcbnew 将自动加载任何新的占用空间，添加新连接并删除冗余连接。此过程称为前向注释，是制作和更新 PCB 时非常常见的过程。

4.4 读取网表文件 - 加载封装

4.4.1 对话框

可从图标图像访问：

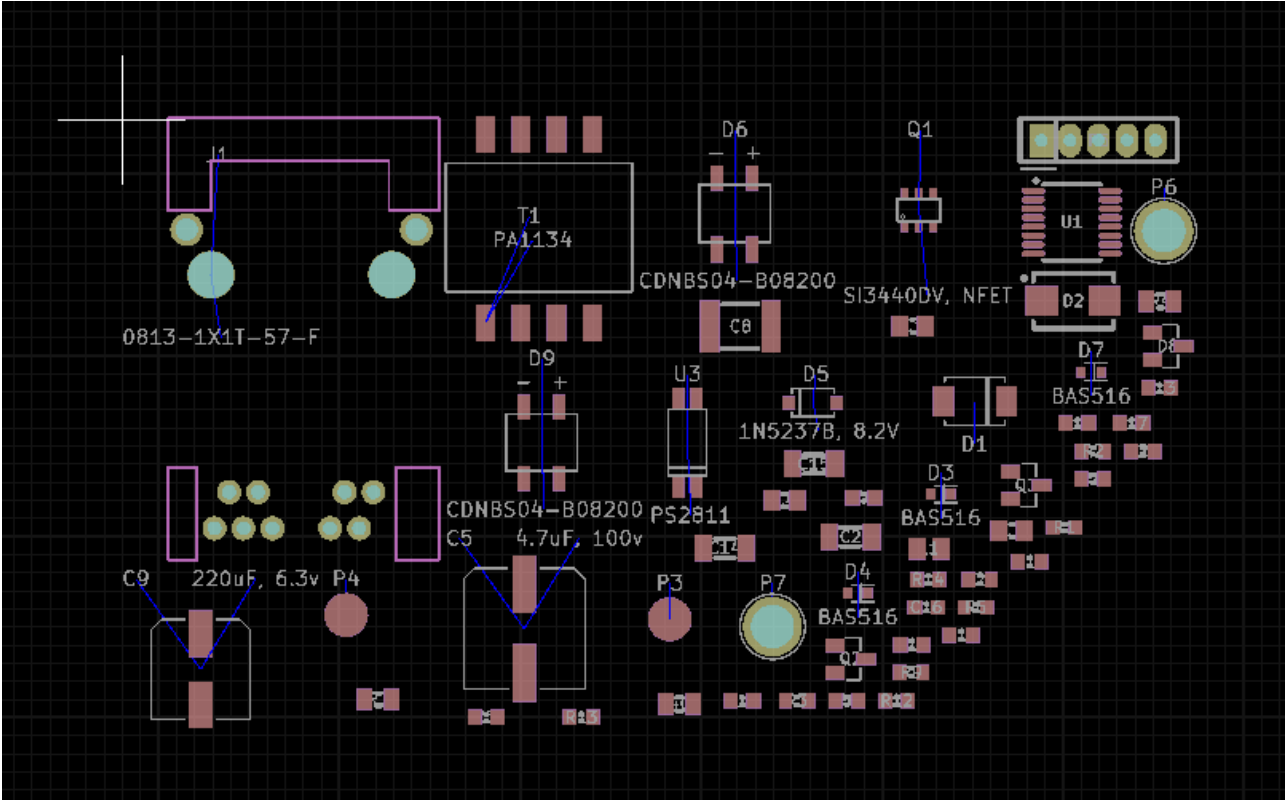


4.4.2 可用选项

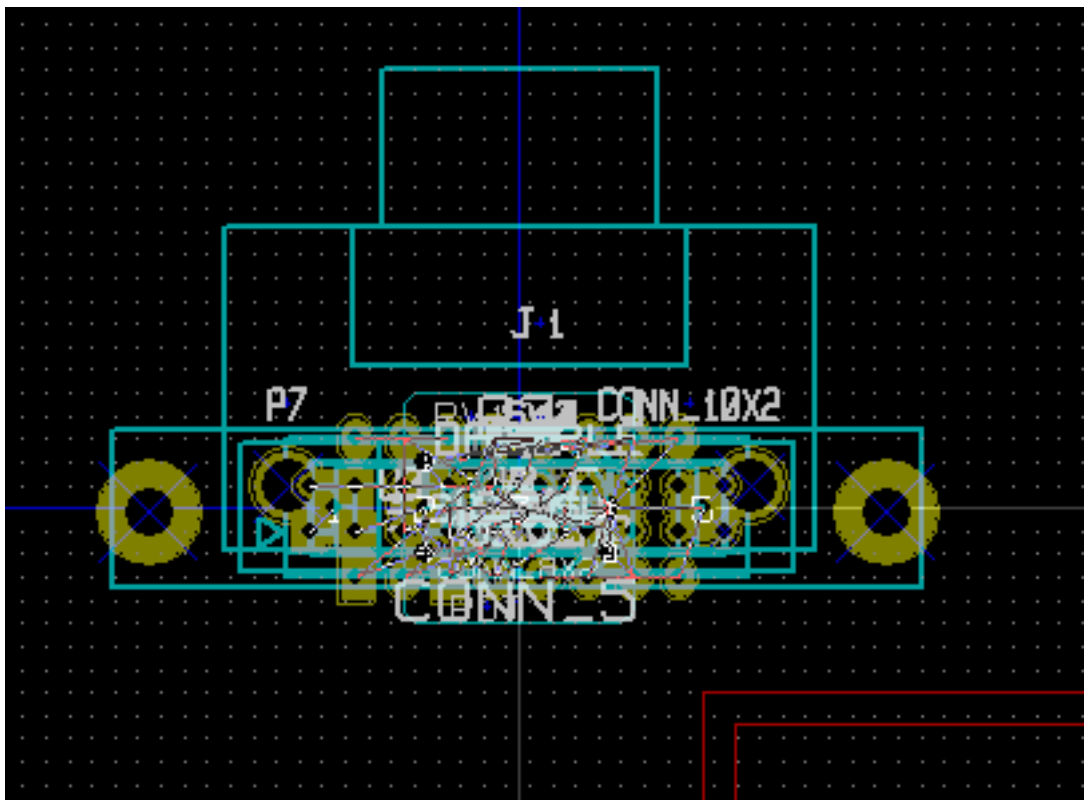
封装选择	链接上的元件和相应的封装：普通链接是参考（正常选项重新注释原理图后可以使用时间戳，如果是之前的话注释被销毁（特殊选项）
交换封装：	如果网表中的封装发生了变化：保持旧的封装或换到新的。
未连接的布线	保留所有现有布线，或删除错误布线
额外的封装	删除板上但不在网表中的封装。具有“已锁定”属性的封装不会被删除。
单焊盘网	删除单个焊盘网。

4.4.3 加载新的封装


使用 GAL 后端，当在网表文件中找到新的封装时，它们将被加载，展开，并准备好作为您想要的组放置。



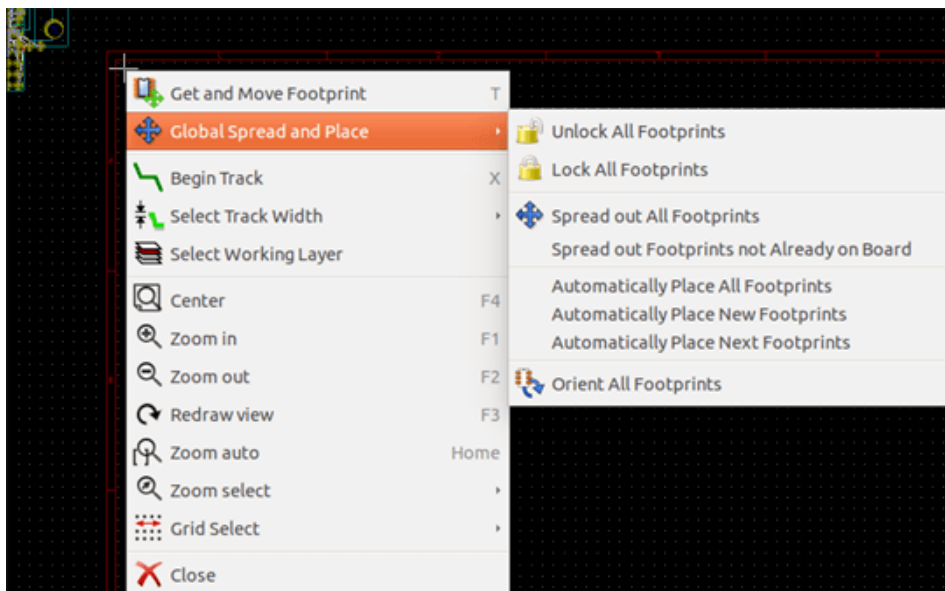
使用遗留后端，当在网表文件中找到新的封装时，它们将被自动加载并放置在坐标（0,0）处。



新的封装可以一个接一个地移动和排列。更好的方法是自动移动（取消堆叠）它们：

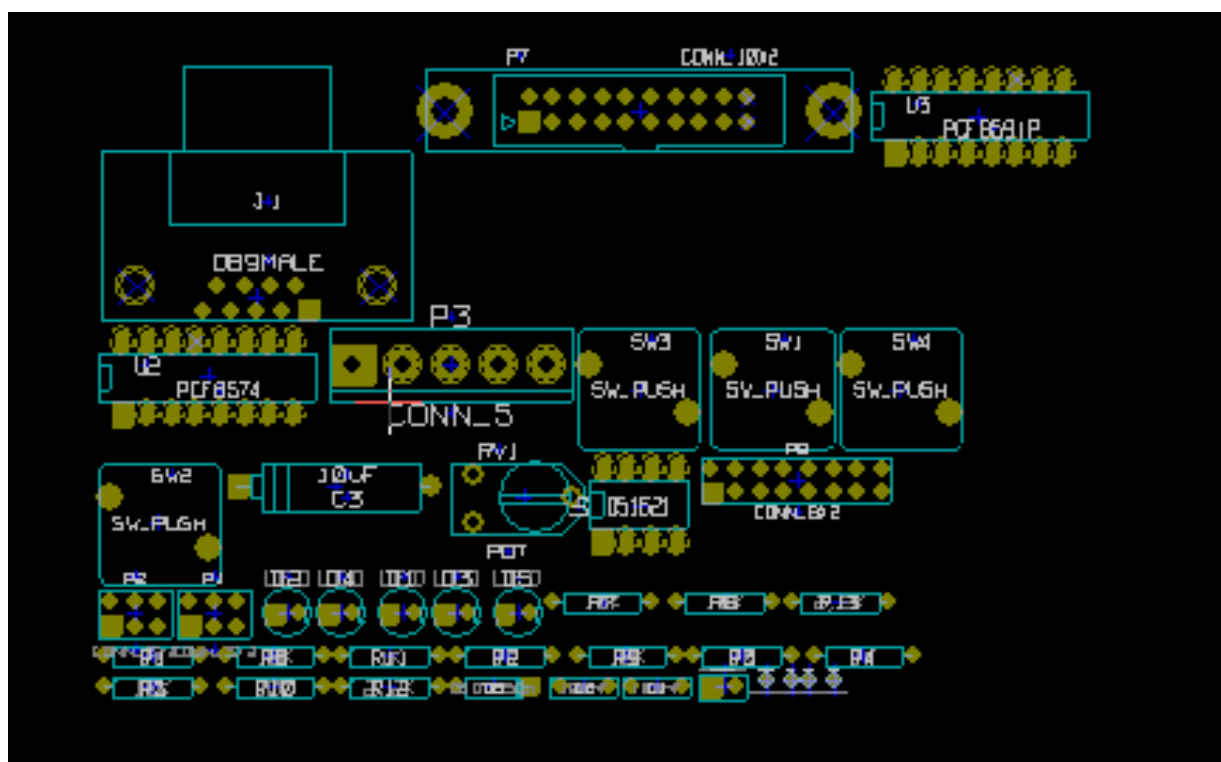
激活封装模式 ()

将鼠标光标移动到合适的（无元件）区域，然后单击右键：



- 如果已有包含现有封装的电路板，则自动放置新封装。
- 首次自动放置所有边框线（创建板时）。

以下屏幕截图显示了结果。



Chapter 5

层

5.1 简介

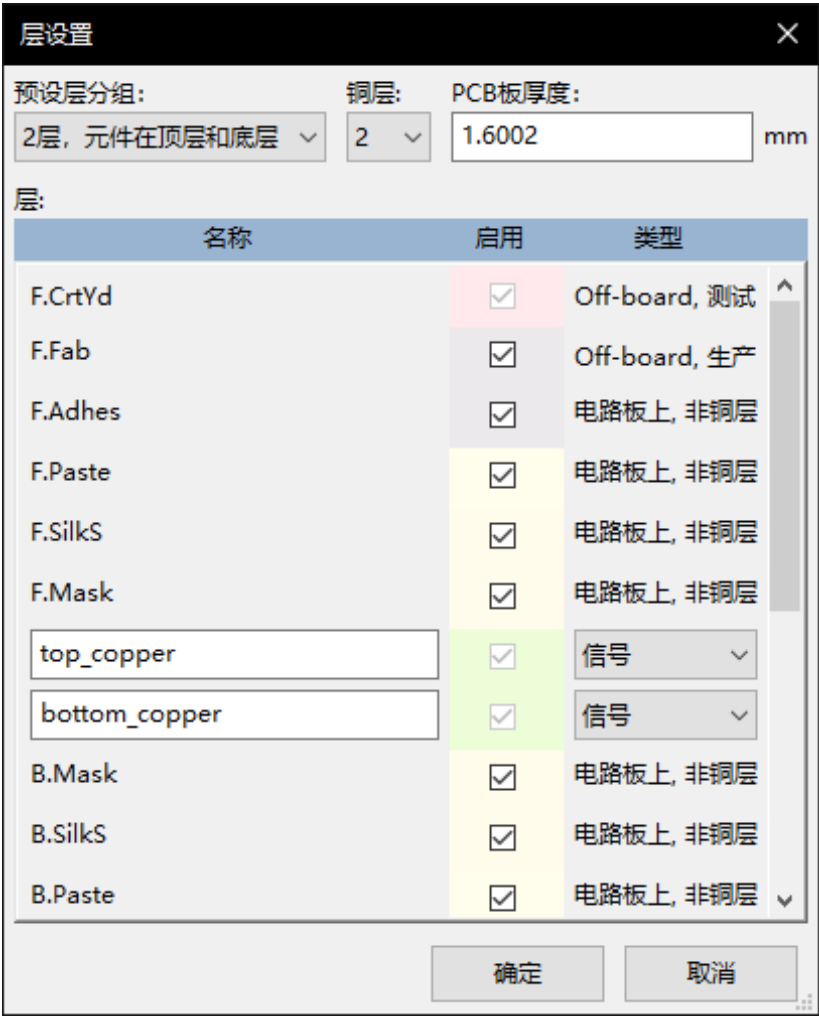
Pcbnew 可以使用 50 个不同的层：

- 1 到 32 个铜层用于布线。
- 14 个固定用途的技术层：
 - 12 对成对层（正面/背面）：**粘合剂**，**焊膏**，**丝印**，**阻焊层**，**封装**，**制造**
 - 2 个独立图层：**边缘切割**，**边距**
- 您可以以任何方式使用的 4 个辅助层：**注释**，**E.C.O. 1**，**E.C.O. 2**，**图纸**

5.2 设置图层

要从菜单栏中打开 **图层设置**，请选择 **设置** → **图层设置**。

在那里配置电路板厚度，铜层的数量，它们的名称和它们的功能。可以禁用未使用的技术层。

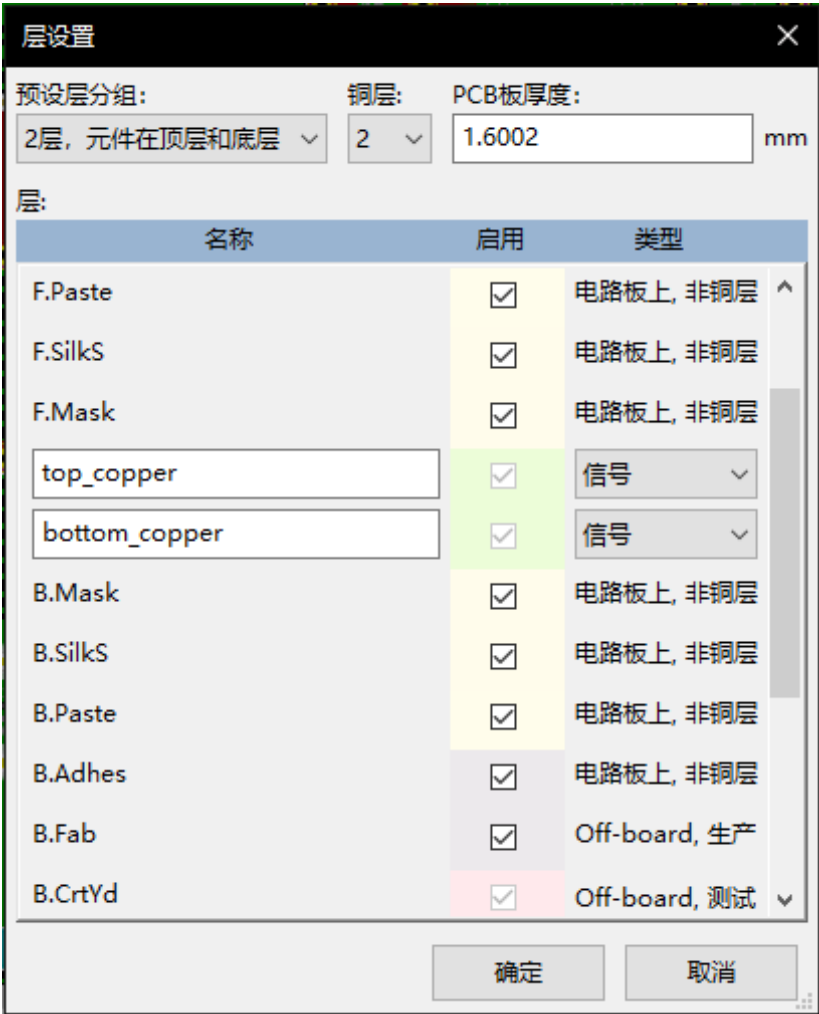


5.3 层描述

5.3.1 铜层

铜层是用于放置和重新布线的常用工作层。层编号从 0 开始（第一个铜层，在前面），结束在 31（后面）。由于元件不能放在 * 内层 *（编号 1 到 30）中，因此只有 0 和 31 层是 * 元件层 *。

任何铜层的名称都是可编辑的。铜层具有在使用外部布线 *Freerouter* 时很有用的功能属性。默认图层名称的示例是 **F.Cu** 和 **In0**，用于图层编号 0。



5.3.2 配对技术层

12 个技术层成对出现：一个用于前面，一个用于后面。你可以用“F”识别它们。或“B”名字前缀。构成这些层之一的覆盖区（焊盘，绘图，文本）的元素将自动镜像，并在翻转覆盖区时移动到补充层。

配对的技术层是：

粘合剂层 (F.Adhes 和 B.Adhes)

这些用于粘合剂的应用，通常在波峰焊之前将 SMD 元件粘到电路板上。

焊膏层 (F.Paste 和 B.Paste)

用于生产锡膏层，通常在回流焊接之前，将焊膏放置在表面贴装元件的焊盘上。通常只有表面贴装焊盘占据这些层。

丝印层 (F.SilkS 和 B.SilkS)

它们是元件图纸出现的层。这就是你绘制元件极性，第一个引脚指示器，安装参考，……

阻焊层 (F.Mask 和 B.Mask)

这些定义了阻焊层。所有焊盘应出现在其中一个层（SMT）或两者（用于通孔）上，以防止绿油覆盖焊盘。

封装层 (F.CrtYd 和 B.CrtYd)

用于显示元件在 PCB 上物理占用的空间大小。

制造层 (F.Fab 和 B.Fab)

制造层主要用于记录目的，以将信息传达给例如 PCB 制造商或组装厂。

5.3.3 独立技术层

边缘切割 (Edge.Cuts)

该层保留用于绘制电路板边框。放置在该图层上的任何元素（图形，文本…）都出现在所有其他图层上。仅使用此图层绘制板边框。

余量 (Margin)

板的边缘后退边框 (?)。

5.3.4 一般用途的图层

这些层可用于任何用途。它们可用于文本，例如装配或布线说明或结构图，用于创建用于装配或加工的文件。他们的名字是：

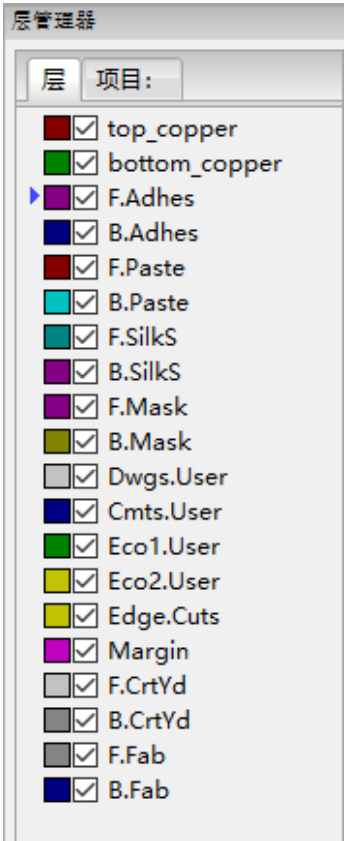
- 注释
- E.C.O. 1
- E.C.O. 2
- 图纸

5.4 选择活动层

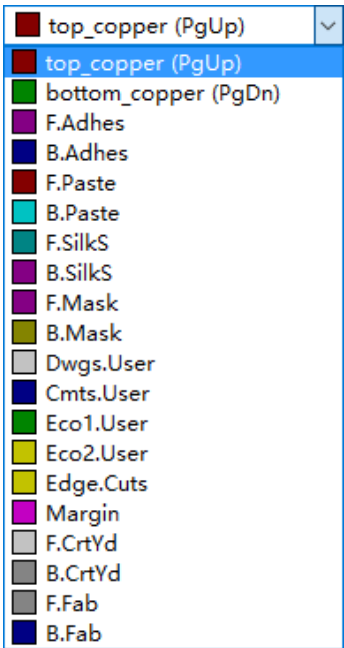
可以通过以下几种方式选择活动工作层：

- 使用右侧工具栏（图层管理器）。
- 使用上方工具栏。
- 使用弹出窗口（用鼠标右键激活）。
- 使用 + 和 - 键（仅适用于铜层）。
- 通过热键。

5.4.1 使用图层管理器进行选择



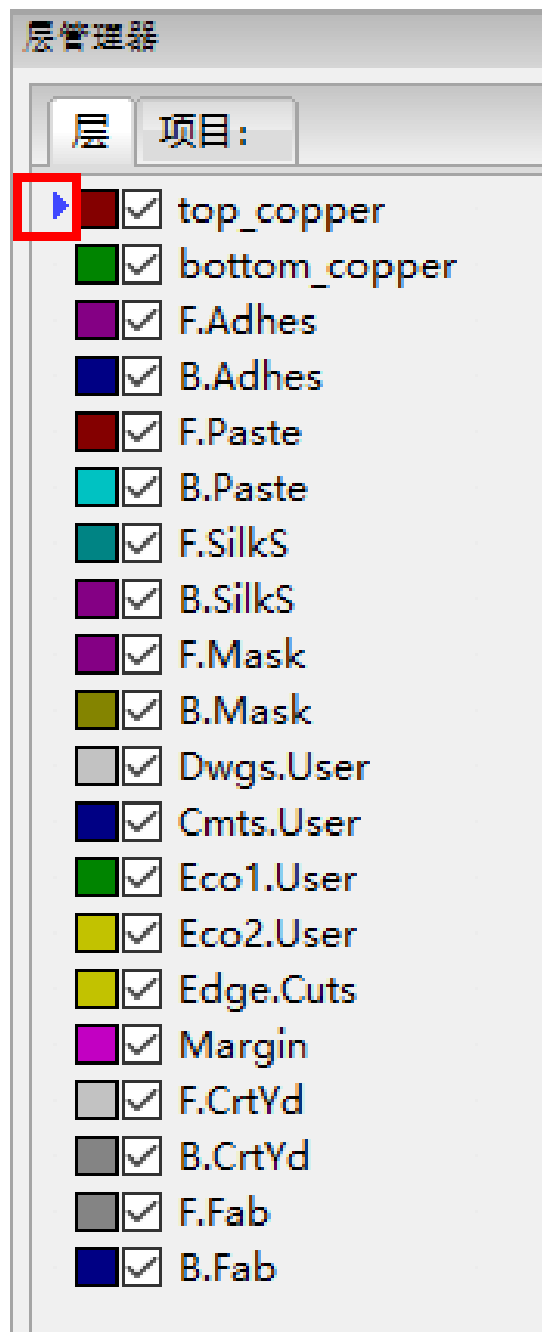
5.4.2 使用上方工具栏选择



这直接选择工作层。

显示用于选择工作层的热键。

5.4.3 使用弹出窗口进行选择

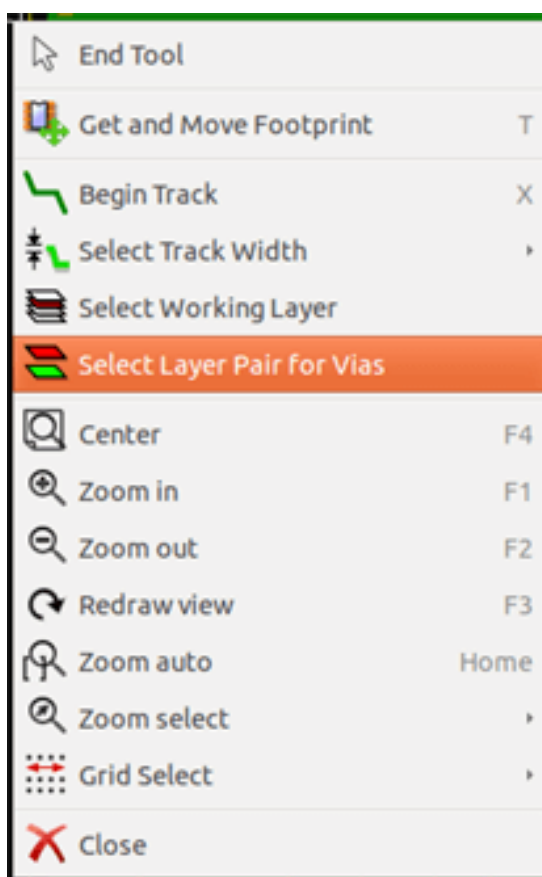


弹出窗口打开一个菜单窗口，为工作层提供选择。

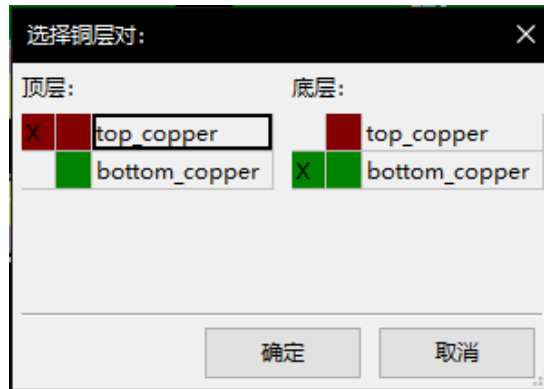


5.5 选择过孔层

如果在右侧工具栏上选择了 **添加布线**和**过孔**图标，则弹出窗口提供更改用于过孔的图层对的选项：



此选项将打开一个菜单窗口，该窗口提供用于过孔的层的选择。



当放置过孔时，工作（有源）层自动切换到用于过孔的层对的交替层（除非在添加过孔时保持“移位”）。
也可以通过热键切换到另一个活动层，如果正在进行跟踪，则会插入一个过孔。

5.6 使用高对比度模式

激活工具（在左侧工具栏中）时进入此模式：



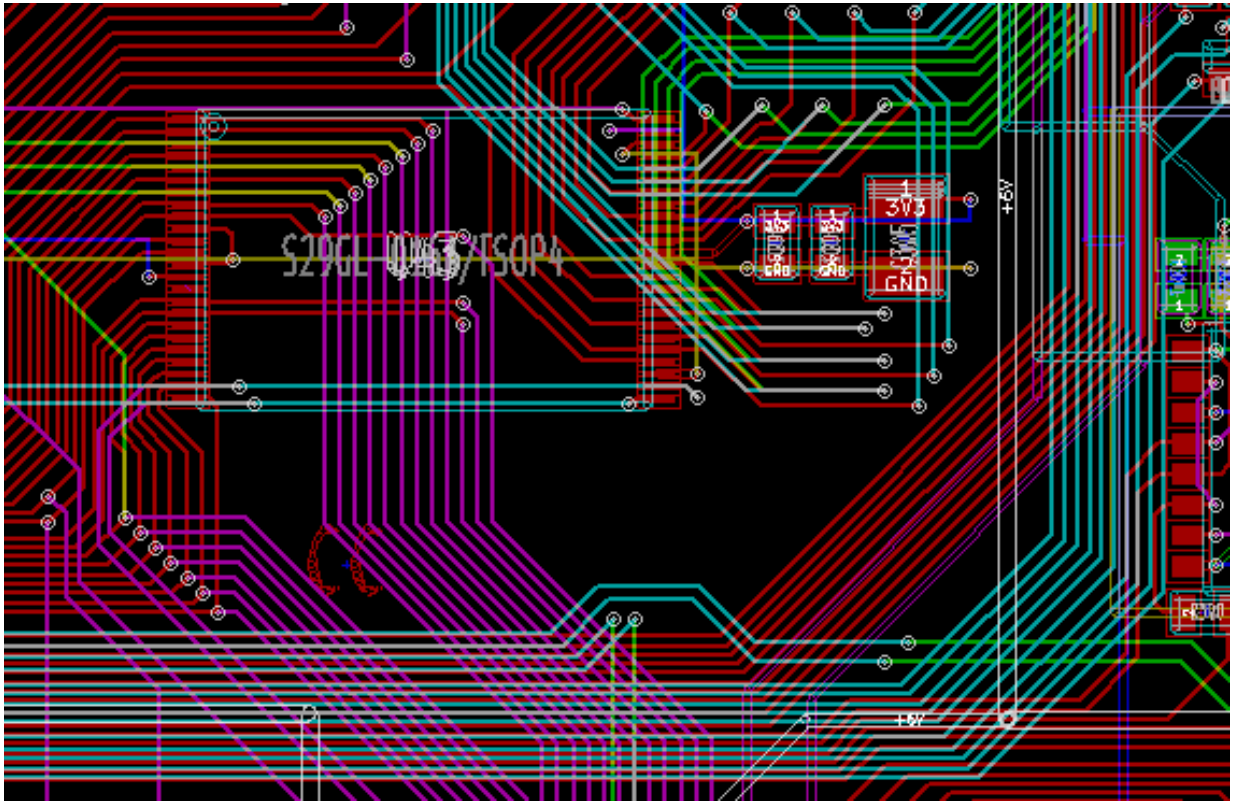
使用此模式时，活动图层的显示方式与普通模式相同，但所有其他图层均以灰色显示。

有两个有用的案例：

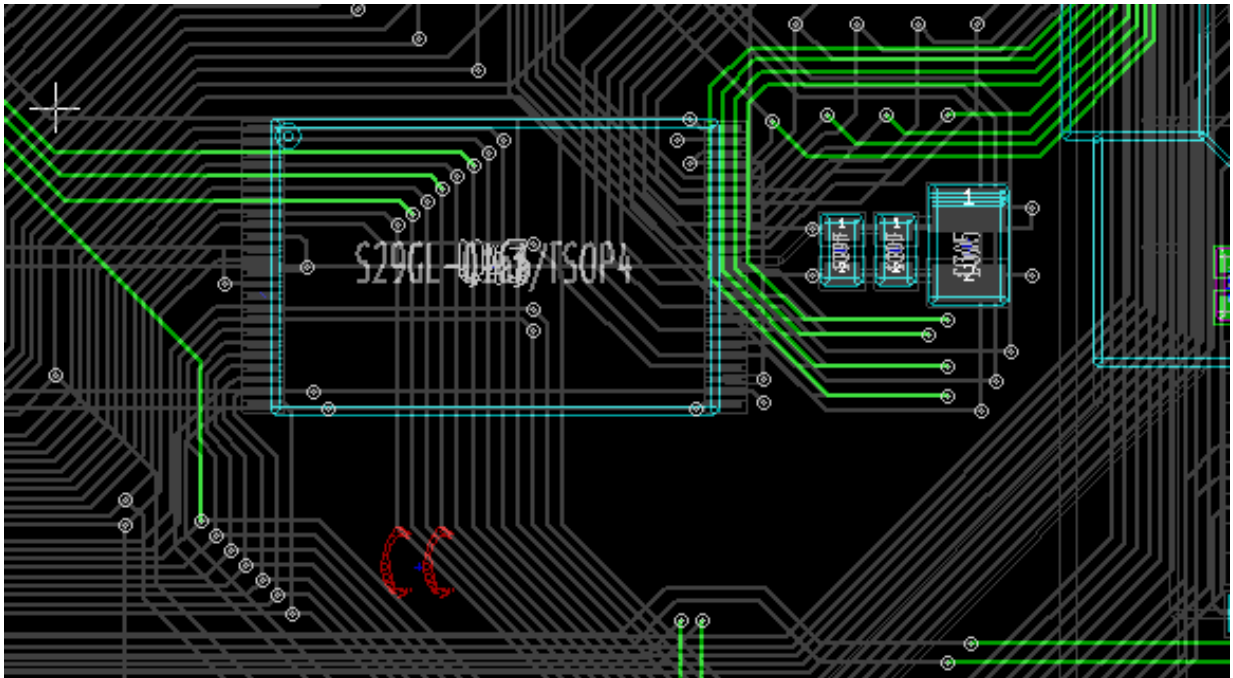
5.6.1 高对比度模式下的铜层

当电路板使用四层以上时，此选项可以更容易地看到活动铜层：

正常模式（背面铜层有效）：



高对比度模式（背面铜层有效）：

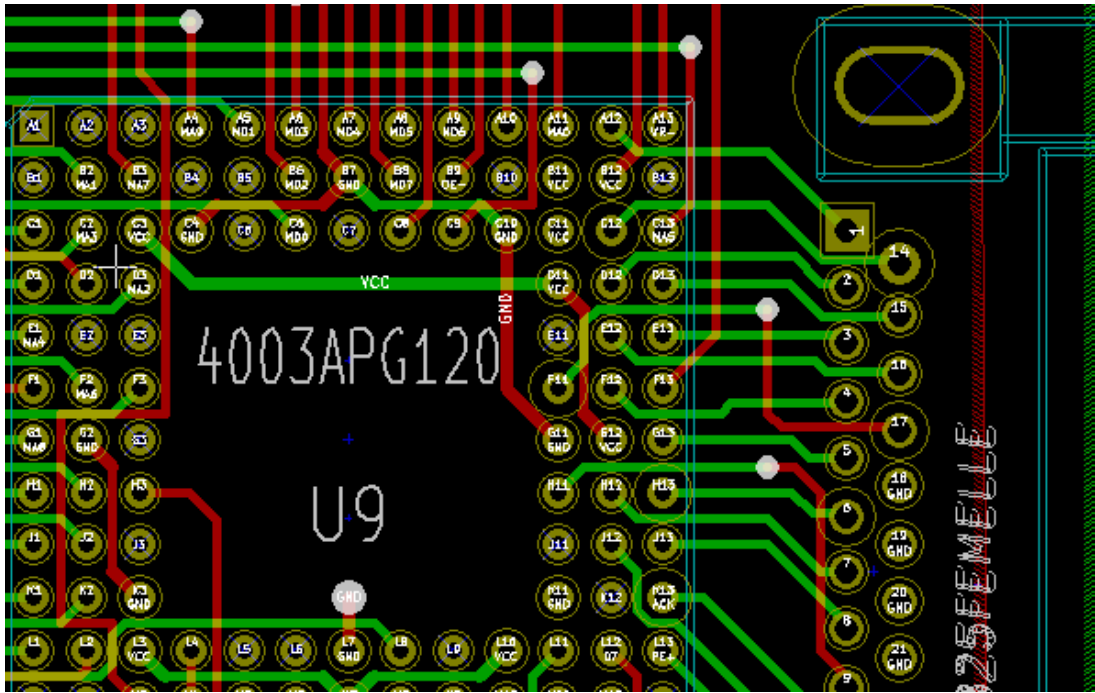


5.6.2 技术层

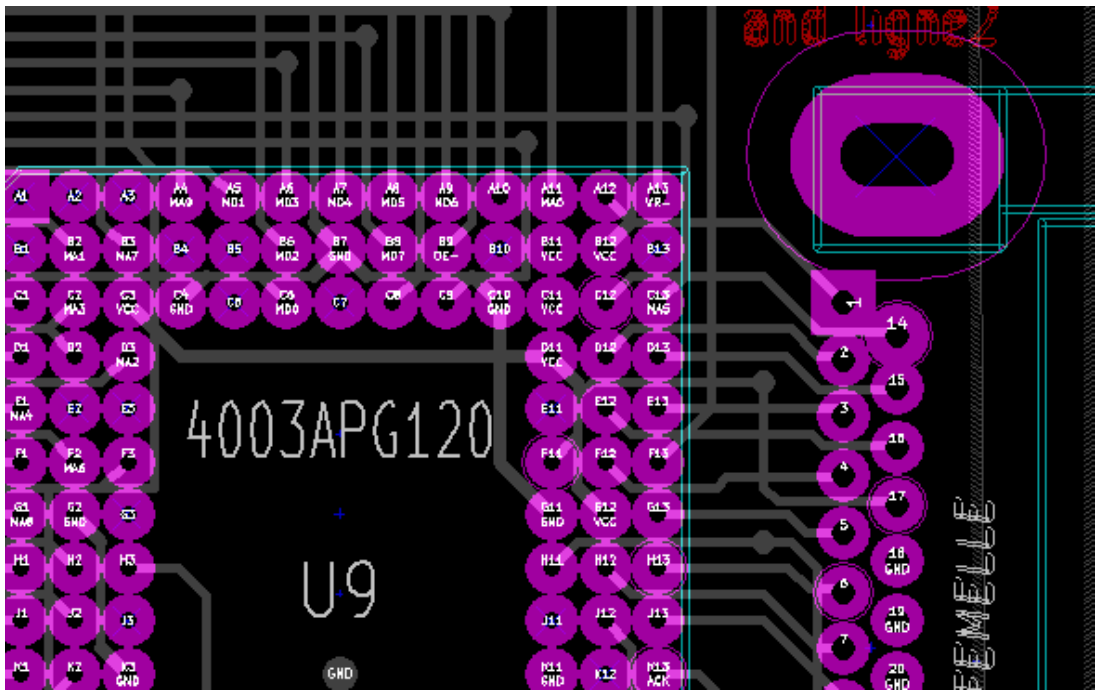
另一种情况是需要检查通常不显示的焊膏层和阻焊层。

如果此模式处于活动状态，则会显示焊盘上的遮罩。

正常模式（前侧阻焊层有效）：



高对比度模式（前侧阻焊层有效）：



Chapter 6

创建和修改电路板

6.1 创建一个板

6.1.1 绘制板边框

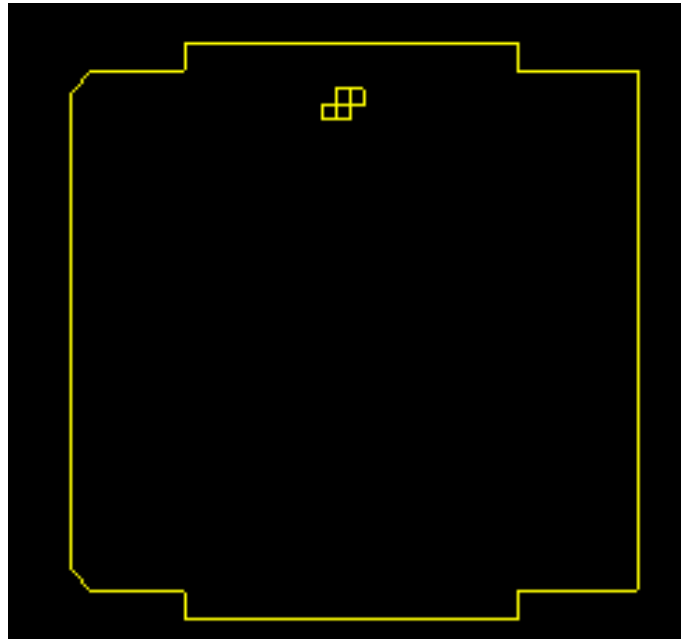
通常首先定义电路板的边框是个好主意。边框绘制为一系列线段。选择“Edge.Cuts”作为活动图层，并使用“添加图形线或多边形”工具跟踪边缘，单击每个顶点的位置，然后双击以完成边框。板通常具有非常精确的尺寸，因此在跟踪边框时可能需要使用显示的光标坐标。请记住，可以使用空格键随时将相对坐标归零，并且还可以使用“Ctrl-U”切换显示单位。相对坐标可以绘制非常精确的尺寸。可以绘制圆形（或弧形）边框：

1. 选择“添加图形圆”或“添加图形圆弧”工具
2. 单击以固定圆心
3. 通过移动鼠标调整半径
4. 再次单击完成。

Note

可以在“参数”菜单中调整边框的宽度（建议宽度 = 150in 1/10 mils）或通过“选项”调整，但除非图形以边框模式以外的方式显示，否则不会显示。

结果边框可能如下所示：



6.1.2 使用 DXF 绘图作为电路板边框

作为直接在 Pcbnew 中绘制板边框的替代方法，也可以从 DXF 图纸导入边框。

使用此功能可以实现比 Pcbnew 绘图功能更复杂的板形状。

例如，机械 CAD 包可用于定义适合特定外壳的板形状。

6.1.2.1 准备 DXF 绘图以导入 KiCad

KiCad 中的 **DXF** 导入功能不支持 **POLYLINES** 和 **ELLIPSIS** 等 DXF 功能，使用这些功能的 DXF 文件需要一些转换步骤来准备导入。

像 LibreCAD 这样的软件包可用于此转换。

作为第一步，任何 **POLYLINES** 都需要拆分（爆炸）成原始的简单形状。在 LibreCAD 中，使用以下步骤：

1. 打开 DXF 文件的副本。
2. 选择板形状（选定的形状用虚线显示）。
3. 在 **修改** 菜单中，选择 **炸开**。
4. 按 ENTER 键。

作为下一步，像 **ELLIPSIS** 这样的复杂曲线需要在“近似”所需形状的小线段中分解。当导出 DXF 文件或以较旧的 **DXF R12** 文件格式保存时，会自动发生这种情况（因为 R12 格式不支持复杂的曲线形状，CAD 应用程序会将这些形状转换为线段。某些 CAD 应用程序允许配置数字或所用线段的长度）。在 LibreCAD 中，段长度通常足够小，可用于板形。

在 LibreCAD 中，使用以下步骤导出为 **DXF R12** 文件格式：

1. 在 **文件** 菜单中，使用 **另存为...**

2. 在 **将绘图另存为**对话框中，对话框底部附近有一个 **保存类型**：选择。选择选项 * 绘制交换 DXF R12 * 。
3. (可选) 在 **文件名**：字段中输入文件名。
4. 点击 **保存**

您的 DXF 文件现已准备好导入 KiCad。

6.1.2.2 将 DXF 文件导入 KiCad

以下步骤描述了将准备好的 DXF 文件作为板形状导入 KiCad。请注意，导入行为略有不同，具体取决于使用的“画布”。

使用“默认”画布模式：

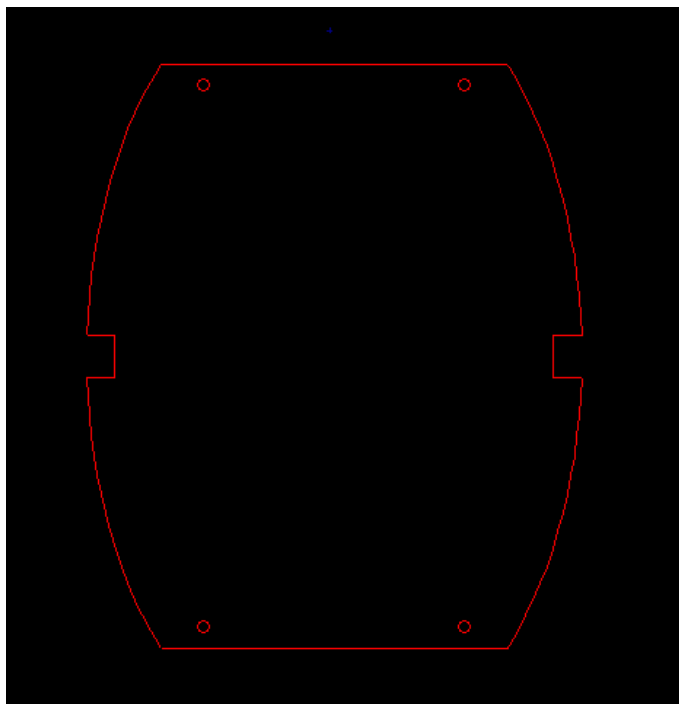
1. 在 **文件**菜单中，选择 **导入**，然后选择 **DXF 文件**选项。
2. 在 **导入 DXF 文件**对话框中，使用“浏览”选择要导入的准备好的 DXF 文件。
3. 在“放置 DXF 原点 (0,0) 点：”选项中，选择 DXF 原点相对于板坐标的位置 (KiCad 板在左上角有 (0,0))。对于“用户定义位置”，在“X 位置”和“Y 位置”字段中输入坐标。
4. 在“层”选择中，选择导入的板层。电路板边框需要 * Edge.Cuts * 。
5. 单击“确定”。

使用“OpenGL”或“Cairo”画布模式：


1. 在 **文件**菜单中，选择 **导入**，然后选择 **DXF 文件**选项。
2. 在 **导入 DXF 文件**对话框中，使用“浏览”选择要导入的准备好的 DXF 文件。
3. 在此模式下，将忽略‘放置 DXF 原点 (0,0) 点：’选项设置。
4. 在“层”选择中，选择导入的板层。电路板边框需要 * Edge.Cuts * 。
5. 单击“确定”。
6. 现在，形状已附加到光标上，可以在电路板区域周围移动。
7. 单击以“放下”板上的形状。

6.1.2.3 示例导入 DXF 形状

下面是一个 DXF 导入示例，其中一个板具有几个由多个短线段近似的椭圆段：

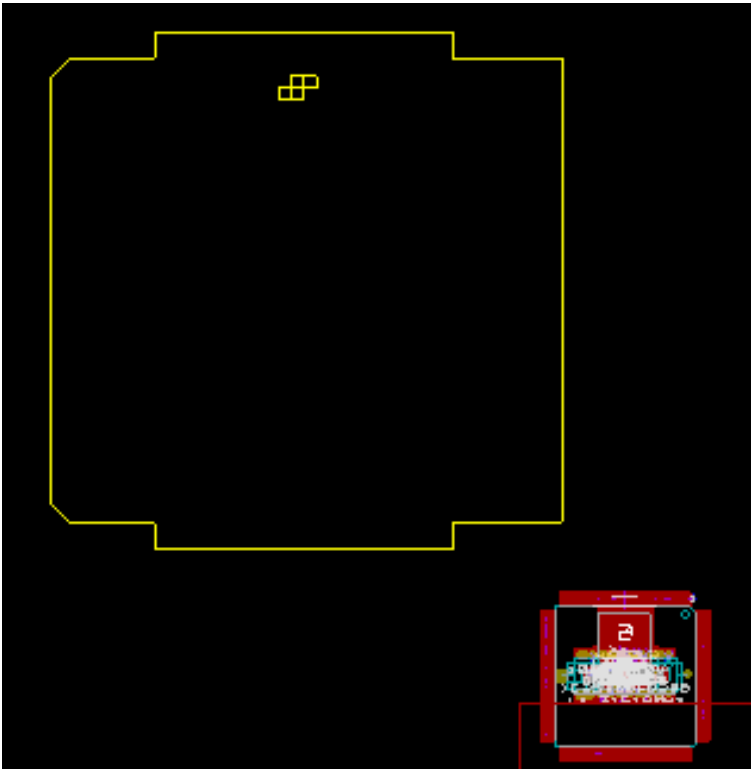


6.1.3 读取原理图生成的网表

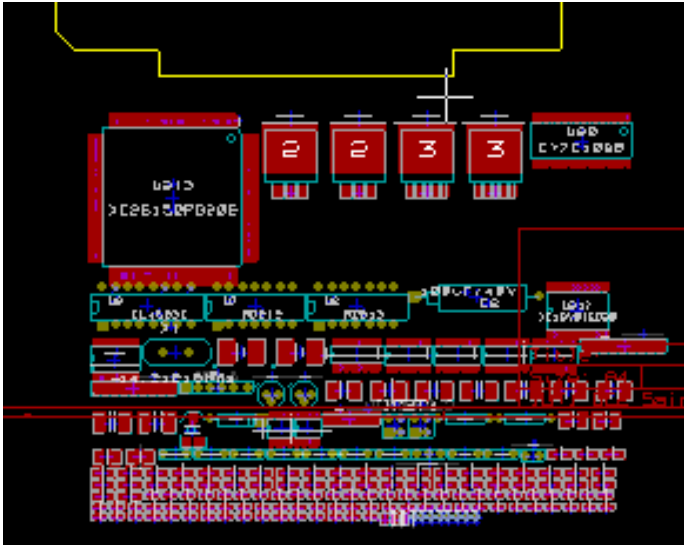
激活图像： 图标以显示网表对话框：



如果窗口标题中网表的名称（路径）不正确，请使用“选择”按钮浏览到所需的网表。然后‘读’网表。任何尚未加载的足迹都会出现，叠加在一起（我们将在下面看到如何自动移动它们）。



如果没有放置封装，则所有封装将出现在同一位置的板上，使其难以识别。可以自动排列它们（使用通过鼠标右键访问的“全局传播和放置”命令）。以下是这种自动安排的结果：



Note

如果通过在 CvPcb 中用新的封装替换现有封装（例如将 1/8W 电阻更改为 1/2W）来修改电路板，则必须在 Pcbnew 加载替换封装之前删除现有元件。但是，如果要用现有的占用空间替换占用空间，则使用通过在相关覆盖区上单击鼠标右键访问的占用空间对话框更容易。

6.2 校正一块板

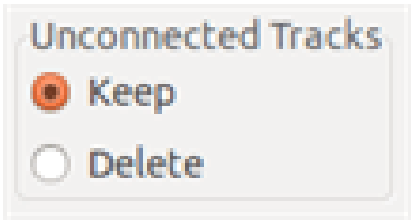
在原理图中相应的变化之后，经常需要校正电路板。

6.2.1 要遵循的步骤

1. 从修改后的原理图中创建新的网表。
2. 如果添加了新元件，请将这些元件链接到 CvPcb 中相应的封装。
3. 阅读 Pcbnew 中的新网表。

6.2.2 删除错误的布线

Pcbnew 能够自动删除因修改而变得不正确的布线。为此，请检查网表对话框的“未连接的布线踪”框中的“删除”选项：

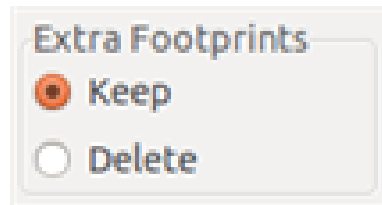


但是，手动修改这些布线通常更快（DRC 功能允许识别）。

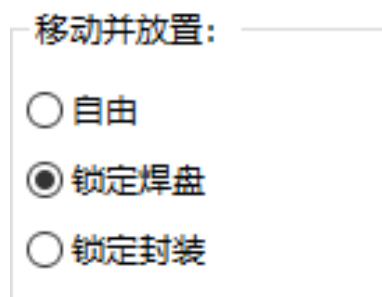
6.2.3 已删除的元件

Pcbnew 可以删除与已从原理图中删除的元件相对应的封装。这是可选的。

这是必要的，因为通常会在 PCB 中添加封装（例如，用于固定螺钉的孔），这些封装从未出现在原理图中。

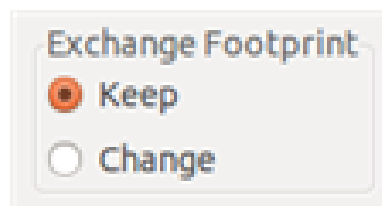


如果选中“额外封装”选项，则将删除与网表中未找到的元件对应的封装，除非他们选择“已锁定”激活。为“机械”封装激活此选项是个好主意：



6.2.4 修改后的封装

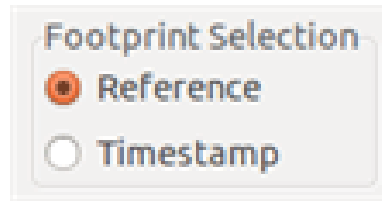
如果在网表中修改了封装（使用 CvPcb），但已放置封装，则 Pcbnew 不会修改封装，除非检查网表对话框的“更改封装”框的相应选项：



更改封装（例如，更换具有不同尺寸的电容器）可以通过编辑封装直接实现。

6.2.5 高级选项 - 使用时间戳选择

有时原理图的符号会发生变化，电路中没有任何重大变化（这会涉及参考文献 - 如 R5, U4 ...）。因此 PCB 保持不变（除了可能用于丝网印刷标记）。然而，在内部，元件和封装由它们的参考表示。在这种情况下，可以在重新读取网表之前选择网表对话框的“时间戳”选项：



使用此选项，Pcbnew 不再通过其引用来标识封装，而是通过其时间戳来标识封装。时间戳由 Eeschema 自动生成（它是元件放置在原理图中的时间和日期）。

**Warning**

使用此选项时应特别小心（首先保存文件!）。这是因为在包含多个部件的元件的情况下该技术是复杂的（例如，7400 具有 4 个部件和一个壳体）。在这种情况下，时间戳不是唯一定义的（对于 7400，最多可以有四个 - 每个部分一个）。然而，时间戳选项通常可以解决重新注释问题。

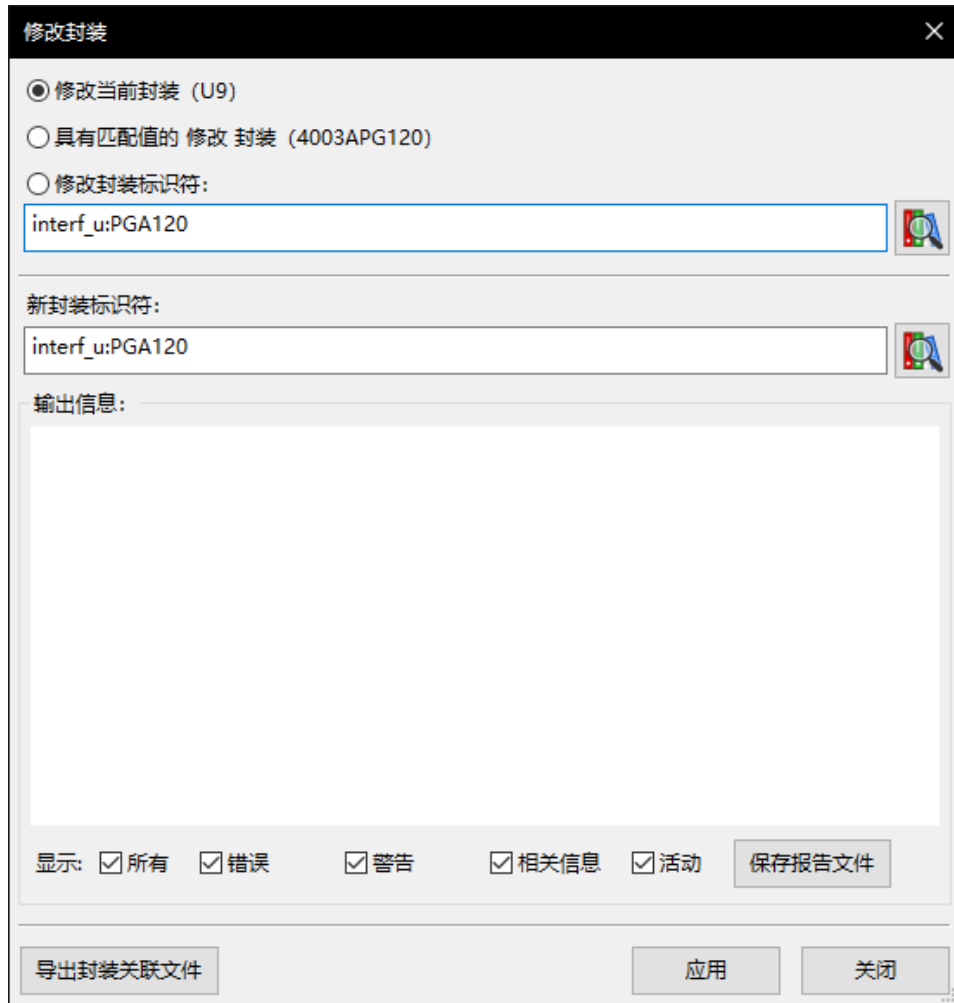
6.3 直接改变已经放置在板上的封装

将封装（或一些相同的封装）更改为另一个封装是非常有用的，并且非常简单：

1. 单击封装以打开“编辑”对话框。
2. 激活更改封装。



变更封装的选项:




必须选择新的封装名称并使用：

- 更改 ‘xx’ 的封装为当前封装
- 更改封装 ‘yy’ 为所有封装（如当前封装）。
- 更改当前所有封装的具有相同值的封装封装，仅限于具有相同价值的元件。
- 更新电路板的所有封装，以便重新载入板上所有封装。

Chapter 7

封装放置

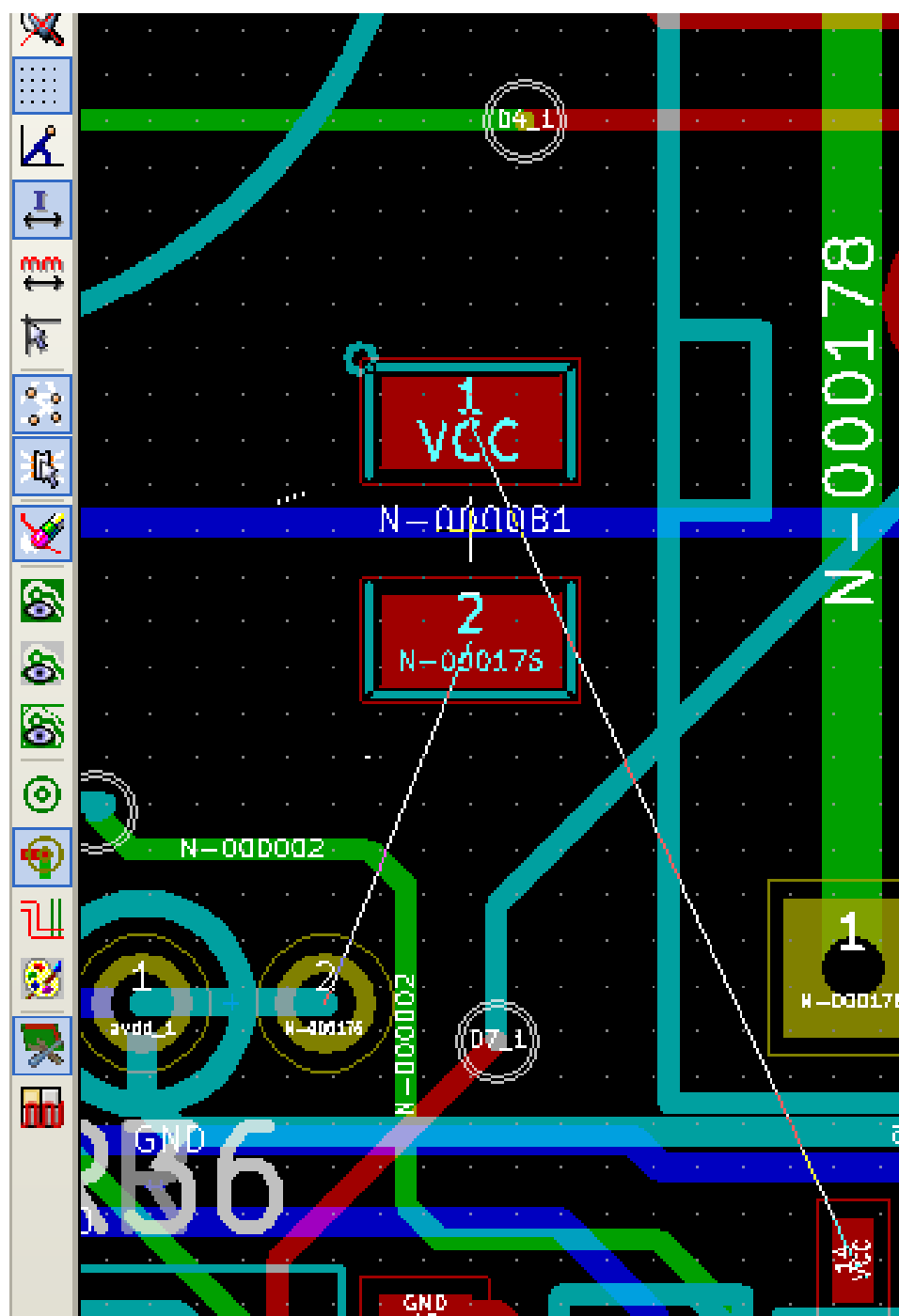
7.1 协助安置

在移动封装的飞线同时，可以显示封装（网络连接）以帮助放置。要启用此功能，必须激活左侧工具栏的图标 。

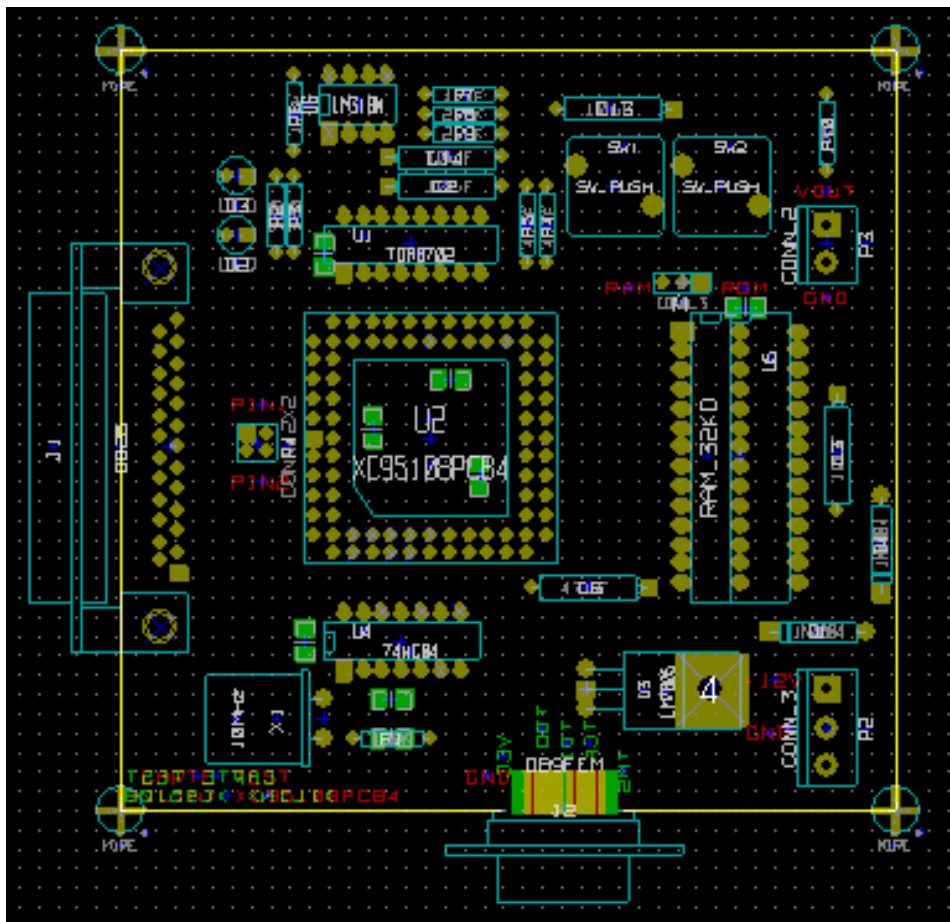
7.2 手动放置

用鼠标右键选择覆盖区，然后从菜单中选择“移动”命令。将足迹移动到所需位置，然后使用鼠标左键将其放置。如果需要，也可以旋转，反转或编辑所选足迹。从菜单中选择取消（或按 Esc 键）中止。

在这里，您可以看到移动过程中封装的飞线显示：




放置所有封装后的电路可能如下所示:



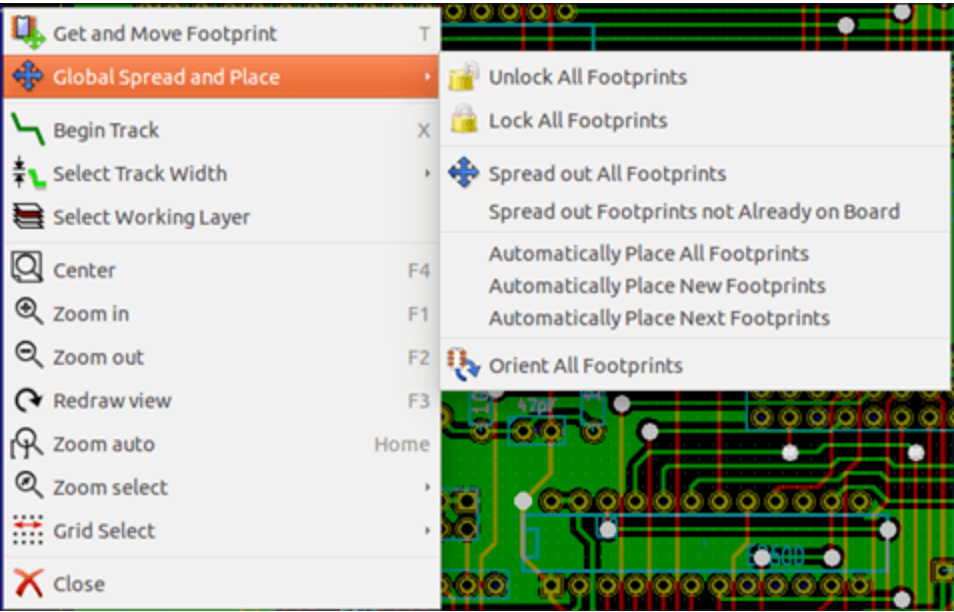
7.3 自动封装分布

一般来说，只有在没有“固定”的情况下才能移动封装。在封装模式下，或通过编辑封装菜单，可以从弹出窗口（在足迹上单击鼠标右键）打开和关闭此属性。

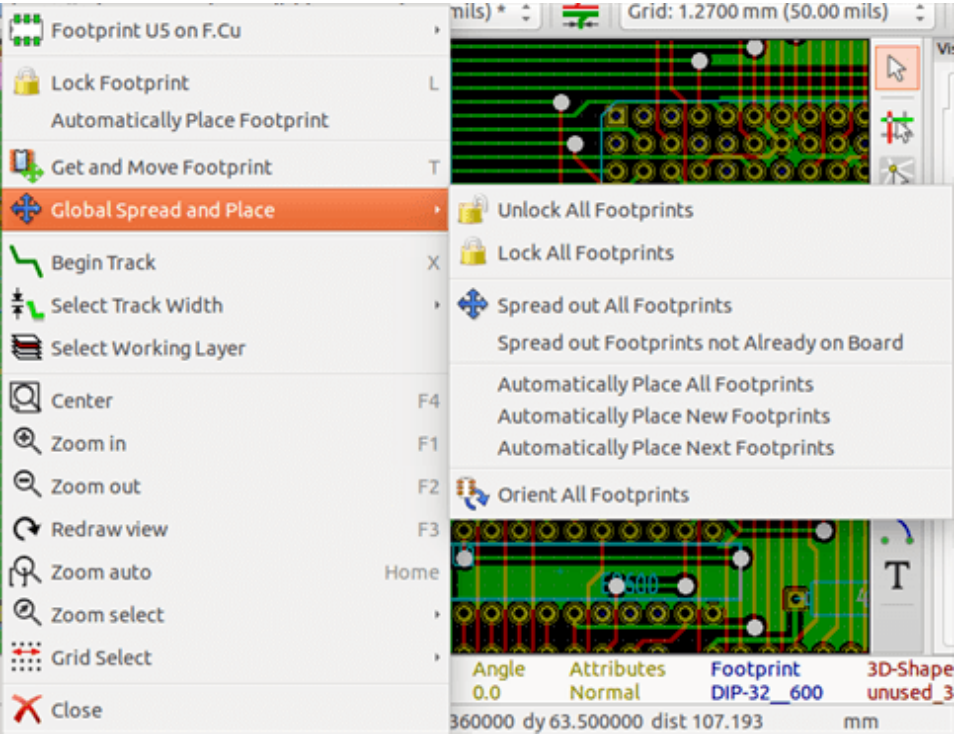
如上一章所述，在读取网表期间加载的新封装似乎堆积在板上的单个位置。Pcbnew 允许自动分配封装，使手动选择和放置更容易。

- 选择上方工具栏上的“封装模式”选项（图标 ）。
- 通过鼠标右键激活的弹出窗口变为：

如果光标下有封装：



如果光标下没有任何内容：



在这两种情况下，都可以使用以下命令：

- **展开所有封装**允许自动分配所有封装没有固定。这通常在首次阅读后使用网表。
- **分散不在板上的封装**允许自动分配尚未放置的封装在 PCB 边框内。此命令需要边框板的目的是确定哪些封装可以自动分发。

7.4 自动放置封装

7.4.1 自动放置的特点


自动放置功能允许将封装放置在电路板的两个面上（但是，在铜层上切换封装不是自动的）。

它还寻求封装的最佳方向（0 度，90 度，-90 度，180 度）。放置是根据优化算法进行的，该算法寻求最小化飞线的长度，并且寻求在具有许多焊盘的较大封装之间创建空间。优化放置顺序，以便最初将这些较大的封装放置在许多焊盘上。

7.4.2 准备

因此，Pcbnew 可以自动放置封装，但是有必要指导这种放置，因为没有软件可以猜出用户想要实现的目标。

在进行自动放置之前，必须：

- 创建电路板的边框（它可能很复杂，但如果表格不是矩形，则必须关闭）。
- 手动放置其位置的元件（连接器，夹孔等）。
- 同样，某些 SMD 封装和关键元件（例如大尺寸）必须位于电路板上的特定侧面或位置，这必须手动完成。
- 完成任何手动放置后，这些封装必须“固定”以防止它们被移动。选择封装模式图标  右键单击封装并在弹出菜单中选择“修复封装”。这也可以通过编辑/封装弹出菜单完成。
- 然后可以执行自动放置。选择足迹模式图标后，右键单击并选择全局移动和放置 - 然后选择自动放置所有封装。

在自动放置期间，如果需要，Pcbnew 可以优化封装的方向。但是，只有在封装被授权的情况下才会尝试旋转（请参阅编辑封装选项）。

通常，电阻器和非极化电容器被授权旋转 180 度。一些封装（例如小晶体管）可以被授权旋转 +/- 90 度和 180 度。

对于每个封装，一个滑块授权 90 度旋转，第二个滑块授权 180 度旋转。设置为 0 可防止旋转，设置为 10 可对其进行授权，中间值表示优先/反对旋转。

一旦将封装放置在板上，就可以通过编辑封装来完成轮换授权。但是，最好将所需选项设置为库中的封装，因为这些设置将在每次使用封装时继承。

7.4.3 交互式自动放置

在自动放置期间可能需要停止（按 Esc 键）并手动重新定位放置。使用命令自动放置下一个封装将从停止的位置重新启动自动放置。

命令自动放置新的封装允许自动放置尚未放置在 PCB 边框内的封装。即使它们没有“固定”，它也不会移动 PCB 边框内的那些。

命令自动放置封装使得可以在鼠标指向的放置上执行自动放置，即使其“固定”属性处于活动状态。

7.4.4 附加说明

Pcbnew 通过尊重板边框的形状自动确定封装的可能放置区域，板形状边框不一定是矩形（可以是圆形，或者有切口等）。

如果电路板不是矩形，则必须关闭边框，以便 Pcbnew 可以确定边框内部和外部的内容。同样，如果有内部切口，则必须关闭它们的边框。

Pcbnew 使用板的边框计算封装的可能放置区域，然后依次将每个封装传递到该区域，以确定放置它的最佳位置。

Chapter 8

设置布线参数

8.1 当前设置

8.1.1 访问主对话框

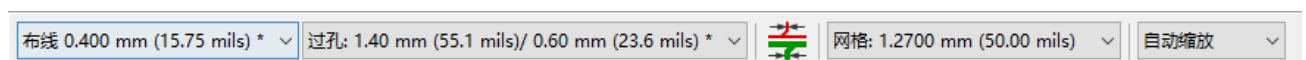
可从以下下拉菜单访问最重要的参数：



并在“设计规则”对话框中设置。

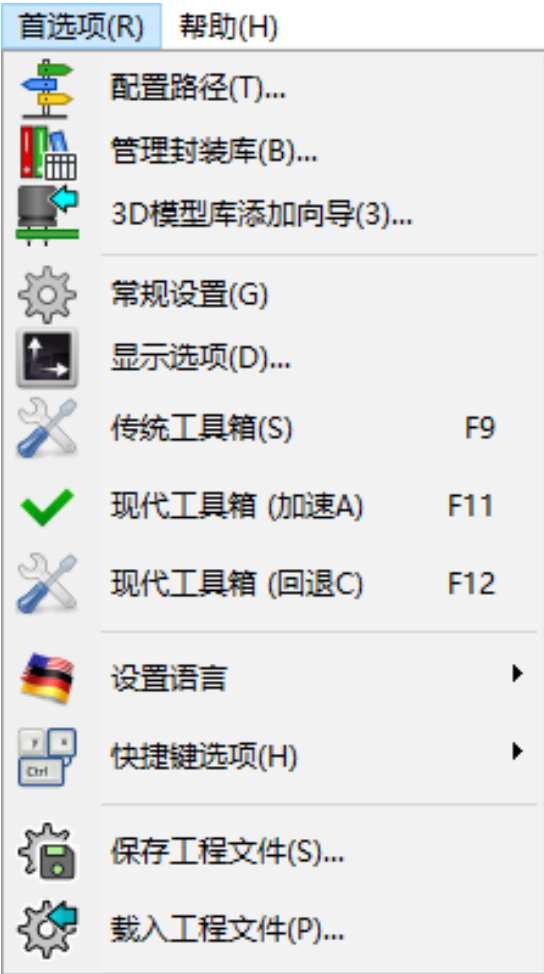
8.1.2 当前设置

当前设置显示在顶部工具栏中。

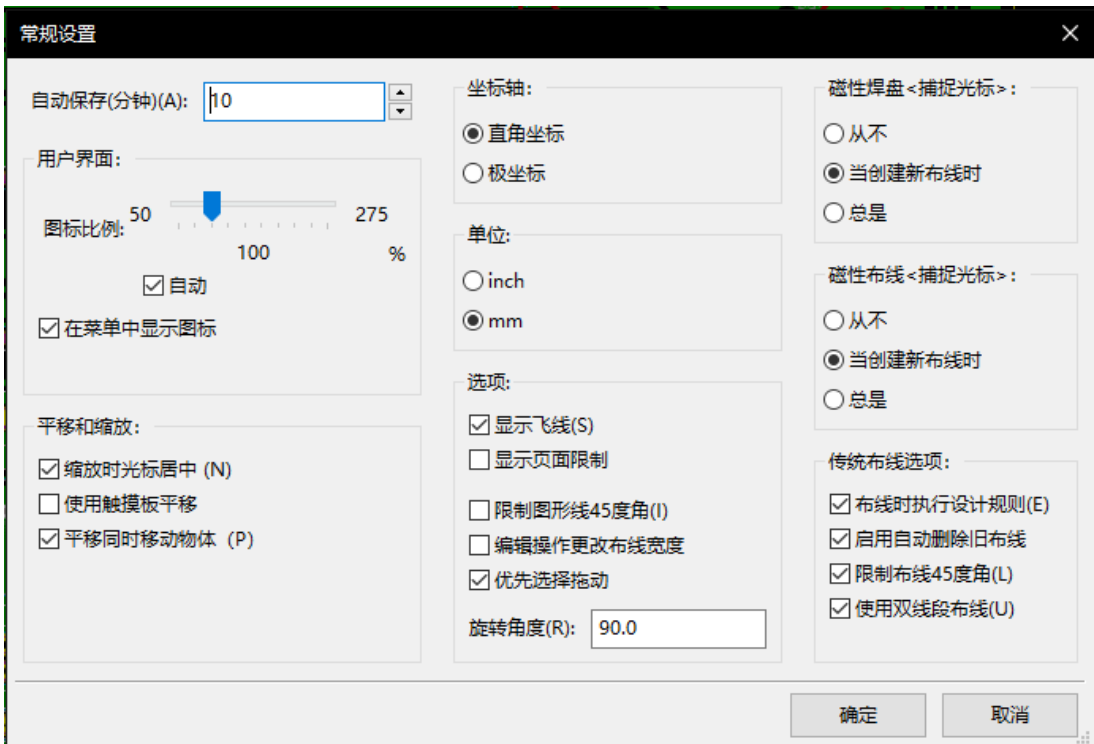


8.2 常规选项

可通过顶部工具栏链接首选项 -> 常规对话框获取常规选项菜单。



该对话框如下所示：



对于布线的创建，必要的参数是：

- **仅限布线 45**：布线段允许的方向为 0,45 或 90 度。
- **双段布线**：创建布线时，将有 2 段显示。
- **布线自动删除**：重新创建布线时，旧布线将是如果被认为是冗余的
- **磁焊盘**：图形光标变成一个焊盘，以中心为中心焊盘区。
- **磁布线**：图形光标成为布线轴。

8.3 网类

Pcbnew 允许您为每个网络定义不同的布线参数。参数由一组网络定义。

- 一组网称为网类。
- 总有一个名为“默认”的网类。
- 用户可以添加其他网类。

网类指定：

- 布线宽度，直径和钻孔。
- 焊盘和布线（或通孔）之间的间隙。
- 布线时，Pcbnew 会自动选择与要创建或编辑的布线网络对应的网络类，从而选择布线参数。

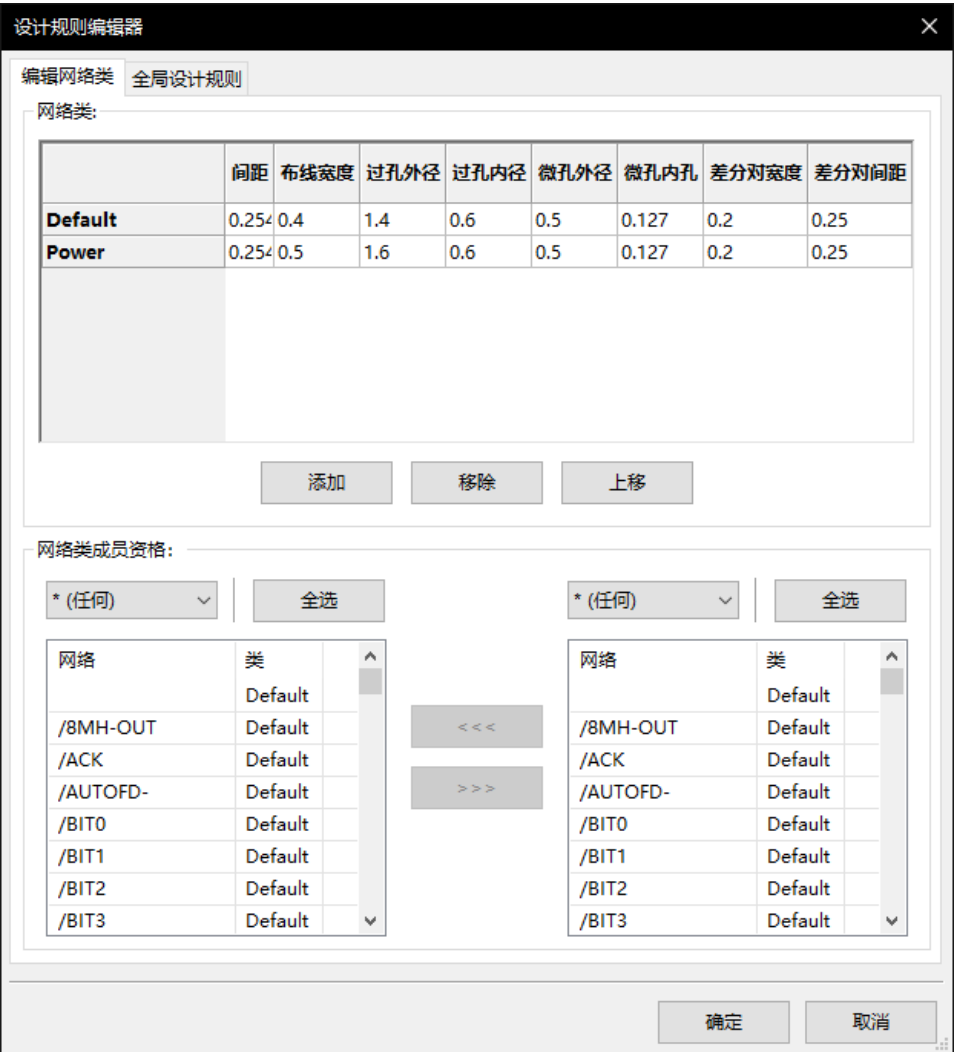
8.3.1 设置布线参数

可在菜单中进行选择：设计规则 → 设计规则。

8.3.2 网类编辑器

网类编辑器允许您：

- 添加或删除网类。
 - 设置布线参数值：间隙，布线宽度，通孔尺寸。
 - 在网中组网类。
-



8.3.3 全局设计规则

全局设计规则是：

- 启用/禁用盲孔/埋空的使用。
- 启用/禁用微过孔使用。
- 布线和过孔的最小允许值。

遇到小于指定最小值的值时，会引发 DRC 错误。第二个对话框面板是：

设计规则编辑器

编辑网络类

全局设计规则

布线选项:

最小布线宽度:0.2mm

最小过孔外径:0.9mm

最小过孔内径:0.5mm

☐ 允许盲孔或埋孔

☐ 允许微孔

最小微孔外径:0.5mm

最小微孔内径:0.127mm

具体过孔外径和布线宽度，可以按需要替换默认的网络类值给任何过孔或布线段。

自定义过孔尺寸:

钻孔值: 空或为 0 => 默认网络类值

	直径	钻孔
过孔 1	1.524	0.762
过孔 2		
过孔 3		
过孔 4		
过孔 5		
过孔 6		
过孔 7		
过孔 8		

自定义布线宽度:

	宽度
布线 1	0.381
布线 2	0.762
布线 3	
布线 4	
布线 5	
布线 6	
布线 7	
布线 8	

确定

取消

此对话框还允许输入布线和过孔尺寸的“库存”。

布线时，可以选择其中一个值来创建布线或过孔，而不是使用网类的默认值。

在小型布线段必须具有特定尺寸的关键情况下非常有用。

8.3.4 过孔参数

Pcbnew 处理 3 种类型的过孔：

- 通孔（通常的过孔）。
- 盲孔或埋孔。
- 微过孔，与埋入的通孔一样，但仅限于与其最近邻居的外部层。它们用于将 BGA 引脚连接到最近的内层。它们的直径通常非常小，并且通过激光钻孔。

默认情况下，所有过孔都具有相同的钻孔值。

此对话框指定过孔参数的最小可接受值。在电路板上，小于此处指定的通道会产生 DRC 错误。

8.3.5 布线参数

指定可接受的最小布线宽度。在电路板上，小于此处指定的磁道宽度会产生 DRC 错误。

8.3.6 具体尺寸

具体过孔外径和布线宽度，可以按需要替换默认的网络类值给任何过孔或布线段。

自定义过孔尺寸:

钻孔值: 空或为 0 => 默认网络类值

	直径	钻孔
过孔 1	1.524	0.762
过孔 2		
过孔 3		
过孔 4		
过孔 5		
过孔 6		
过孔 7		
过孔 8		

自定义布线宽度:

	宽度
布线 1	0.381
布线 2	0.762
布线 3	
布线 4	
布线 5	
布线 6	
布线 7	
布线 8	

可以输入一组额外的布线和/或过孔尺寸。在布线时，可以按需使用这些值，而不是当前网络类值的值。

8.4 示例和典型尺寸

8.4.1 布线宽度

使用尽可能大的值并符合此处给出的最小尺寸。

单位	CLASS 1	CLASS 2	CLASS 3	CLASS 4	CLASS 5
mm	0.8	0.5	0.4	0.25	0.15
mils	31	20	16	10	6

8.4.2 绝缘（间隙）

单位	CLASS 1	CLASS 2	CLASS 3	CLASS 4	CLASS 5
mm	0.7	0.5	0.35	0.23	0.15
mils	27	20	14	9	6

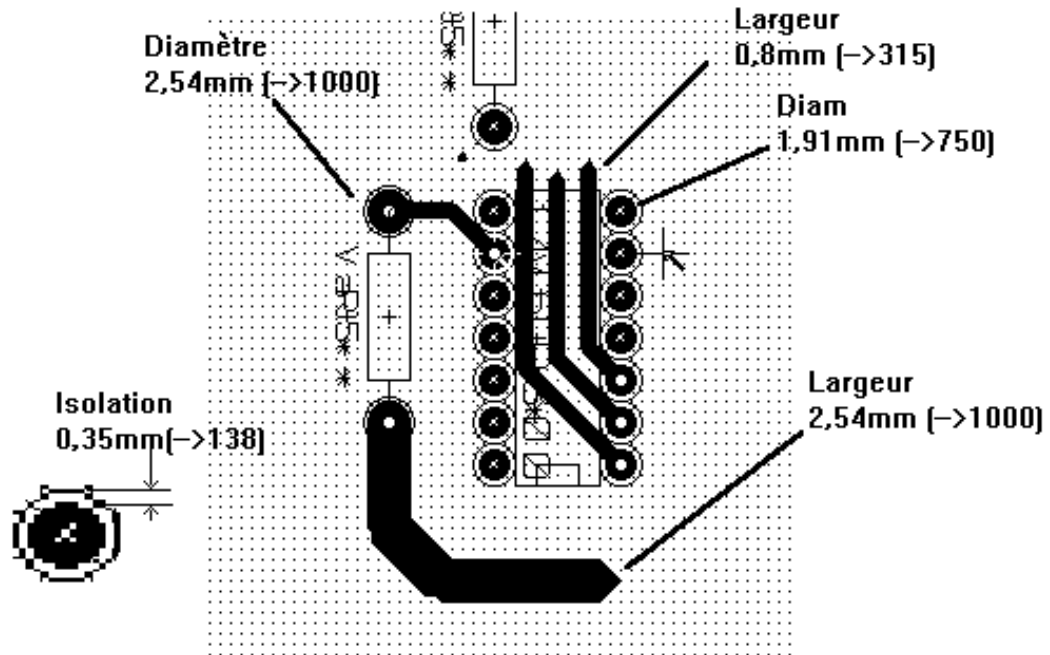
通常，最小间隙与最小布线宽度非常相似。

8.5 例子

8.5.1 粗制

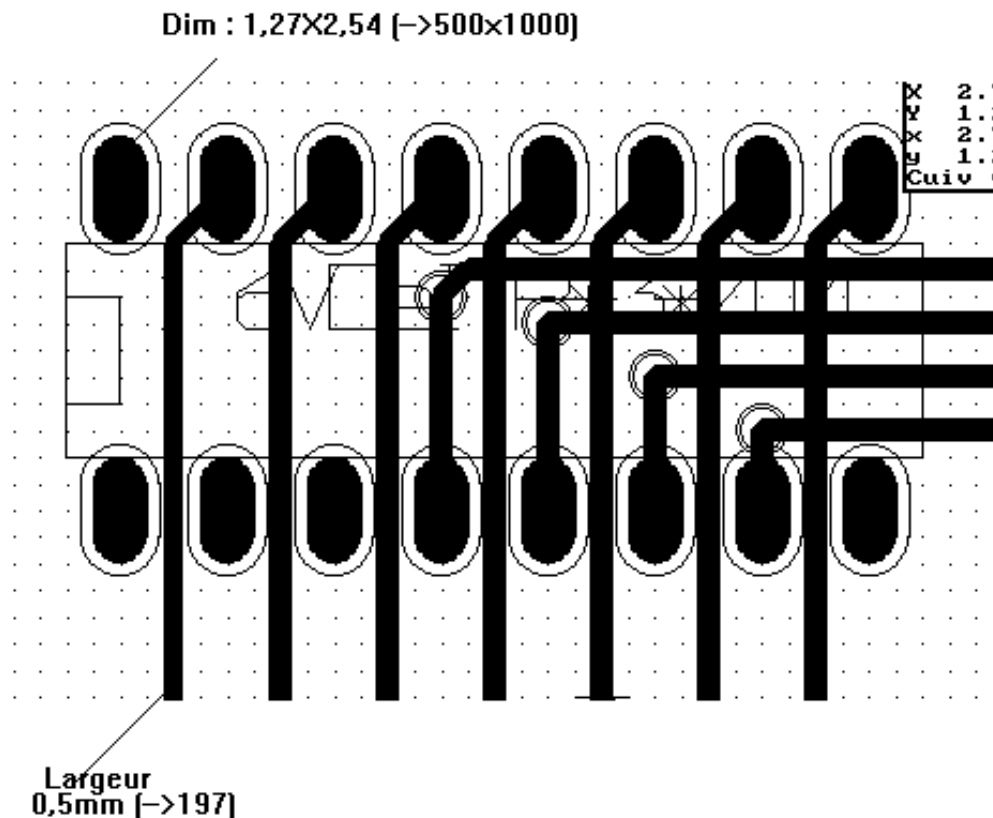
- 间隙：0.35 mm (0.0138 inches)。

- 布线宽度: 0.8 mm (0.0315 inches)。
- IC 和过孔的焊盘直径: 1.91 mm (0.0750 inches)。
- 直插元件的焊盘直径: 2.54 mm (0.1 inches)。
- 地 (GND) 布线宽度: 2.54 mm (0.1 inches)。



8.5.2 标准


- 间隙: 0.35 mm (0.0138 inches)。
- 布线宽度: 0.5 mm (0.0127 inches)。
- IC 的焊盘直径: 使它们伸长, 以便允许布线在 IC 焊盘之间通过, 并使焊盘提供足够的粘合表面 (1.27 x 2.54 mm - > 0.05 x 0.1 inches)。
- 过孔: 1.27 mm (0.0500 inches)。




8.6 手动布线

通常建议使用手动布线，因为它是提供对布线优先级控制的唯一方法。例如，最好首先布线电源，使其宽而短，并使模拟和数字电源保持良好分离。之后，应该布线敏感信号轨道。在其他问题中，自动布线通常需要许多过孔。但是，自动布线可以提供有关封装定位的有用信息。根据经验，您可能会发现自动布线对于快速“明显”的非常有用，但其余的最好还是手动布线。


8.7 创建布线时的帮助

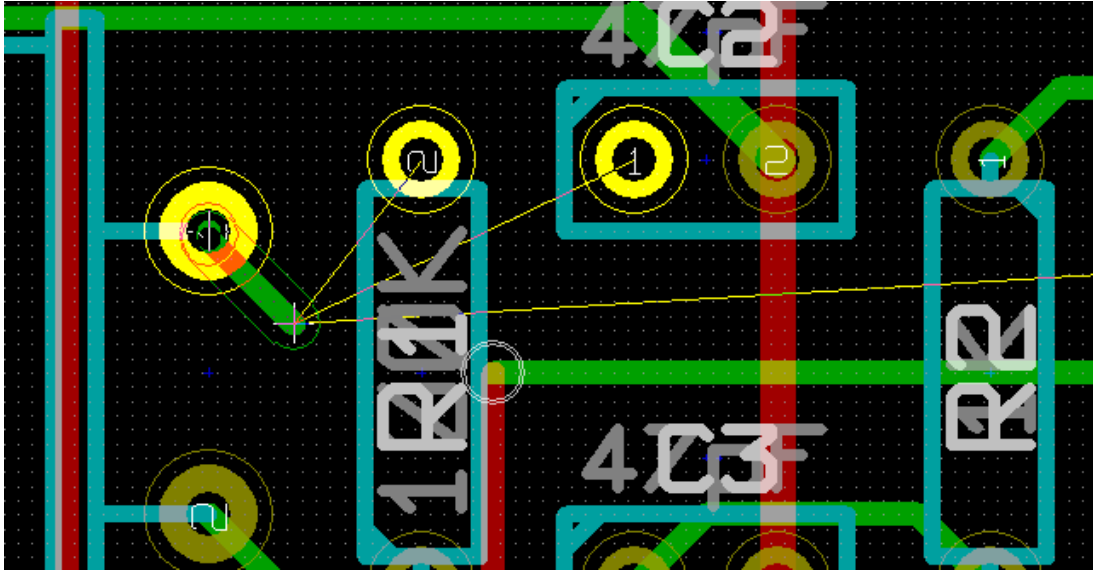
如果按钮图像： 被激活，Pcbnew 可以显示完整的飞线。

按钮图像： 允许人们突出显示网络（单击一个焊盘或现有布线以高亮显示相应的网络）。


DRC 在创建布线时实时检查布线。无法创建与 DRC 规则不匹配的布线。可以通过单击按钮禁用 DRC。但是，不建议这样做，仅在特定情况下使用它。

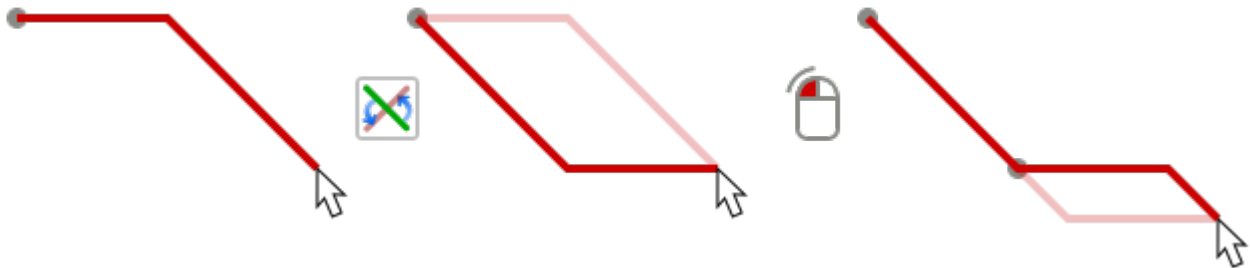
8.7.1 创建布线

单击按钮图像可以创建一个布线：。新的布线必须在焊盘或另一个布线上开始，因为 Pcbnew 必须知道用于新布线的网络（为了匹配 DRC 规则）。



移动鼠标时，会绘制一条布线，将布线原点与当前鼠标位置连接起来。将使用最多两个段（例如，向右，然后切换到对角线）绘制布线。布线锁定在角节点中时单击。

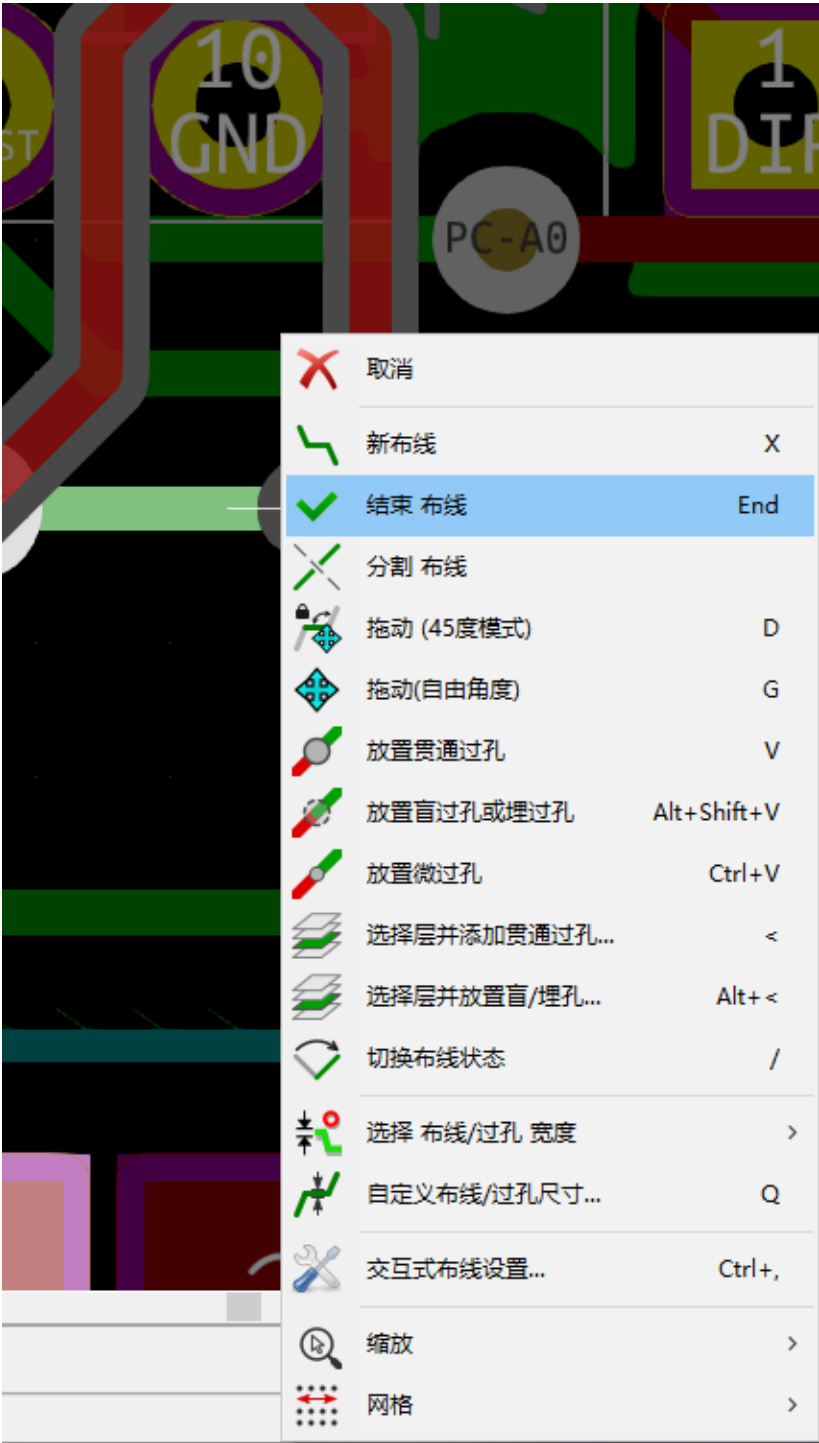
首先绘制布线的方向（例如，首先，然后是对角线，或者首先是对角线，然后是右侧）称为“布线样式”，可以使用热键“/”或按钮图像进行切换：。




在非传统画布中布线时按住“Ctrl”会将布线限制为单个水平或垂直段。切换样式变为单个差分对线段。在路由时按住“Shift”可以移除“捕捉到对象”的重力。

创建新焊盘时，Pcbnew 会显示指向最近的未连接焊盘的链接。

通过双击，弹出菜单或热键“结束”结束布线。



8.7.2 移动和拖动布线

当按钮图像： 处于活动状态时，可以使用热键“M”移动光标所在的布线。如果要拖动布线，可以使用热键“G”。

8.7.3 插入过孔


只有在布线正在进行时才能插入过孔：

- 通过弹出菜单。
- 通过热键 “V”。
- 通过使用适当的热键切换到新的铜层。

在放置过孔时，在添加过孔时按住 “Shift” 结束布线。这在向平面添加连接时非常有用，因此活动层不会更改，也不需要按下额外的键来退出布线。

8.8 选择/编辑布线宽度和过孔尺寸

单击布线或焊盘时，Pcbnew 会自动选择相应的网类，并且布线大小和过孔尺寸将从此网类导出。
如前所述，全局设计规则编辑器具有插入额外布线和过孔尺寸的工具。

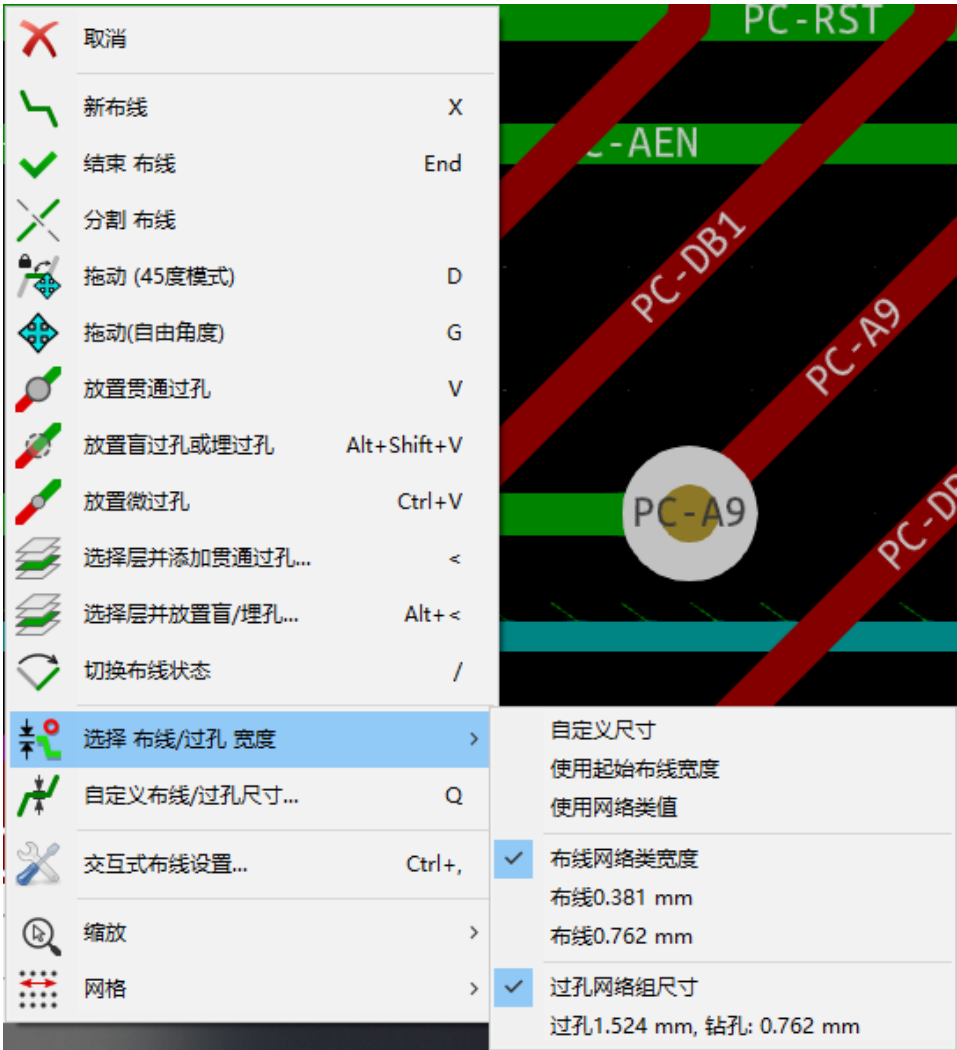
- 水平工具栏可用于选择大小。
- 当按钮图像： 处于活动状态时，可以从弹出菜单中选择当前布线宽度（创建布线时也可以访问）。
- 用户可以使用默认的网类值或指定的值。

8.8.1 使用水平工具栏

	
	布线宽度选择。符号 * 是默认网类的标记值选择。
	过孔尺寸选择。符号 * 是默认网类值选择的标记。
	启用时：自动选择布线宽度。在现有布线上启动布线时，新布线的宽度与现有布线的宽度相同。
	网格大小选择。
	缩放选择。

8.8.2 使用弹出菜单

可以选择新的尺寸进行布线，或者更改为先前创建的过孔或布线对：



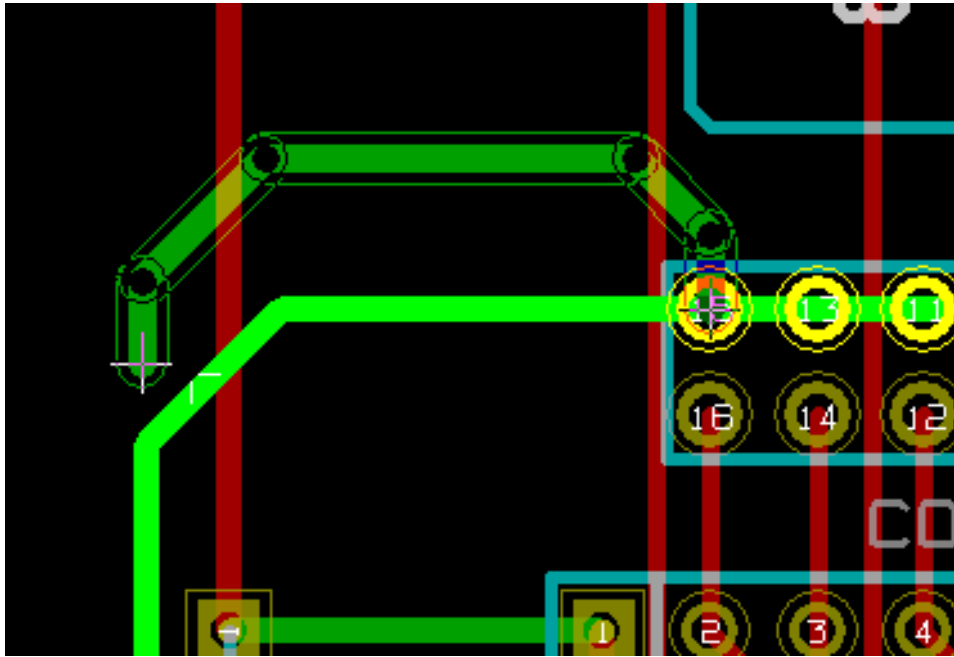
如果要更改许多过孔（或布线）大小，最好的方法是必须编辑的网络使用特定的网类（请参阅全局更改）。

8.9 编辑和更改布线

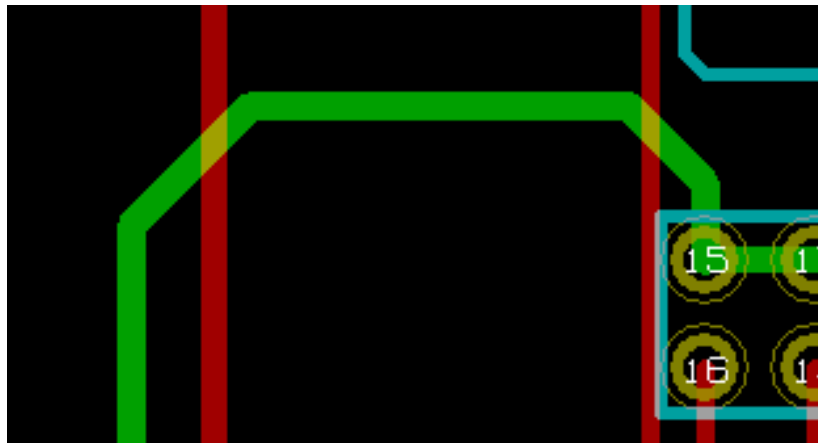
8.9.1 更改布线

在许多情况下，需要重新绘制布线。

新布线（正在进行中）：



等完成了：



如果冗余，Pcbnew 将自动删除旧布线。

8.9.2 全局更改

通过右键单击布线，可以通过弹出窗口访问全局布线和过孔尺寸对话框编辑器：

编辑(E) 视图(V) 设置(S) 放置(P) 布线		
	撤消(U)	Ctrl+Z
	重做(R)	Ctrl+Y
		
	剪切(C)	Ctrl+X
	复制(C)	Ctrl+C
	粘贴(P)	Ctrl+V
	删除(D)	
		
	查找(F)...	Ctrl+F
		
	编辑所有布线和过孔...	
	设置封装字段尺寸(R)...	
	修改封装...	
	移动和交换层(M)...	
		
	全局删除(G)...	
	清除布线和过孔(C)...	

对话框编辑器允许全局更改布线和/或过孔：

- 目前的网。
- 整个板。

布线

和过孔全局选项

✕

当前设置:

当前网络:

网络 000 <no net>

▼

当前网络类:

Default

	布线尺寸	过孔外径	过孔内径	微孔外径	微孔内孔
网络类值	0.4 mm	1.4 mm	0.6 mm	0.5 mm	0.127 mm
当前值	默认	默认	0.6 mm	默认	默认

编辑全局选项:

☐ 将当前网络的布线和过孔设置为当前选定的用户值

☒ 设置当前网络的布线和过孔的网络类值

☐ 网络类值设置所有布线和过孔

☐ 网络类值设置所有过孔(不包含布线)

☐ 网络类值设置所有布线(不包含过孔)

确定

取消

Chapter 9

交互式布线

通过交互式布线，您可以通过推移或绕走 PCB 上与您当前绘制的布线相撞的项目来快速有效地布线 PCB。

支持以下模式：

- **高亮显示碰撞**，高亮显示所有违规对象漂亮，闪亮的绿色和显示违反间隙区域。
- **推**，试图推动和推动所有碰撞的物品目前布线。
- **绕走**，试图通过拥抱/绕走来避开障碍物他们。

9.1 配置

在使用交互式布线之前，请设置以下两项内容：

- **间隙设置**要设置间隙，请打开 设计规则对话框并确保至少看到默认的间隙值明智的。

交互布线设置

×

模式:

☐ 高亮碰撞

☐ 推挤

☒ 绕走

选项:

鼠标拖动行为: 移动项

☐ 自由角度模式 (不推挤/绕走)

☐ 绕过障碍物

☒ 移除多余的布线

☒ 优化焊盘连接

☒ 平滑拖动线段

☐ 允许违反DRC规则

优化工作:

低

高


确定

取消

- 启用 OpenGL 模式通过选择 视图 → 切换画布到 OpenGL 菜单选项或按 F11 。



9.2 布线

要激活布线工具，请按 交互式布线按钮图片： 或 X 键。光标将变为十字形，工具名称将显示在状态栏中。

要启动布线，请单击任何项目（焊盘，布线或过孔）或再次按 X 键将鼠标悬停在该项目上。新曲目将使用起始项的网络。在空 PCB 空间上单击或按 X 可启动未分配网络的布线。

移动鼠标以定义布线的形状。布线将尝试跟踪鼠标轨迹，拥抱不可移动的障碍物（例如焊盘）和推迟碰撞轨迹/过孔，具体取决于模式。撤回鼠标光标将使推送的项目回弹到以前的位置。

单击同一网络中的焊盘/布线/通过完成布线。单击空白区域可修复到目前为止布线的段，并继续布线跟踪。

要停止布线并撤消所有更改（推送项目等），只需按 **Esc** 即可。

按下 **V** 或从上下文菜单中选择 放置通孔，同时布线会在路径的末尾附加一个过孔。按 **V** 再次禁用通过放置。单击任何位置可建立通道并继续布线（除非保持 “Shift”）。

按 **/** 或从上下文菜单中选择 切换布线样式可切换直线或对角线之间的初始布线线段的方向。

Note

默认情况下，布线器捕捉到项目的中心/轴。在布线或选择项目时按住 **Shift** 可以禁用捕捉。

9.3 设置布线宽度和通孔尺寸

有多种方法可以预先选择布线宽度/通孔尺寸或在布线期间更改它：

- 使用标准的 KiCad 快捷方式。
- 按 **W** 或从上下文菜单中选择 自定义布线宽度以键入自定义布线宽度/通孔大小。
- 从上下文菜单的 选择布线宽度子菜单中选择预定义的宽度。
- 使用初始布线宽度菜单中的 选择初始布线宽度来从起始项目（或已连接到它的布线）中选择宽度。

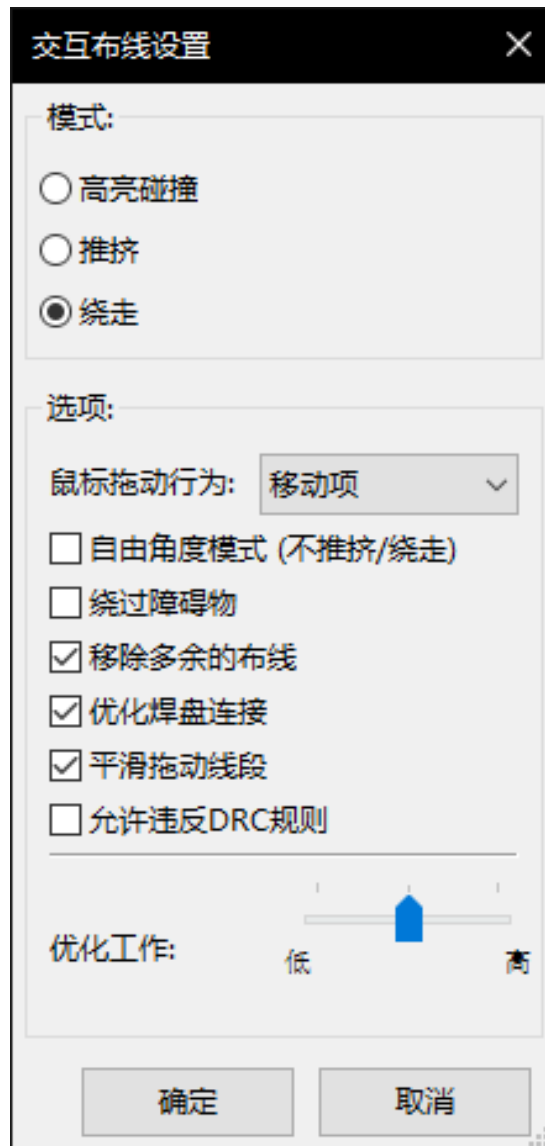
9.4 拖动

路由器可以拖动布线段，角落和过孔。要拖动项目，请按住 **Ctrl** 键单击它，将鼠标悬停在鼠标上并按 **G** 或从上下文菜单中选择 拖动布线/过孔。通过再次单击完成拖动或按 **Esc** 中止。

9.5 选项

在布线模式下，可以通过按 **E** 或从上下文菜单中选择 布线选项来配置布线行为。它会打开一个如下所示的窗口：

选项是：



- **模式** - 选择布线如何处理 DRC 违规（推, 走拥抱等）
- **推孔过孔** - 禁用时，过孔被视为不可移动的物体并且拥抱而不是推。
- **跳过障碍物** - 启用后，路由器会尝试移动在固体障碍物（例如垫）后面碰撞痕迹而不是“反映”了碰撞
- **删除冗余布线** - 在布线时删除循环（例如，如果新布线确保与现有布线相同的连接性布线被删除）。循环删除在本地工作（仅在开始之间并结束当前布线的跟踪）。
- **自动缩减** - 启用后，布线会尝试穿过焊盘/过孔以干净的方式，避免锐角和锯齿状穿过痕迹。
- **平滑拖动的段** - 启用后，路由器会尝试合并几个锯齿状的段成一个直的（拖动模式）。
- **允许 DRC 违规**（仅限 高亮碰撞模式）- 允许即使违反 DRC 规则也要建立一条布线。
- **优化工作量** - 定义布线应花费多少时间优化布线/推送布线。更多的努力意味着清洁布线（但速度较慢），较少的工作意味着更快的布线，但有点锯齿状痕迹。

Chapter 10

创建铜区

铜区域由边框（闭合多边形）定义，并且可以包括孔（边框内的闭合多边形）。可以在铜层上或在技术层上绘制区域。

10.1 在铜层上创建区域

DRC 引擎检查填充铜区域的焊盘（和布线）连接。必须填充（不仅仅是创建）区域以连接焊盘。Pcbnew 目前使用布线段或多边形来填充铜区域。

每个选项都有其优点和缺点，主要缺点是在较慢的机器上增加了屏幕重绘时间。然而，最终结果是一样的。

出于计算时间原因，每次更改后都不会重新创建区域填充，但仅限于：

- 如果执行填充区命令。
- 执行 DRC 测试时。

在更换布线或焊盘后，必须填充或重新填充铜区域。铜区（通常是地面和电力层）通常连接到网上。

要创建铜区，您应该：

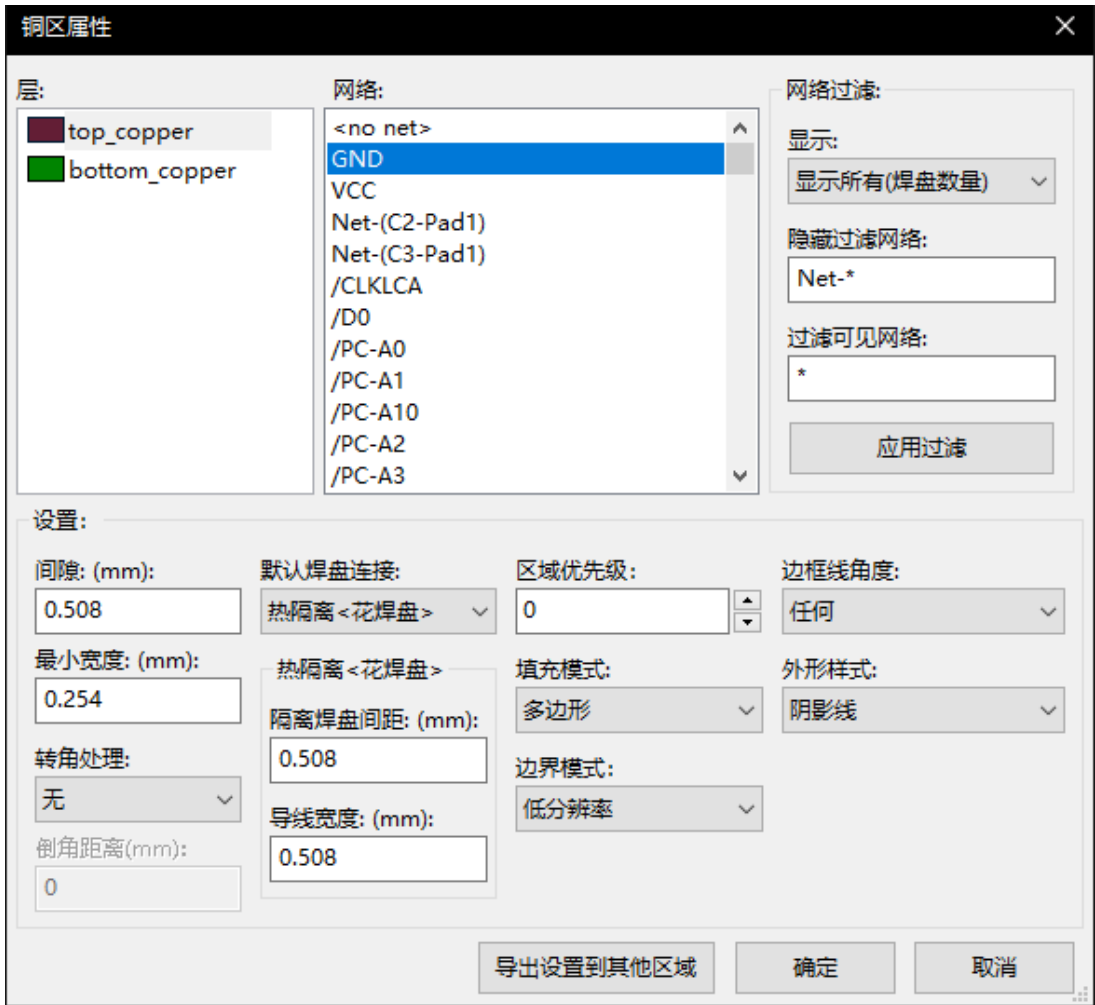
- 选择参数（网名，图层……）。打开图层并突出显示此网不是强制性的，但这是一种很好的做法。
- 创建区域限制（如果不是，则将填充整个板）。
- 填充区域。

Pcbnew 尝试将所有区域填充为一块，通常，没有未连接的铜块。可能会发生一些地区仍未填补的情况。没有网的区域没有清洁，可以有隔热区域。

10.2 创建区域

10.2.1 创建区域的限制

使用工具。有源层必须是铜层。单击以启动区域边框时，将打开以下对话框。



您可以指定此区域的所有参数：

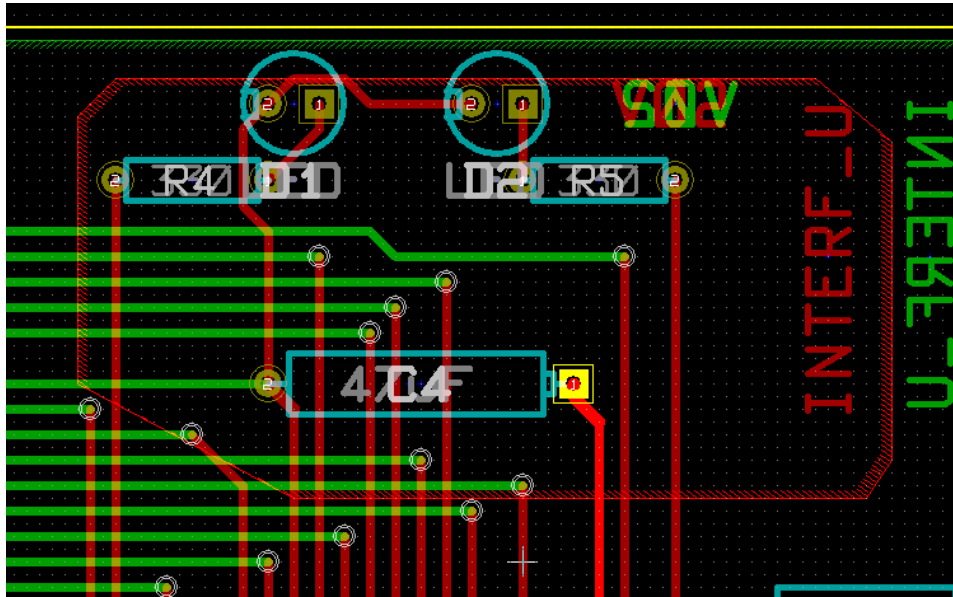
- 网
- 层
- 填充选项
- 焊盘选项
- 优先级

在此图层上绘制区域限制。此区域限制是一个多边形，通过在每个角上单击鼠标左键创建。双击将结束并关闭多边形。如果起点和终点不在同一坐标，Pcbnew 将添加从终点到起点的段。

Note

- 创建区域边框时，DRC 处于活动状态。
- Pcbnew 将不接受产生 DRC 错误的角落。

在下图中，您可以看到区域限制的示例（细阴影线中的多边形）：

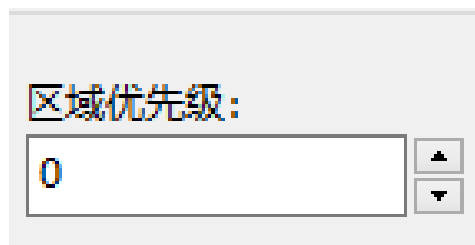


10.2.2 优先级

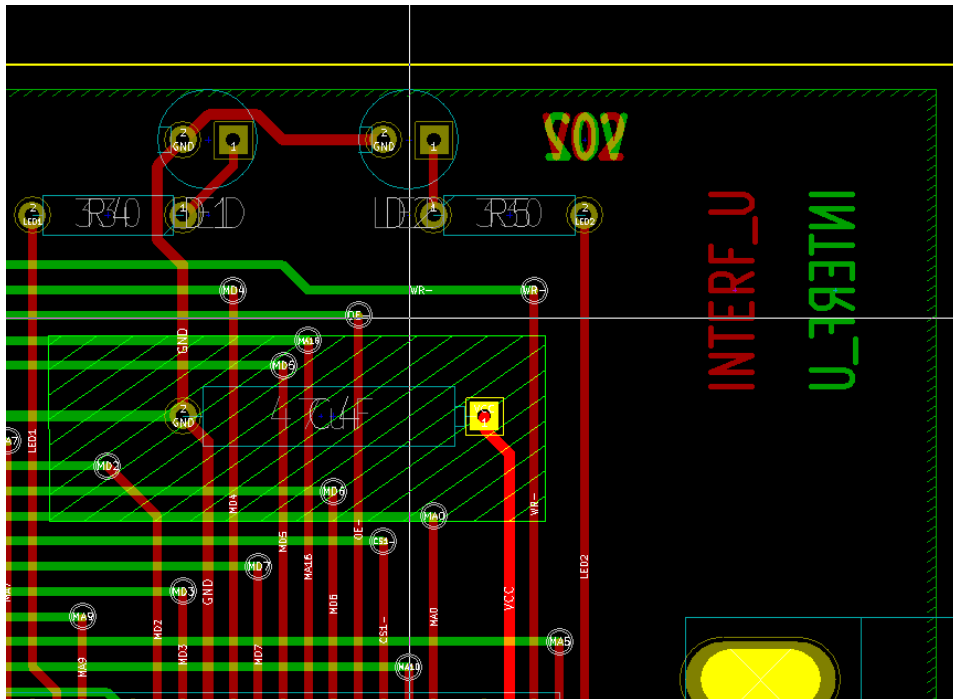
有时必须在大区域内创建一个小区域。

如果小区的优先级高于大区域，则可以执行此操作。

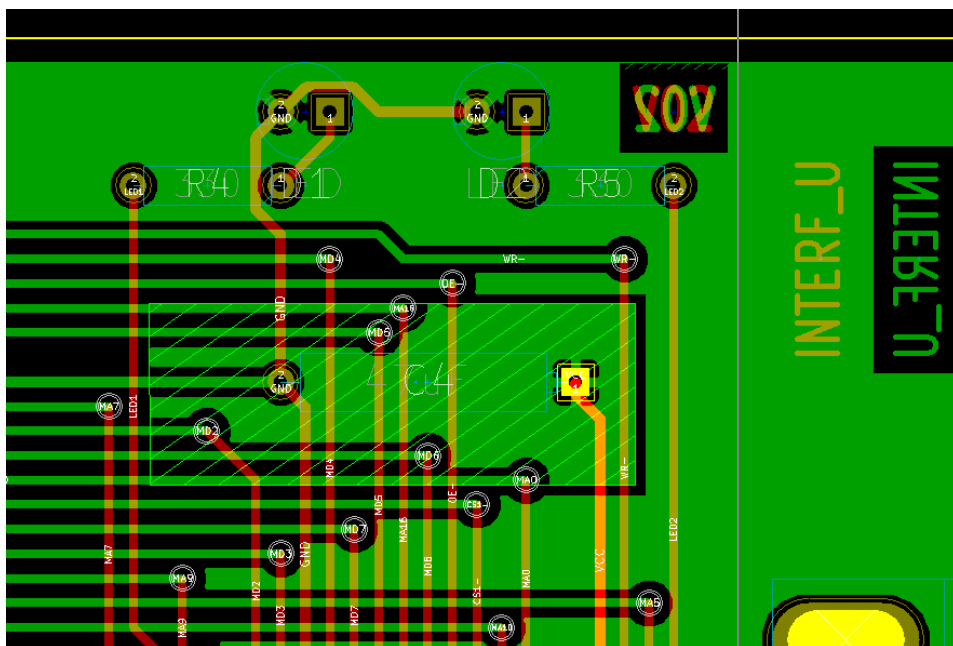
等级设置：



这是一个例子：

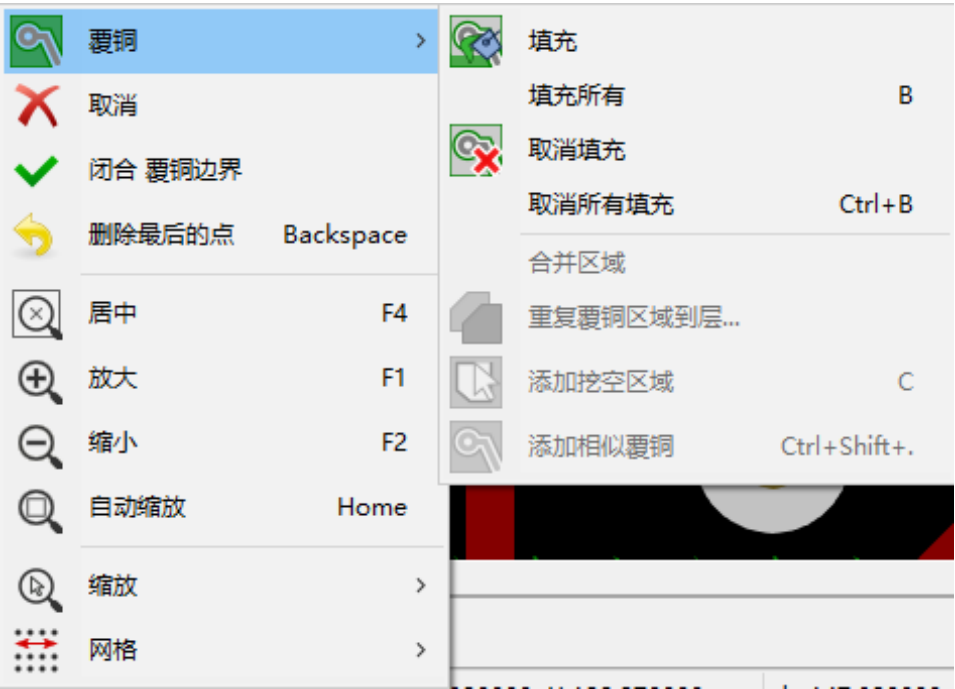


填充后:

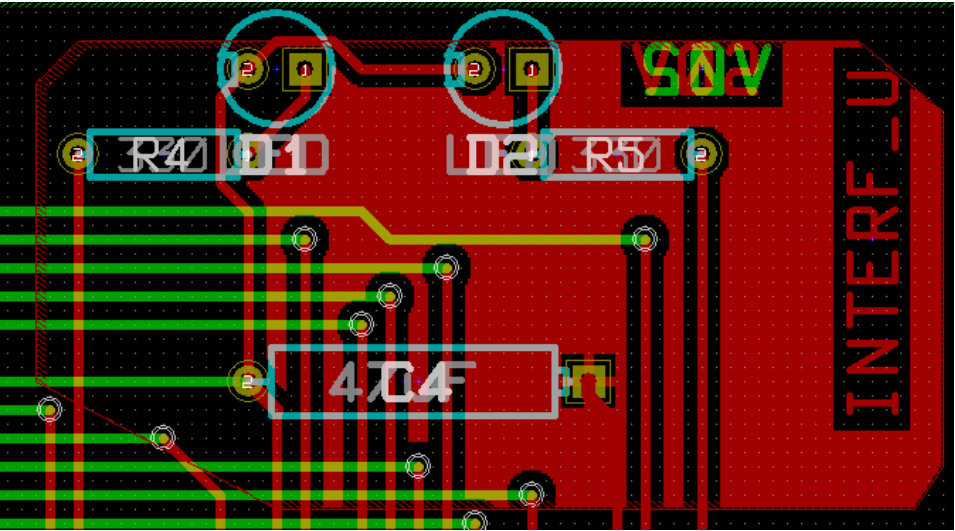


10.2.3 填充区域

填充区域时，Pcbnew 将删除所有未连接的孤铜。要访问区域填充命令，请右键单击边缘区域。



激活“填充区域”命令。下面是多边形内起点的填充结果：

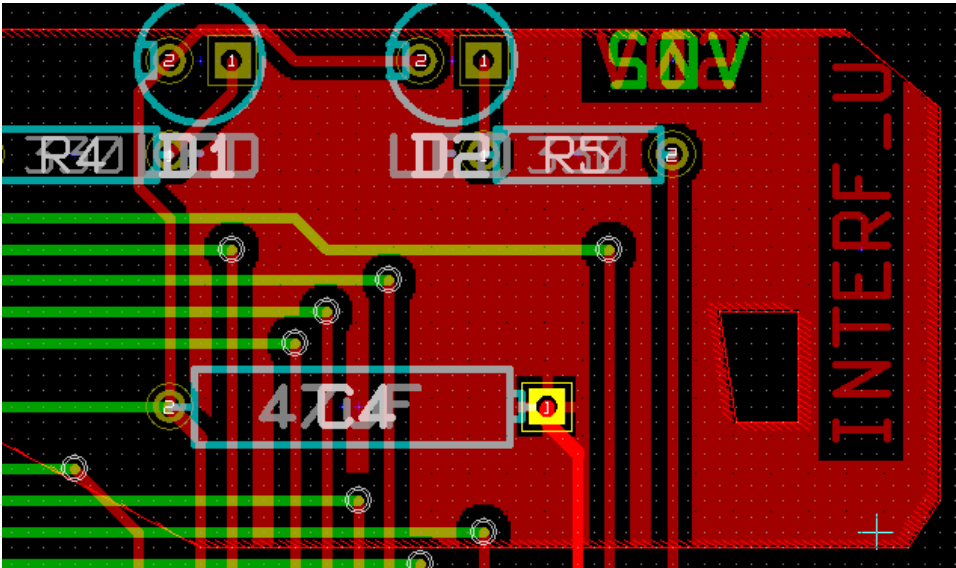


多边形是填充区域的边界。您可以在区域内看到未填充区域，因为此区域无法访问：

- 布线创建边框，和
- 填充这个区域没有起点。

Note

您可以使用多个多边形来创建剪切区域。在这里您可以看到一个例子：



10.3 填充选项

设置:

间隙: (mm): <input type="text" value="0.508"/>	默认焊盘连接: 热隔离 <花焊盘> ▼	区域优先级: <input type="text" value="0"/>	边框线角度: 任何 ▼
最小宽度: (mm): <input type="text" value="0.254"/>	热隔离 <花焊盘> 隔离焊盘间距: (mm): <input type="text" value="0.508"/>	填充模式: 多边形 ▼	外形样式: 阴影线 ▼
转角处理: 无 ▼	导线宽度: (mm): <input type="text" value="0.508"/>	边界模式: 低分辨率 ▼	
倒角距离(mm): <input type="text" value="0"/>			

填充区域时，您必须选择：

- 填充模式。
- 间隙和最小铜厚度。
- 如何在区域内（或连接到该区域）绘制焊盘。
- 热释放参数。

10.3.1 填充模式

可以使用多边形或线段填充区域。结果是一样的。如果您遇到多边形模式（慢速刷新屏幕）问题，则应使用分段。

10.3.2 间隙和最小铜厚度

清除的一个很好的选择是网格比路由网格稍大。最小铜厚度值确保铜区域不会太小。



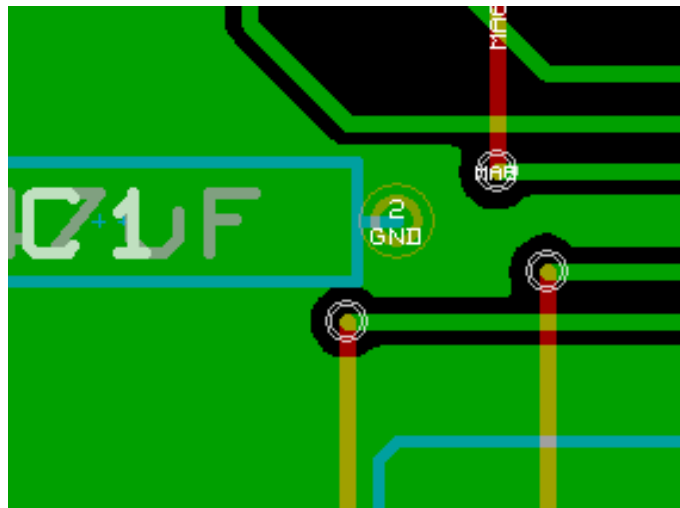
Warning

如果该值太大，则无法绘制出类似热浮雕中的热形状的小形状。

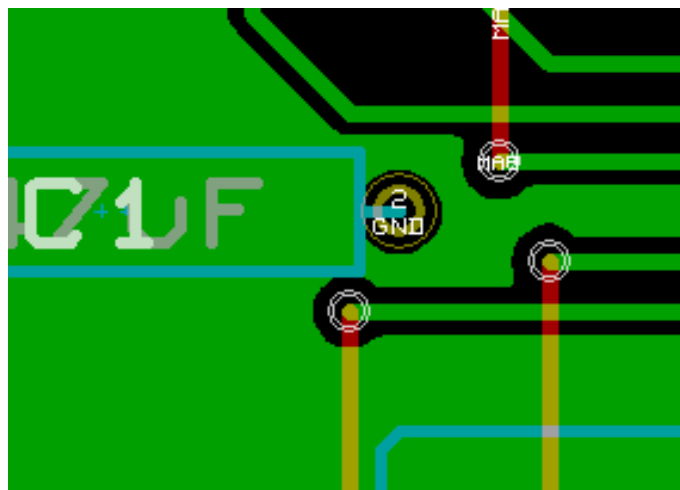
10.3.3 焊盘选项

网的焊盘可以包括在区域中或从区域中排除，或者通过热释放连接。

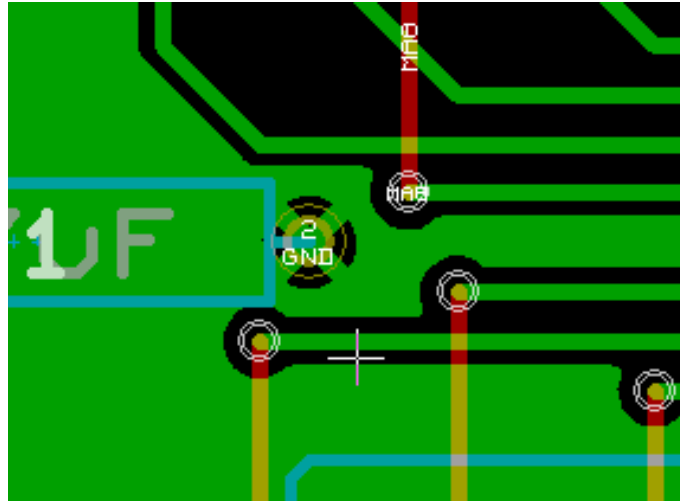
- 如果包括在内，由于大铜区域的高热质量，焊接和非焊接可能非常困难。



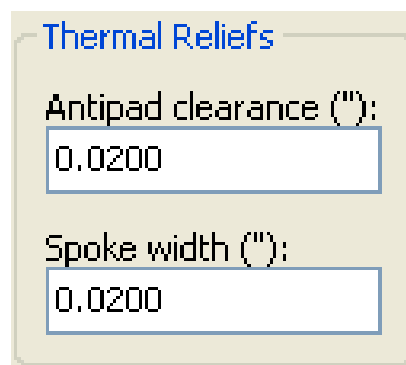
- 如果排除，则与区域的连接不会很好。
 - 仅当存在连接区域区域的布线时，才能填充区域。
 - 焊盘必须通过布线连接。



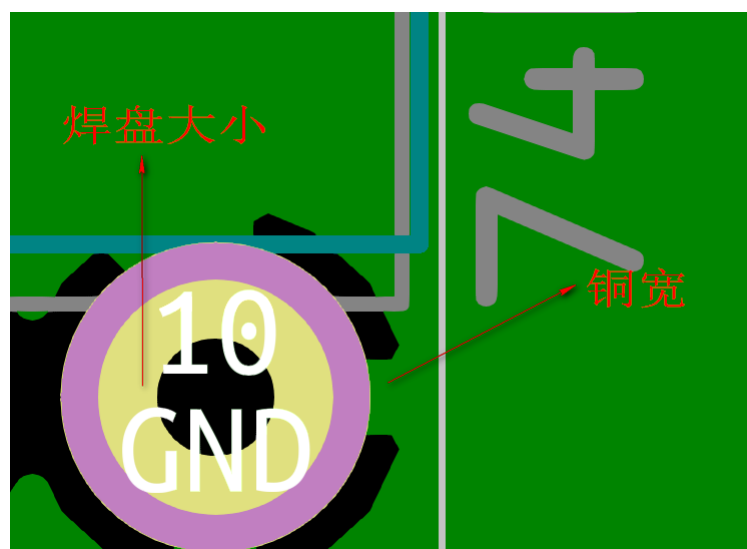
- 热释放是一个很好的妥协。
 - 焊盘由 4 个布线段连接。
 - 一段宽度是用于布线宽度的当前值。



10.3.4 热释放参数



您可以为热释放设置两个参数：



10.3.5 选择参数

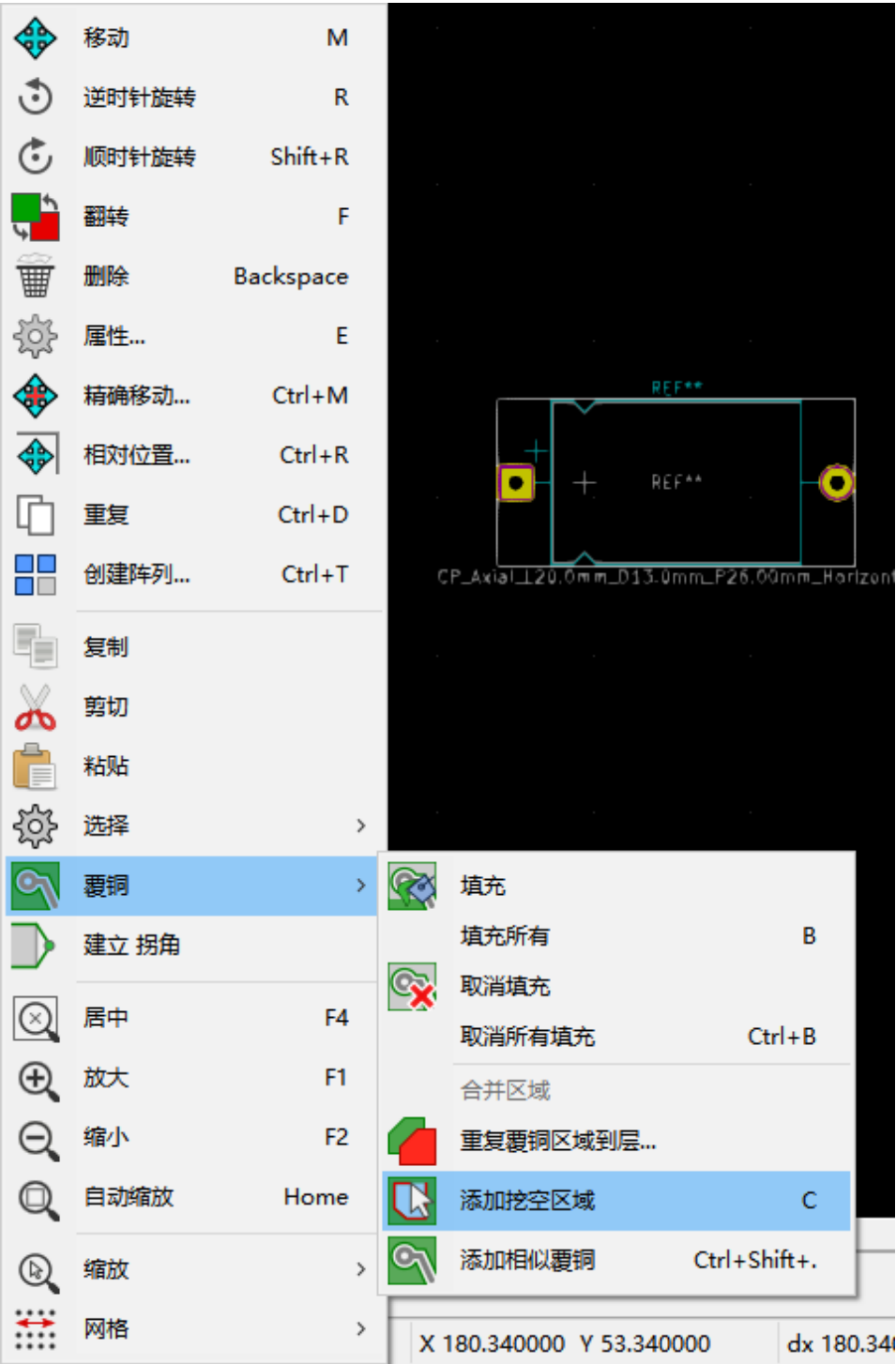
热释放的铜宽度值必须大于铜区的最小厚度值。如果没有，他们就无法画出来。

此外，此参数或反触摸尺寸的值太大，不允许为小焊盘创建热释放（例如用于 SMD 元件的焊盘尺寸）。

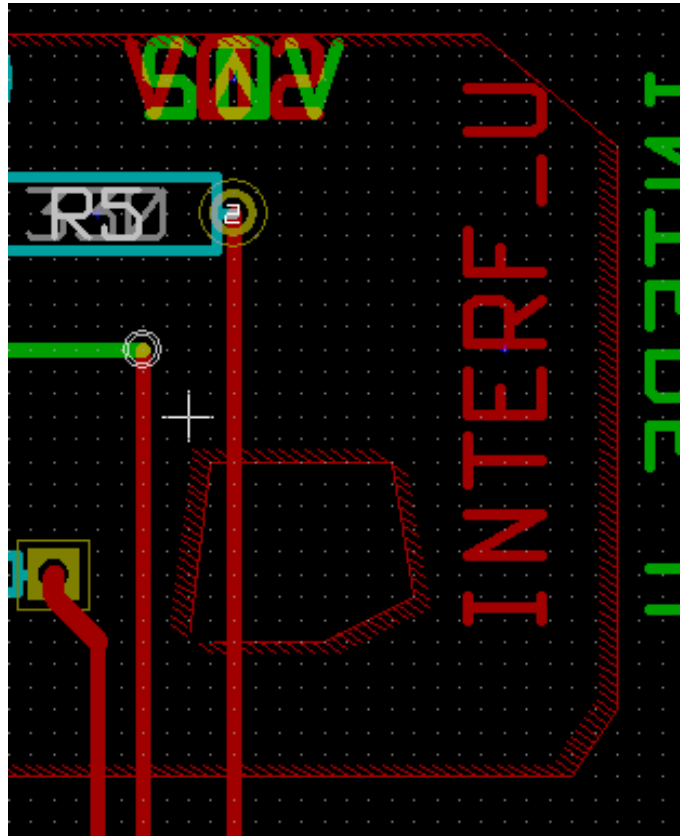
10.4 在区域内添加剪切区域

区域必须已存在。要添加剪切区域（区域内的非填充区域）：

- 右键单击现有边缘边框。
- 选择添加剪切区域。



- 创建新边框。

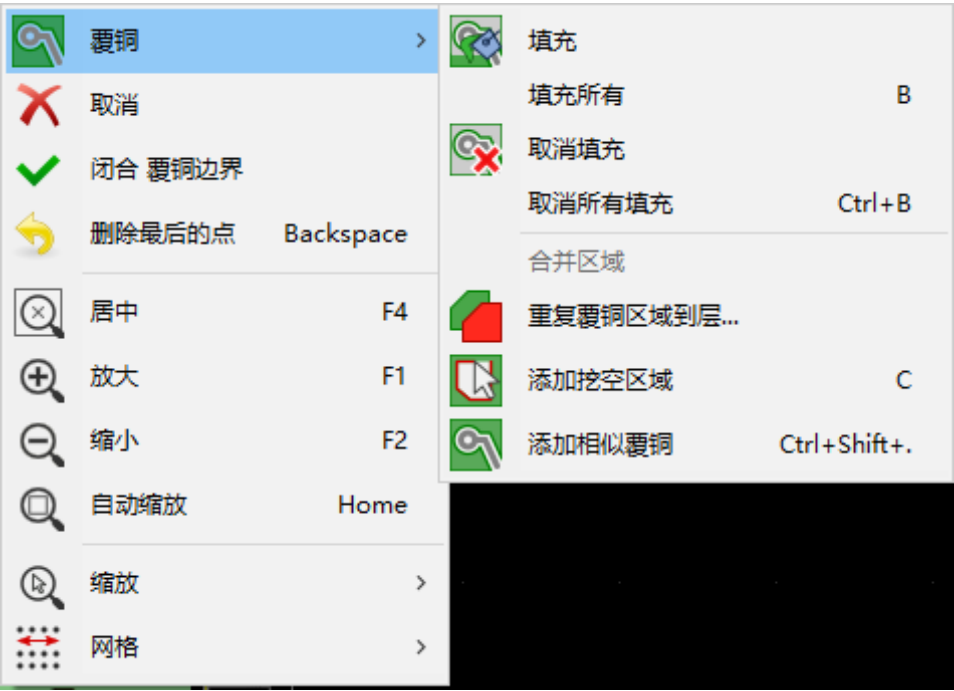


10.5 边框编辑

边框可以通过以下方式修改：

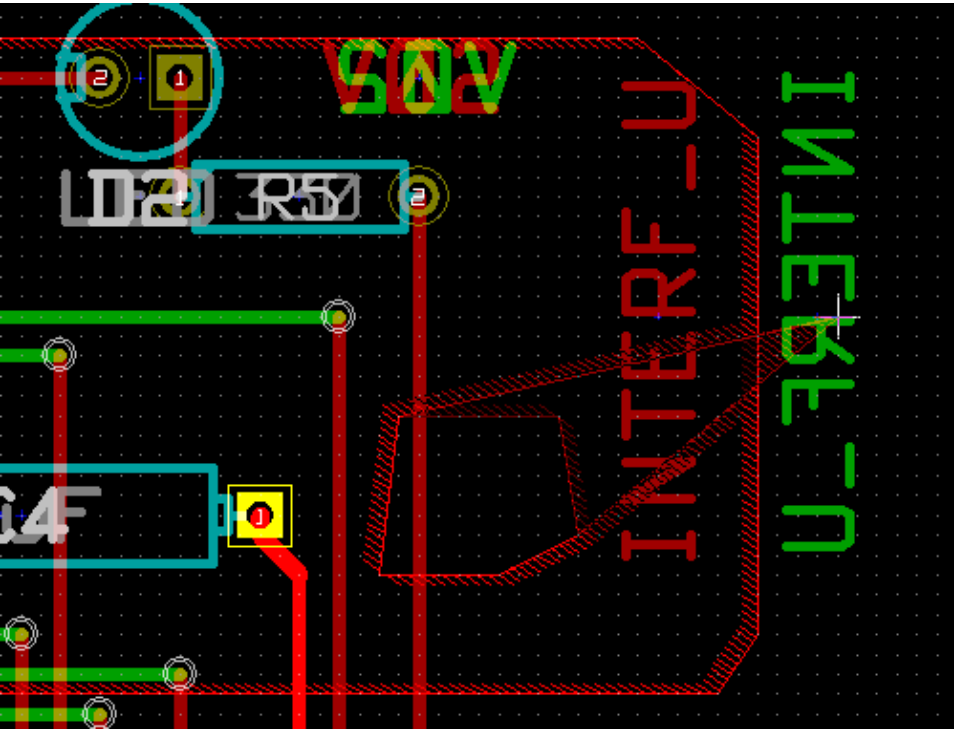
- 移动角落或边缘。
- 删除或添加角落。
- 添加类似区域或剪切区域。

如果多边形重叠，它们将被组合。

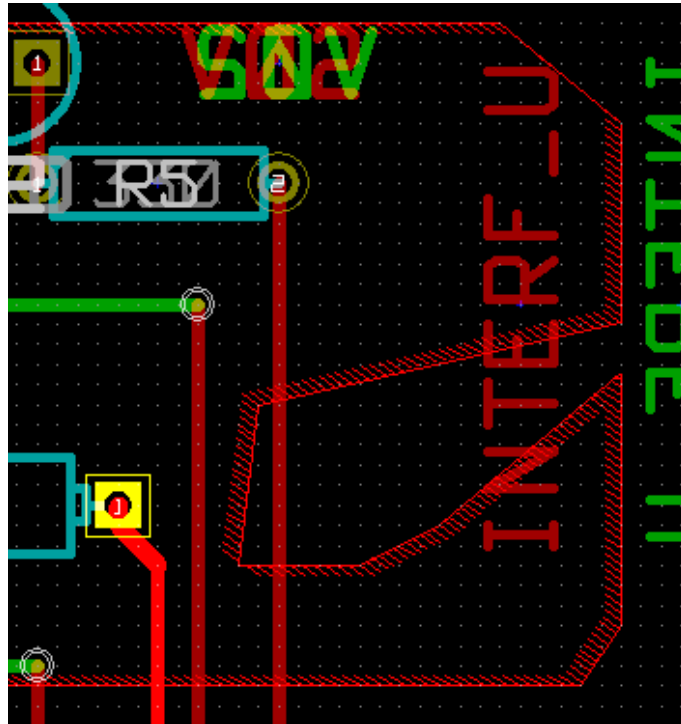


为此，右键单击一个角或边缘，然后选择正确的命令。

这是一个已被移动的角落（来自切口）：



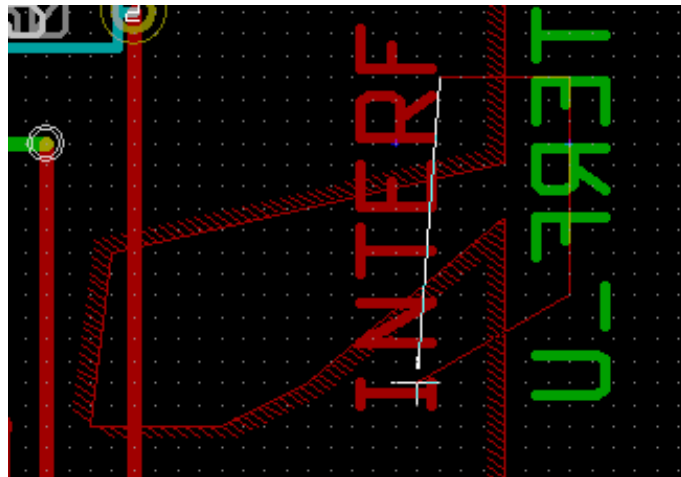
这是最终结果：



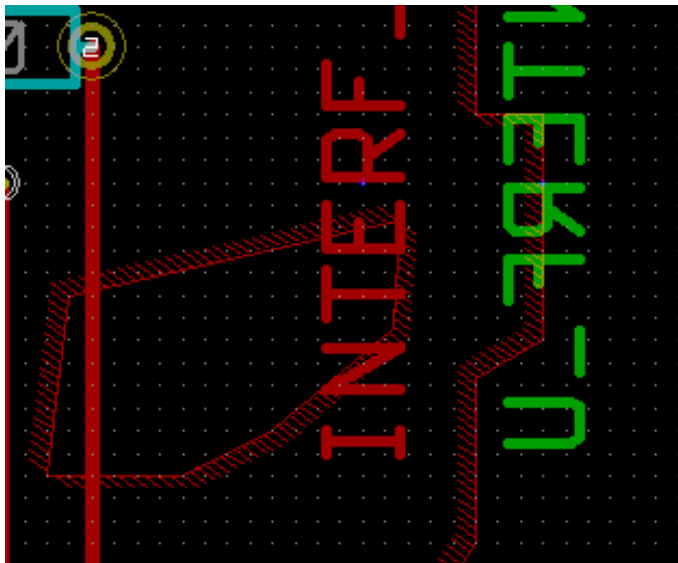
多边形结合在一起。

10.5.1 添加类似的区域

添加类似区域：



最后结果：




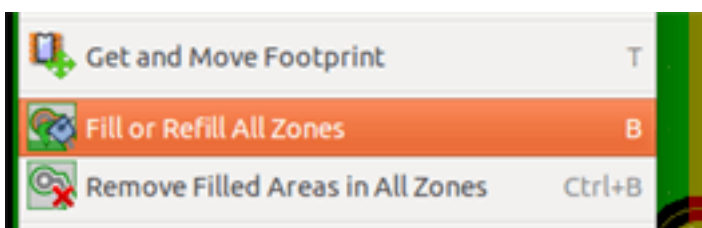
10.6 编辑区域参数

右键单击边框并使用“编辑区域参数”时，将打开“区域参数”对话框。可以输入初始参数。如果区域已经填满，则需要重新填充。

10.7 最后区域填充

完成电路板后，必须填写或重新填充所有区域。去做这个：

- 通过按钮图像激活工具区域：。
- 单击鼠标右键以显示弹出菜单。



- 使用填充或重新填充所有区域：



Warning

如果填充网格很小，计算可能需要一些时间。

10.8 更改区域网名

编辑原理图后，您可以更改任何网络的名称。例如，VCC 可以改为 +5V。

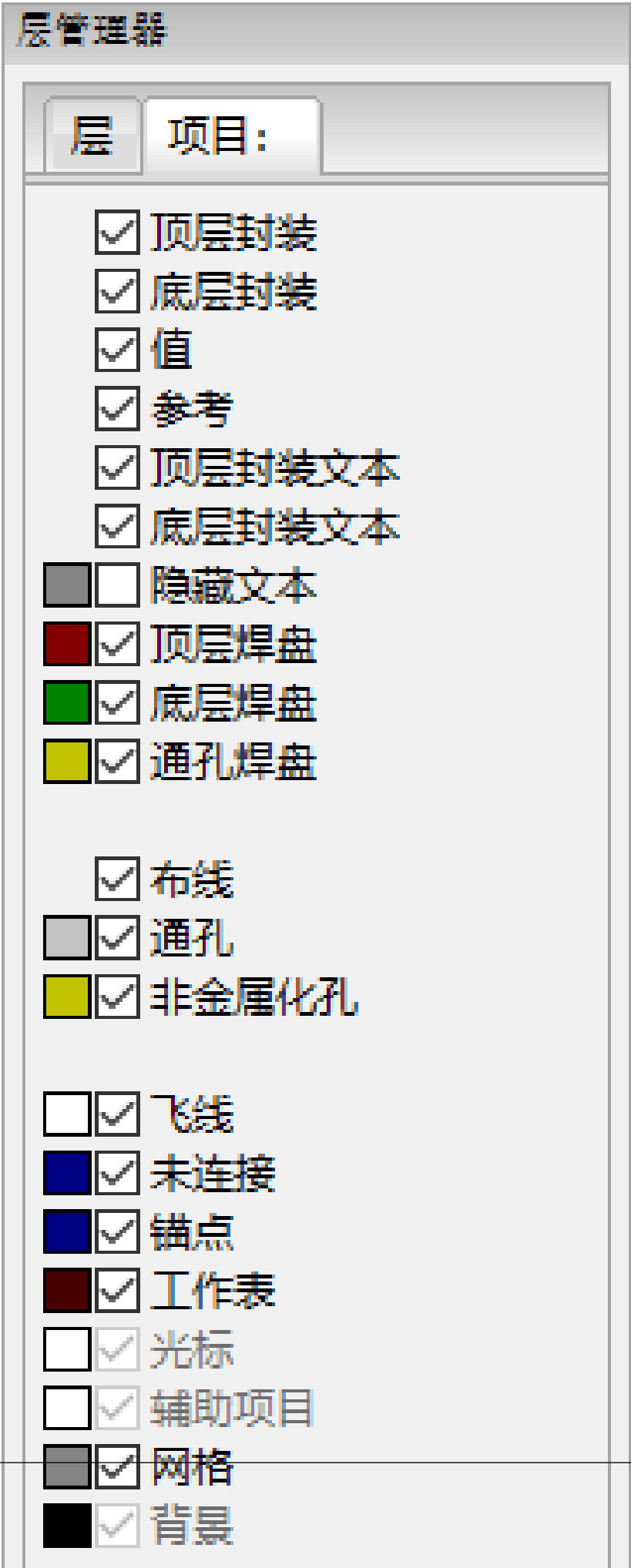
当创建全局 DRC 时，Pcbnew 检查区域网名是否存在，如果不存在则显示错误。
手动编辑区域参数对于将旧名称更改为新名称是必要的。

10.9 在技术层上创建区域

10.9.1 创建区域限制

这是使用按钮图像完成的：。活动层必须是技术层。

单击以启动区域大纲时，将打开此对话框：



选择技术层以放置区域并绘制区域边框，如前面针对铜层边框。

Note

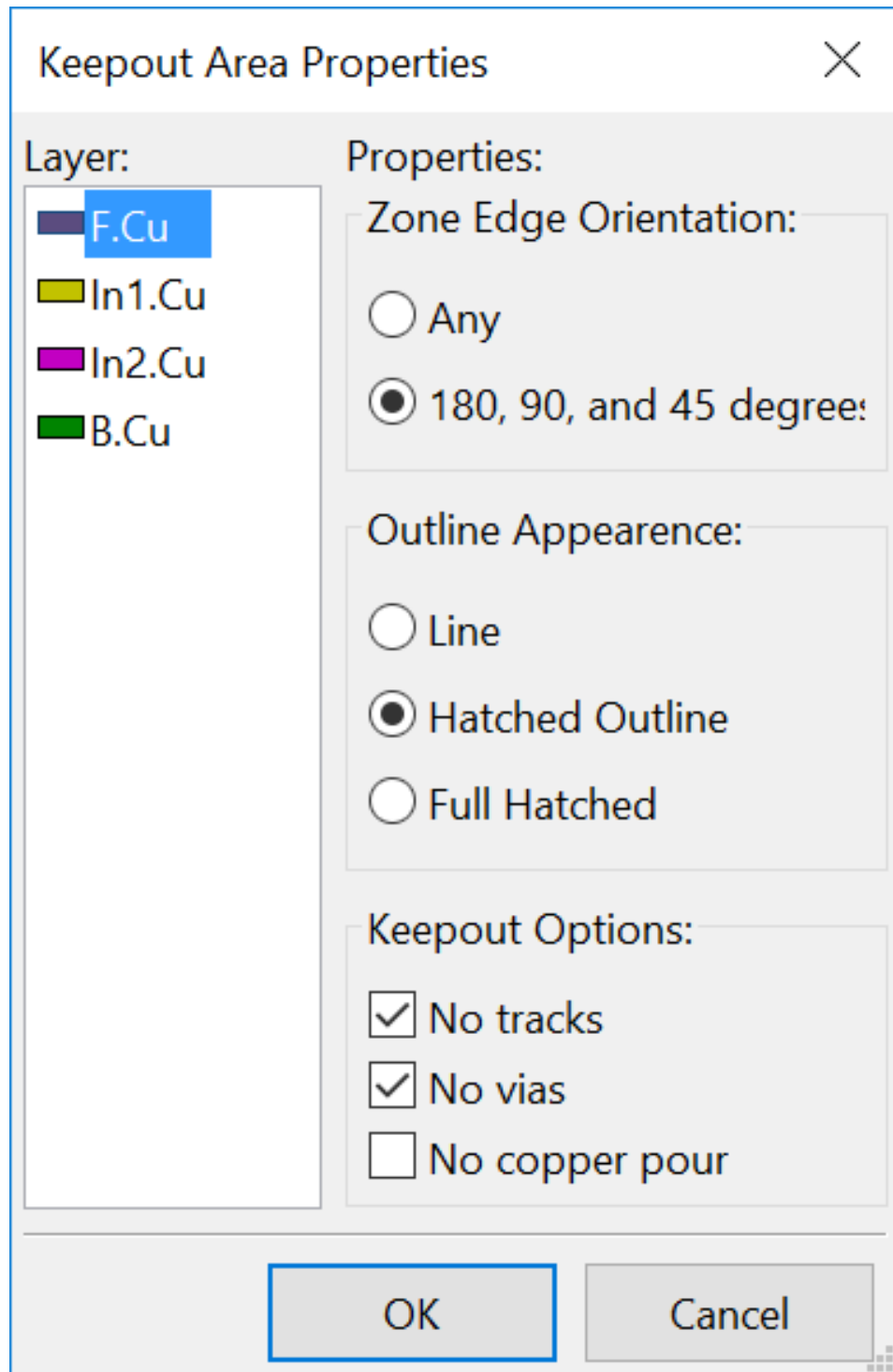
- 对于编辑边框，请使用与铜区相同的方法。
 - 如有必要，可以添加切口区域。
-

10.10 创建禁止布线区域

选择工具 

活动层应该是铜层。

单击新禁区的起点后，将打开对话框：



可以选择不允许的项目：

- 布线。
- 过孔。
- 敷铜。

当布线或过孔位于不允许它的禁区内时，将引发 DRC 错误。

对于铜区域，不会填充没有敷铜的禁区内的区域。禁区与区域类似，因此编辑其边框类似于铜区编辑。

Chapter 11

用于电路制造的文件

现在让我们看看为生产印刷电路板创建必要文件的步骤。

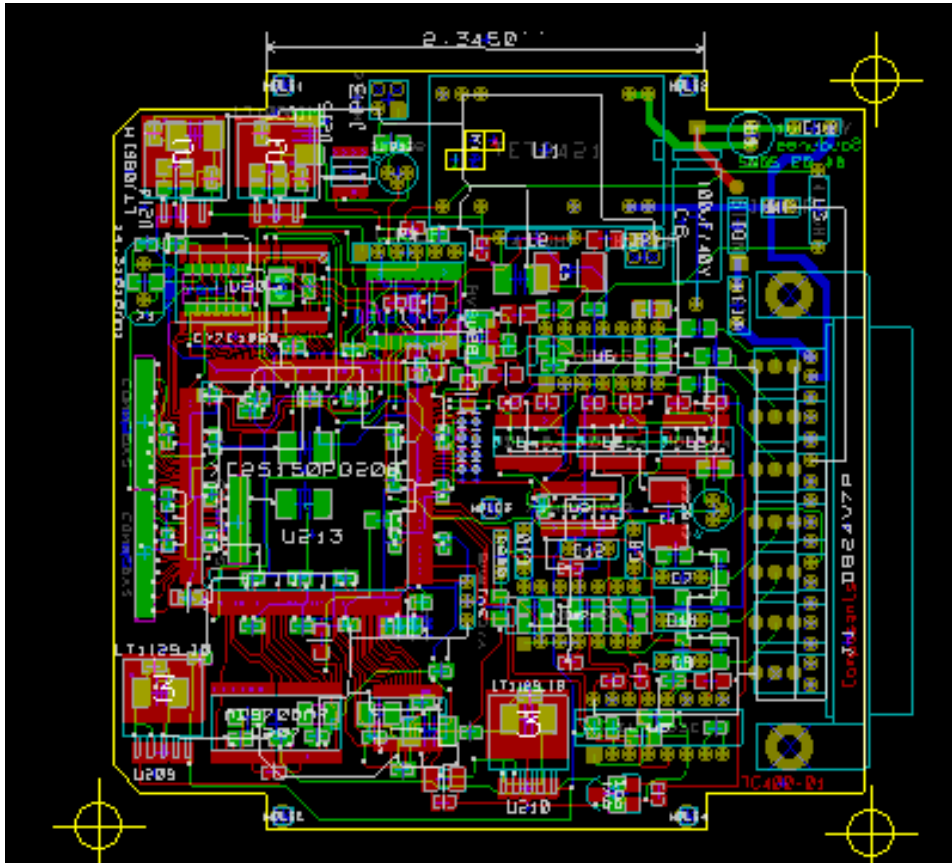
KiCad 生成的所有文件都放在工作目录中，该目录与包含印刷电路板的 xxxx.brd 文件的目录相同。

11.1 最后的准备

生成印刷电路板所需的文件包括以下准备步骤。

- 使用项目名称标记任何图层（例如，“顶部或前部”和“底部或后部”），方法是在每个图层上放置适当的文本。
- 铜层上的所有文本（有时称为“焊料”或“底部”）必须进行镜像。
- 创建任何地平面，根据需要修改布线以确保它们是连续的。
- 放置对齐十字准线和可能的板边框尺寸（这些通常放置在一个通用层上）。

下面是一个示例，显示除了地平面之外的所有这些元素，为了更好的可见性而省略了这些元素：




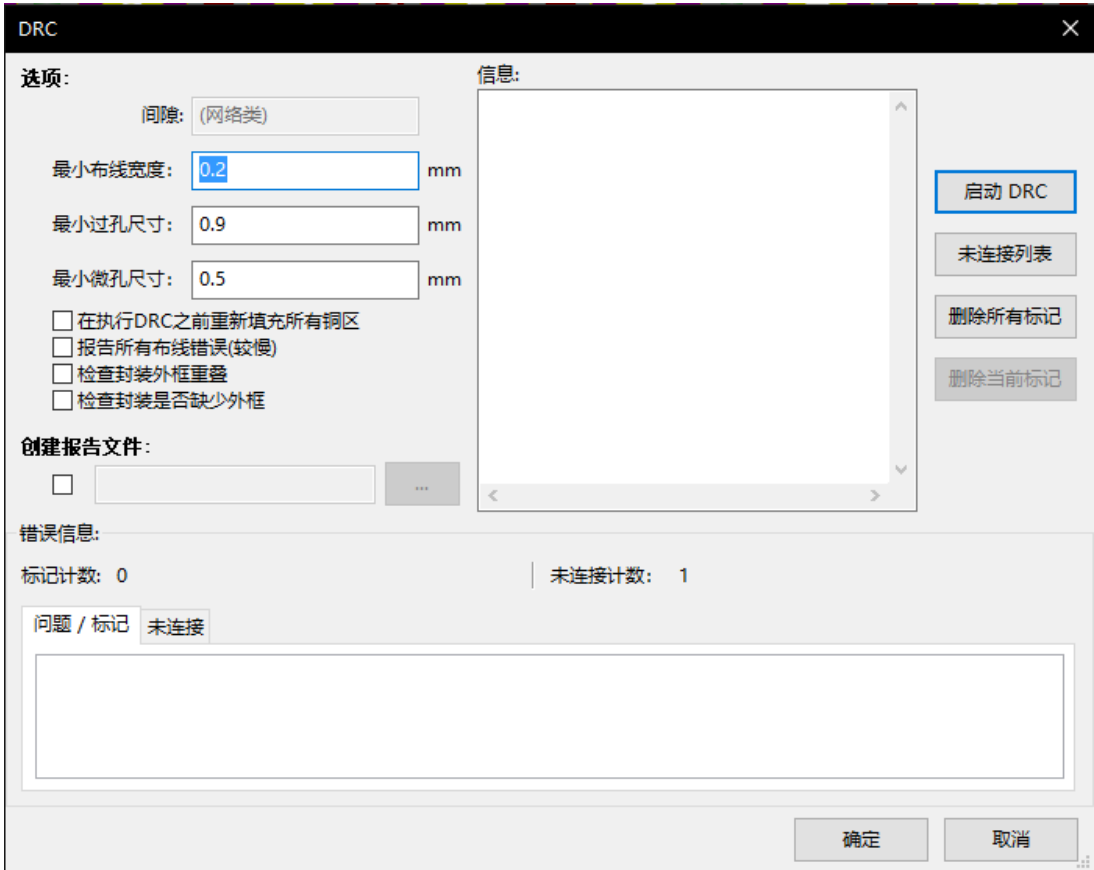
还包括 4 个铜层的颜色键：



11.2 最终的 DRC 测试


在生成输出文件之前，强烈建议进行全局 DRC 测试。

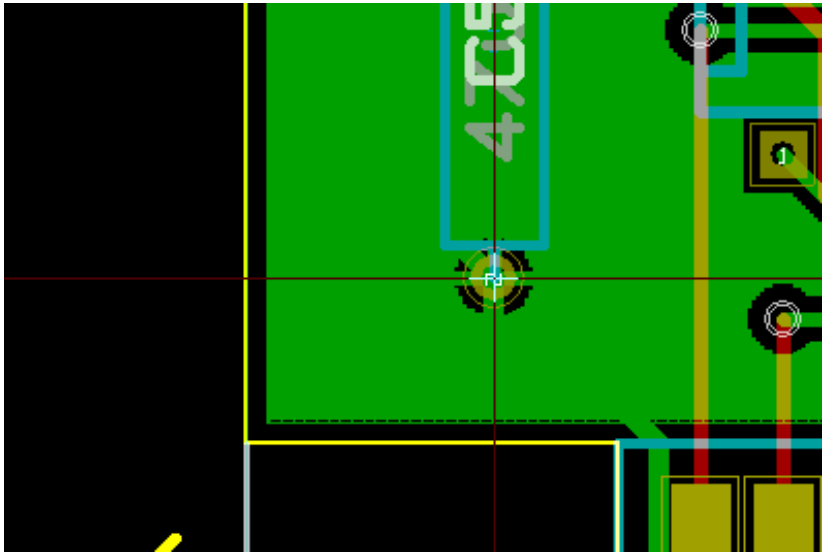
在启动 DRC 时填充或重新填充区域。按下按钮图像： 以启动以下 DRC 对话框：



相应地调整参数，然后按“开始 DRC”按钮。
最后的检查可以防止任何令人不快的意外。

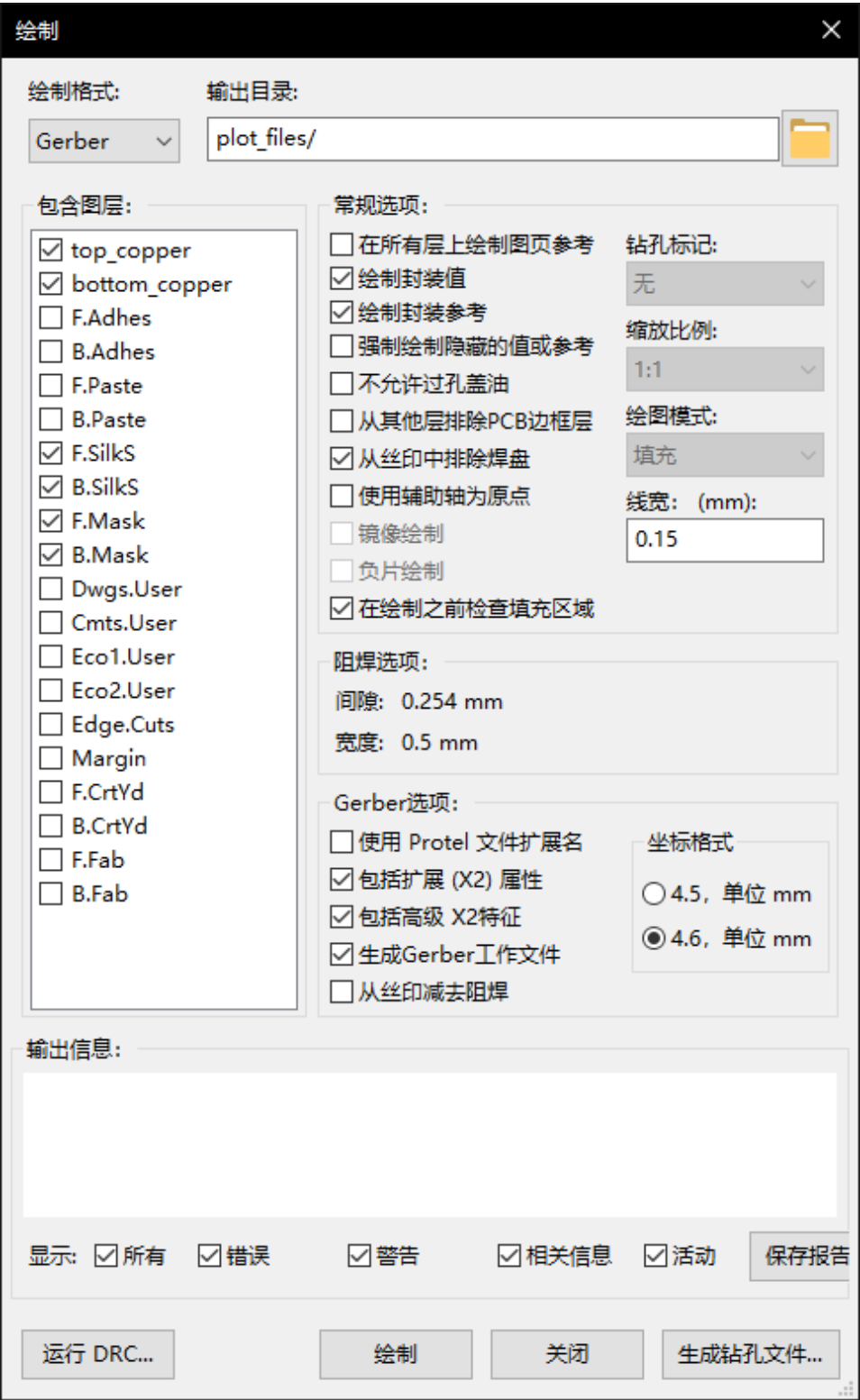
11.3 设置坐标原点

设置照片图和钻取文件的坐标原点，必须将辅助轴放在此原点上。激活图标图像：。通过左键单击所选位置来移动辅助轴。

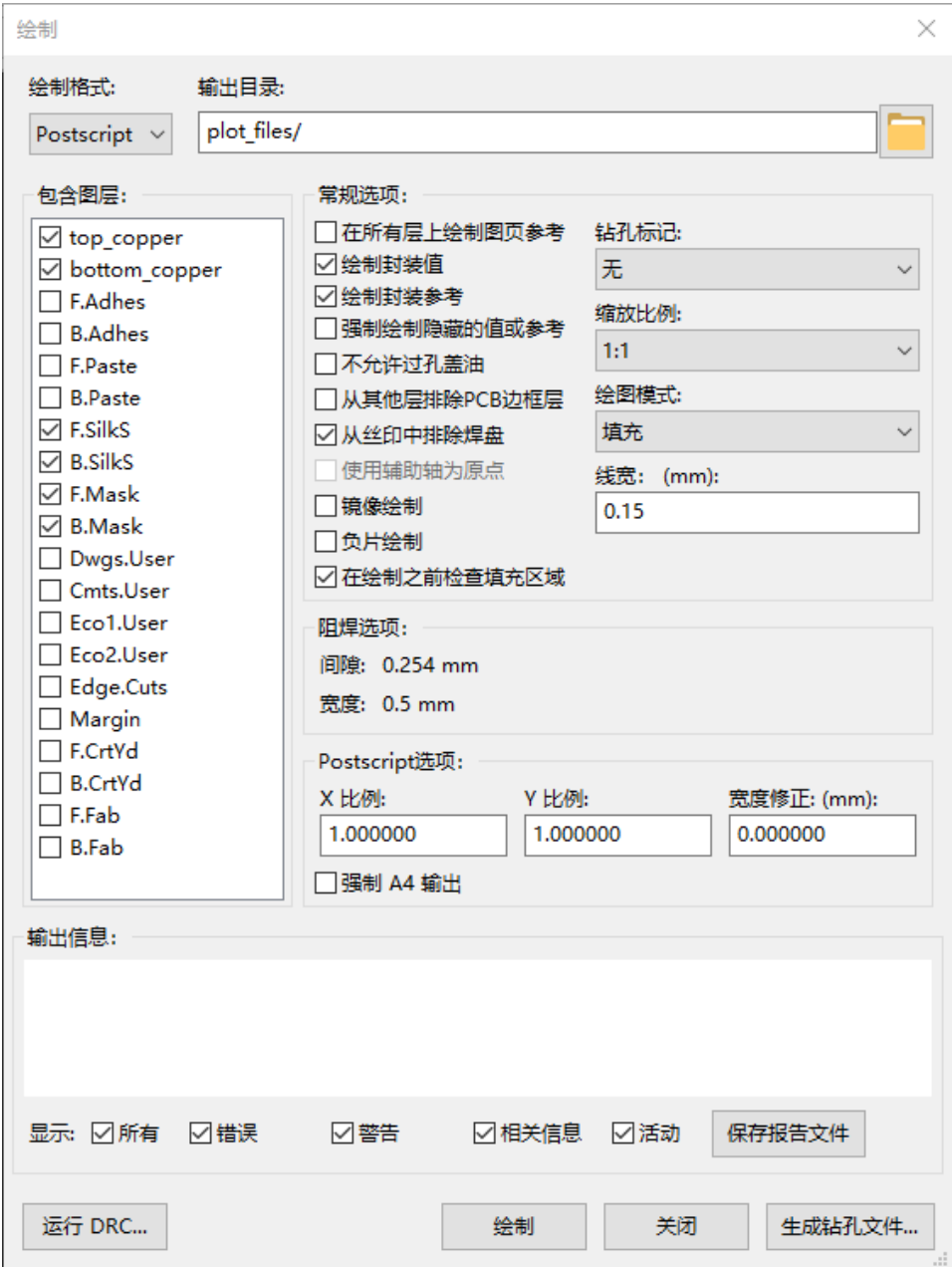


11.4 生成用于照片布线的文件

这是通过文件/绘图菜单选项完成的，并调用以下对话框：



通常，文件采用 GERBER 格式。然而，可以以 HPGL 和 POSTSCRIPT 格式生成输出。选择 Postscript 格式后，将出现此对话框。



在这些格式中，可以使用精细刻度调整来补偿绘图仪的精度，并使输出的真实比例为 1：



11.4.1 GERBER 格式

对于每一层，Pcbnew 按照 GERBER 274X 标准生成一个单独的文件，默认为 4.6 格式（文件中的每个坐标用 10 位数字表示，其中 4 位在小数点之前，6 位在它之后），单位为英寸，和 1 的比例。

通常需要为所有铜层创建文件，并根据电路为丝印层，阻焊层和焊膏层创建文件。通过选择相应的复选框，可以一步生成所有这些文件。

例如，对于带有丝印层，阻焊层和焊膏层（用于 SMD 元件）的双面电路，应生成 8 个文件（“xxxx”表示.brd 文件的名称）。

- 元件侧层的 xxxx-F_Cu.gbr。
- 铜侧层为 xxxx-B_Cu.gbr。
- 元件侧丝印层标记为 xxxx-F_SilkS.gbr。
- 铜侧丝印层标记为 xxxx-B_SilkS.gbr。
- 元件侧焊膏层为 xxxx-F_Paste.gbr。
- 铜侧焊膏层为 xxxx-B_Paste.gbr。
- 元件侧阻焊层为 xxxx-F_Mask.gbr。
- 铜侧阻焊层为 xxxx-B_Mask.gbr。

GERBER 文件格式：

Pcbnew 使用的格式是 RS274X 格式 4.6，英制，前导零省略，Abs 格式。这些是非常常见的设置。

11.4.2 POSTSCRIPT 格式

在 postscript 输出的情况下，输出文件的标准扩展名是.ps。对于 HPGL 输出，跟踪可以是用户选择的比例，并且可以进行镜像。如果原点 = 中心选项处于活动状态，则假定跟踪表坐标的原点位于图形的中心。

如果“打印纸张参考”选项处于活动状态，则会跟踪纸张盒。

11.4.3 绘图选项

Gerber 格式：

常规选项:

☐ 在所有层上绘制图页参考

☒ 绘制封装值

☒ 绘制封装参考

☐ 强制绘制隐藏的值或参考

☐ 不允许过孔盖油

☐ 从其他层排除PCB边框层

☒ 从丝印中排除焊盘

☐ 使用辅助轴为原点

☐ 镜像绘制

☐ 负片绘制

☒ 在绘制之前检查填充区域

钻孔标记:

无

缩放比例:

1:1

绘图模式:

填充

线宽: (mm):

0.15

阻焊选项:

间隙: 0.254 mm

宽度: 0.5 mm

Gerber选项:

☐ 使用 Protel 文件扩展名

☒ 包括扩展 (X2) 属性

☒ 包括高级 X2特征

☒ 生成Gerber工作文件

☐ 从丝印减去阻焊

坐标格式

☐ 4.5, 单位 mm

☒ 4.6, 单位 mm

其他格式:

常规选项:

☐ 在所有层上绘制图页参考

☒ 绘制封装值

☒ 绘制封装参考

☐ 强制绘制隐藏的值或参考

☐ 不允许过孔盖油

☐ 从其他层排除PCB边框层

☒ 从丝印中排除焊盘

☐ 使用辅助轴为原点

☐ 镜像绘制

☐ 负片绘制

☒ 在绘制之前检查填充区域

钻孔标记:

无

缩放比例:

1:1

绘图模式:

填充

线宽: (mm):

0.15

阻焊选项:

间隙: 0.254 mm

宽度: 0.5 mm

GERBER 格式特定选项:

使用 Protel 文件扩展名	使用 .gbl .gtl .gbs .gts .gbp .gtp .gbo .gto 代替 .gbr 进行文件扩展名。
包括扩展属性	将扩展属性输出到文件。
从丝印层中减去阻焊层	从焊膏区域去除所有丝印。

11.4.4 其他格式

标准扩展名取决于输出文件类型。

某些格式无法使用某些选项。

绘图可以在用户选择的比例下完成，并且可以进行镜像。

“打印钻孔选项”列表提供了填充，钻孔到正确直径或钻有小孔（用于指导手钻）的焊盘选项。

如果“打印纸张参考”选项处于活动状态，则会跟踪纸张盒。

11.5 阻焊层和焊膏层的全局间隙设置

可以为阻焊层和焊膏层全局设置掩模间隙值。这些间隙可以设置为以下级别。

- 在焊盘级。
- 在封装级。
- 在全局内。

并且 Pcbnew 按优先级顺序使用。

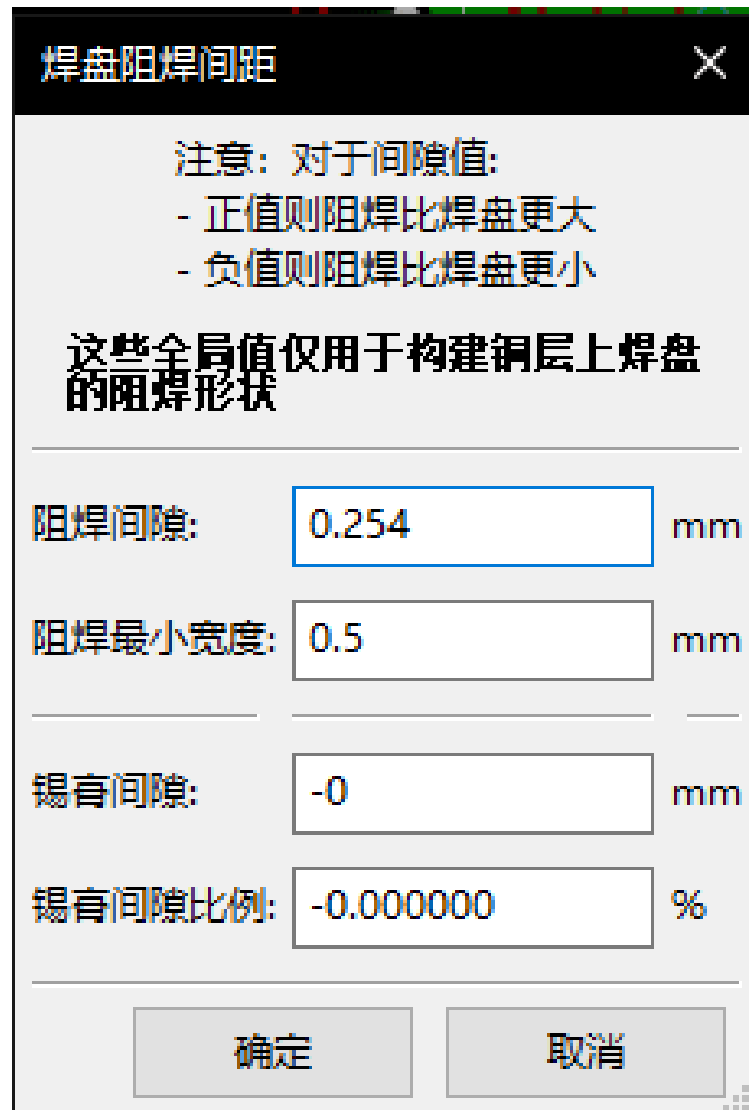
- 焊盘值。如果为空：
- 封装值。如果为空：
- 全局值。

11.5.1 访问

可以通过过孔“尺寸”菜单访问此菜单选项：



对话框如下：



11.5.2 阻焊层间隙

接近 0.2mm 的值通常是好的。该值为正，因为阻焊层通常大于焊盘。

可以在 2 个焊盘之间设置阻焊层宽度的最小值。

如果实际值小于最小值，则将合并 2 个阻焊层形状。

11.5.3 锡膏层间隙

最终间隙是焊膏层间隙和焊盘尺寸百分比的总和。

该值为负，因为焊膏层通常小于焊盘。

11.6 生成钻孔文件

始终需要遵循 EXCELLON 标准创建钻孔文件 xxxx.drl。

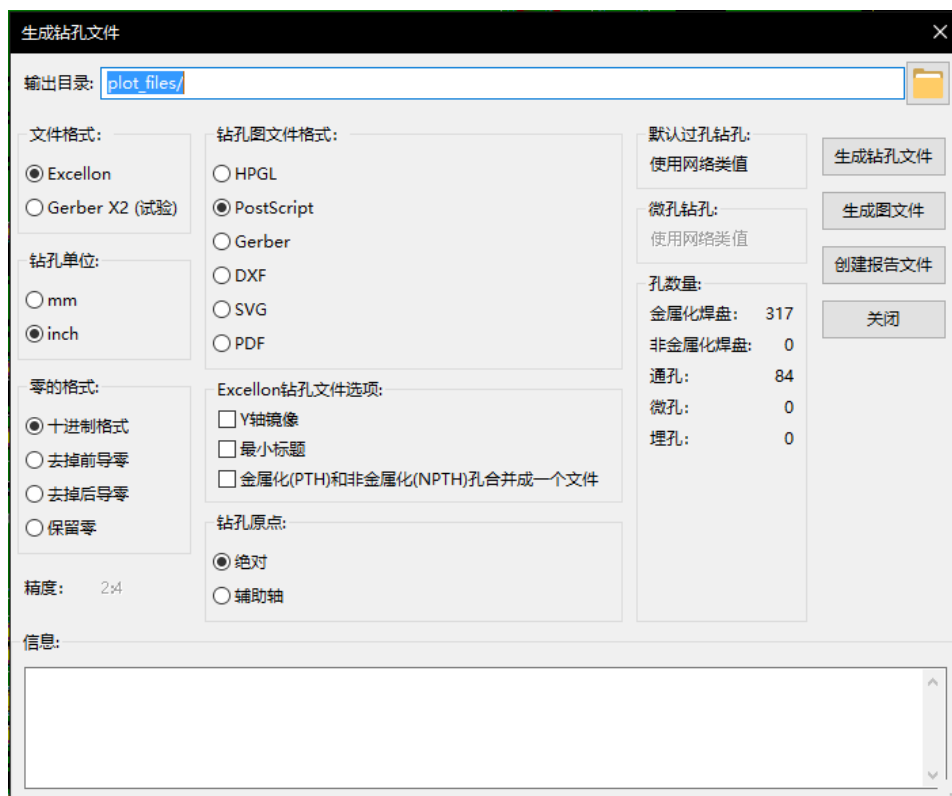
还可以生成可选的钻孔报告和可选的钻孔图。

- 可以使用多种格式绘制钻孔图。
- 钻孔报告是纯文本文件。

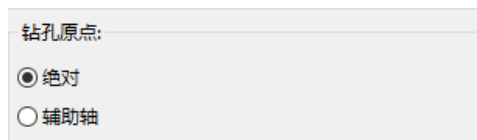
这些文件的生成通过以下方式控制：

- “创建钻孔文件”按钮，或
- 文件/制作输出/钻孔文件菜单选择。

“钻孔工具”对话框将如下所示：



要设置坐标原点，请使用以下对话框：



- 绝对：使用绝对坐标系。
- 辅助轴：坐标相对于辅助轴，使用图标（右侧工具栏）进行设置。

11.7 生成布线文档

要生成布线文档文件，可以跟踪组件和铜丝印层。通常，仅元件侧丝印标记足以连接 PCB。如果使用铜端丝印，则应对其包含的文本进行镜像以便可读。

11.8 生成用于自动元件插入的文件

可以通过后处理/创建 Cmp 文件菜单选项访问此选项。但是，除非至少有一个封装激活了普通 + 插入属性，否则不会生成任何文件（请参阅编辑封装）。将生成一个或两个文件，具体取决于 PCB 的一侧或两侧是否存在可插入元件。对话框将显示创建的文件的名称。

11.9 高级布线选项

下面描述的选项（文件/绘图对话框的一部分）允许对布线过程进行细粒度控制。在打印布线文档的丝印标记时，它们特别有用。

交互布线设置

模式:

☐ 高亮碰撞

☐ 推挤

☒ 绕走

选项:

鼠标拖动行为: 移动项

☐ 自由角度模式 (不推挤/绕走)

☐ 绕过障碍物

☒ 移除多余的布线

☒ 优化焊盘连接

☒ 平滑拖动线段

☐ 允许违反DRC规则

优化工作:

低

高

确定

取消

可用选项包括：

在所有图层上绘制工作表参考	布线表格边框和墨盒。
在丝印上绘制焊盘	启用/禁用丝印层上的焊盘边框印刷（如果焊盘已被宣布出现在这些图层上）。防止任何焊盘以禁用模式印刷。
绘制封装值	允许在丝印上打印 VALUE 文本。
绘制封装参考	允许在丝印上打印 REFERENCE 文本。
强制绘制不可见的 VALUE/REFERENCE	强制打印声明为不可见的字段（REFERENCE，VALUE）。结合‘绘制封装值’和‘绘制封装参考’，此选项可以生成用于指导接线和维修的文档。事实证明，这些选项对于使用元件的电路也是必要的 small（SMD）允许两个单独文本字段的可读放置。
不要盖住过孔	删除过孔上的掩层。
从其他层中排除 PCB 边缘层	GERBER 格式具体。不要在边缘图层上绘制图形项目。
使用 Protel 文件扩展名	GERBER 格式具体。创建文件时，请为每个文件使用特定扩展名。如果禁用，则 Gerber 文件扩展名为.gbr。

Chapter 12

封装编辑器 - 管理库

12.1 封装编辑器概述

Pcbnew 可以同时维护几个库。因此，当加载封装时，将搜索库列表中出现的所有库，直到找到封装的第一个实例。在下文中，请注意活动库是在封装编辑器中选择的库，现在将描述该程序

封装编辑器可以创建和编辑封装：

- 添加和删除焊盘。
- 更改单个焊盘的焊盘属性（形状，层）或全局覆盖焊盘的所有焊盘。
- 编辑图形元素（线条，文本）。
- 编辑信息字段（值，参考等）。
- 编辑相关文档（描述，关键字）。

封装编辑器允许维护活动库以及：


- 在活动库中列出封装。
- 从活动库中删除封装。
- 将封装保存到活动库。
- 保存印刷电路所包含的所有封装。

也可以创建新库。

库扩展名是 ‘.mod’。

12.2 访问封装编辑器

可以通过两种不同的方式访问封装编辑器：

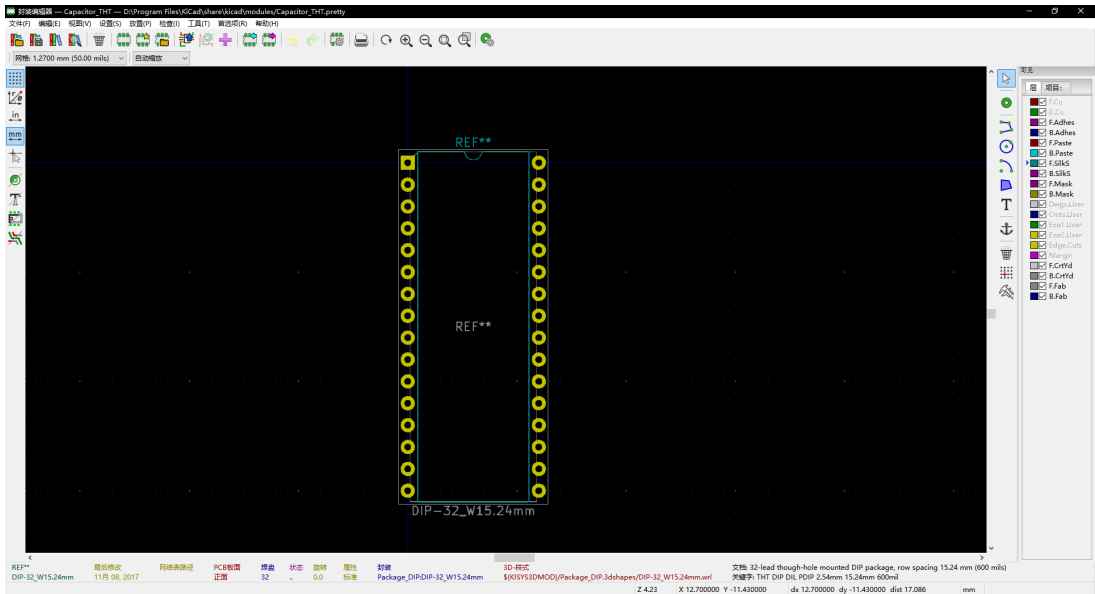
- 直接通过 Pcbnew 主工具栏中的图标。
- 在活动封装的编辑对话框中（参见下图：通过上下文菜单访问），有按钮封装编辑器。



在这种情况下，电路板的活动封装将自动加载到“封装编辑器”中，从而可以立即进行编辑或存档。

12.3 封装编辑器用户界面

通过调用封装编辑器，将出现以下窗口：










12.4 封装编辑器中的顶部工具栏





封装编辑器中的顶部工具栏从此工具栏中，可以使用以下功能：

	选择活动库。
	将当前封装保存到活动库，并将其写入磁盘。
	创建一个新库并将当前封装保存在其中。
	打开封装查看器
	访问用于从活动库中删除封装的对话框。
	创建新的封装。
	使用向导创建封装
	从活动库加载封装。
	从印刷电路板加载（导入）封装。
	将当前封装导出到印刷电路板。当封装先前是从当前的板导入的时候。它将取代板上相应的封装（即，尊重位置和取向）。
	将当前封装导出到印刷电路板。这将是复制到位置 0 的印刷电路板上。
	从“导出”命令创建的文件中导入封装。

	导出封装。此命令基本上与创建一个库，唯一的区别是它创建了一个库在用户目录中，在标准库中创建库时目录（通常是 kicad/modules）。
	撤消和重做
	调用封装属性对话框。
	调用打印对话框。
	标准缩放命令。
	调用焊盘编辑器。
	检查封装的正确性

12.5 创建一个新库


通过按钮图像创建新库：，在这种情况下，文件默认创建在库目录中或通过按钮，在这种情况下，默认情况下会在工作目录中创建该文件。

文件选择对话框允许指定库的名称并更改其目录。在这两种情况下，库都将包含正在编辑的封装。






Warning
如果存在同名的旧库，则会在没有警告的情况下覆盖它。


12.6 在活动库中保存封装

使用此按钮图像执行保存封装（从而修改活动库的文件）的操作：。如果已存在同名的封装，则将替换它。由于您将依赖于库封装的准确性，因此在保存之前值得仔细检查封装。



建议将引用或值字段文本编辑为库中标识的封装名称。

12.7 将封装从一个库转移到另一个库

- 通过按钮图像选择源库：。
- 通过按钮图像加载封装：。
- 通过按钮图像选择目标库：。

- 通过按钮图像保存封装：

您可能还希望删除源封装。

- 使用重新选择源库图像：
- 通过按钮图像删除旧的封装：

12.8 在活动库中保存电路板的所有封装

可以将给定电路板设计的所有封装复制到活动库中。这些封装将保留其当前的库名称。此命令有两个用途：

- 在丢失库的情况下，创建存档或使用板上的封装完成库。
- 更重要的是，它通过为库创建文档来促进库维护，如下所示。

12.9 库封装的文档

强烈建议记录您创建的封装，以便快速无误地进行搜索。

例如，谁能够记住 TO92 封装的所有多个引脚变体？“封装属性”对话框提供了此问题的简单解决方案。




该对话框接受：

- 一行注释/描述。
- 多个关键字。

该描述与 Cvpcb 中的元件列表一起显示，并且在 Pcbnew 中，它在封装选择对话框中使用。

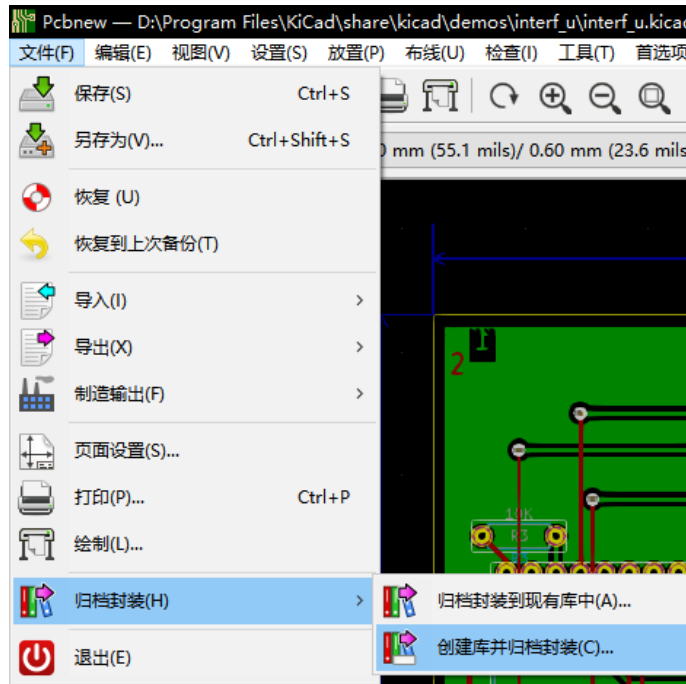
关键字使搜索仅限于与特定关键字对应的那些封装。

当直接加载封装（右侧 Pcbnew 工具栏的图标图像：)时，可以在对话框中输入关键字。因此，输入文本 ‘=CONN’ 将导致显示其关键字列表包含单词 “CONN” 的封装列表。

12.10 记录库 - 推荐的做法

建议间接创建库，方法是创建一个或多个辅助电路板，构成库的（部分）源，如下所示：创建 A4 格式的电路板，以便能够轻松地进行打印（scale = 1）。

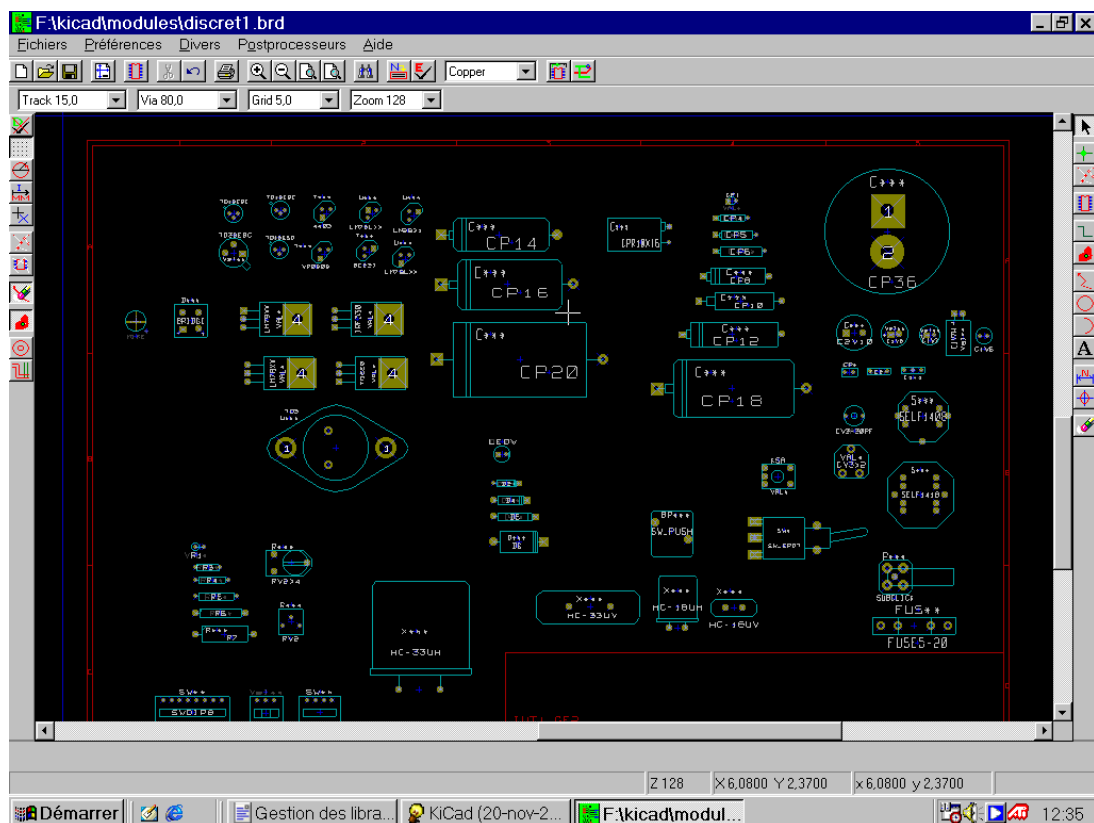
创建库将包含在此电路板上的封装。将使用文件/存档封装/创建封装存档命令创建库本身。



因此，库的“真实来源”将是辅助电路板，并且在该电路板上将进行任何随后的封装改变。当然，几个电路板可以保存在同一个库中。

为不同类型的组件（连接器，分立器件……）制作不同的库通常是个好主意，因为 Pcbnew 能够在加载封装时搜索许多库。

以下是此类库源的示例：



这种技术有几个优点：

- 该电路可以按比例打印并作为库的文档而无需进一步努力。
- Pcbnew 的未来变化可能需要重新生成库，如果使用了这种类型的电路板源，则可以非常快速地完成。这很重要，因为电路板文件格式在未来的开发过程中保证兼容，但库文件格式不是这种情况。

12.11 封装库管理

可以使用封装库管理编辑 Pcbnew 中的封装库列表。这允许您手动添加和删除封装库，还允许您通过按“附加向导”按钮调用封装库向导。

封装库向导也可以通过“首选项”菜单调用，并可以从文件或 Github URL 自动添加库（检测其类型）。官方库的 URL 是: <https://github.com/KiCad>

有关封装库表和管理器和向导的更多详细信息，请参阅 CvPcb 参考手册的 *Footprint Library Tables* 部分。

12.12 3D 形状库管理

3D 形状库可以通过 3D 形状库向导下载。它可以从菜单首选项 - > 3D 形状库下载程序中调用。

Chapter 13

封装编辑器 - 创建和编辑封装

13.1 封装编辑器概述

封装编辑器用于编辑和创建 PCB 封装。这包括：

- 添加和删除焊盘。
- 封装编辑器用于编辑和更改焊盘属性（形状，层），用于单个焊盘或封装中的所有焊盘。这包括创建 PCB 封装。
- 添加和编辑图形元素（边框，文本）。
- 编辑字段（值，参考等）。
- 编辑相关文档（描述，关键字）。

13.2 封装元素

封装是要插入 PCB 中的元件的物理表示（封装），并且必须链接到原理图中的相关元件。每个封装包括三个不同的元素：

- 焊盘。
- 图形边框和文本。
- 字段。

此外，如果将使用自动放置功能，则必须正确定义许多其他参数。这同样适用于生成自动插入文件。

13.2.1 焊盘

两个焊盘属性很重要：

- 几何形状（形状，层，钻孔）。
-

- 焊盘编号，由最多四个字母数字字符组成。因此，以下都是有效的垫号：1,45 和 9999，还有 AA56 和 ANOD。焊盘编号必须与原理图中相应引脚编号的焊盘编号相同，因为它定义了 Pcbnew 链接引脚和焊盘的匹配引脚和焊盘编号。

13.2.2 边框


图形边框用于绘制封装的物理形状。有几种不同类型的边框可供选择：直线，圆，圆弧和文本。边框没有电学意义，它们只是图形辅助工具。

13.2.3 字段

这些是与封装相关联的文本元素。两个是强制性的并且始终存在：参考字段和值字段。在将封装加载到您的电路板期间读取网表时，Pcbnew 会自动读取和更新这些内容。参考值由适当的原理图参考（U1，IC3 等）代替。该值将替换为原理图中相应部件的值（47K，74LS02 等）。可以添加其他字段，这些字段的行为类似于图形文本。

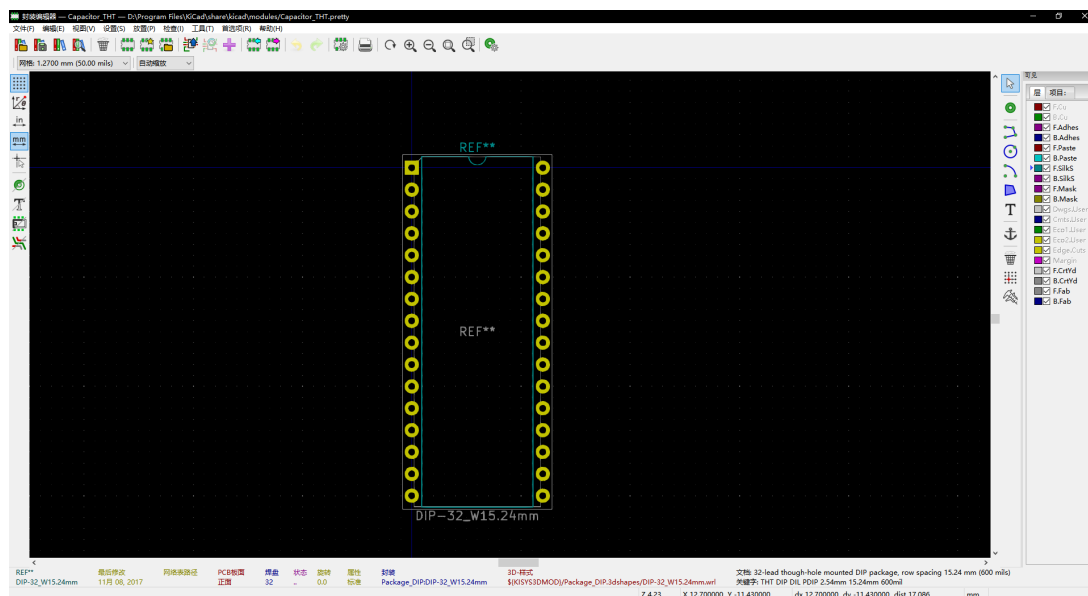
13.3 启动封装编辑器并选择要编辑的封装

封装编辑器可以通过两种方式启动：

- 直接通过图片：Pcbnew 主工具栏中的  图标。这允许创建或修改库中的封装。
- 双击封装将启动“封装属性”菜单，该菜单提供“转到封装编辑器”按钮。如果使用此选项，则电路板的占用空间将加载到编辑器中，以进行修改或保存。

13.4 封装编辑器工具栏

调用封装编辑器将启动一个如下所示的新窗口：










13.4.1 编辑工具栏 (右侧)

此工具栏包含以下工具：








- 放置焊盘。
- 添加图形元素（边框，文本）。
- 定位锚点。
- 删除元素。

具体功能如下：

	没有工具。
	添加焊盘。
	绘制线段和多边形。
	画圆圈。
	绘制圆弧。
	添加图形文本（此工具不管理字段）。
	定位封装锚点。
	删除元素。
	网格起源。(网格偏移)。适用于焊盘的放置。网格原点可以放在给定的位置（第一个要放置的焊盘），并且可以将网格大小设置为焊盘间距。因此放置焊盘非常容易

13.4.2 显示工具栏 (左侧)

这些工具管理封装编辑器中的显示选项：

	显示网格。
	显示极坐标。
	使用毫米或英寸的单位
	切换光标十字线形状
	以边框模式显示焊盘。
	以边框模式显示文本。
	以边框模式显示边框。

	切换高对比度模式
---	----------

13.5 上下文菜单

鼠标右键调出依赖于光标下方元素的菜单。

用于编辑封装参数的上下文菜单：



	移动	M
	逆时针旋转	R
	顺时针旋转	Shift+R
	翻转	F
	删除	Backspace
	属性...	E
	精确移动...	Ctrl+M
	相对位置...	Ctrl+R
	重复	Ctrl+D
	创建阵列...	Ctrl+T
	复制	
	剪切	
	粘贴	
	用封装编辑打开	Ctrl+E
	更新封装...	
	修改封装...	
	选择	>
	锁定	>
	居中	F4
	放大	F1
	缩小	F2
	自动缩放	Home
	缩放	>
	网格	>

3D-模

zontal

\$(KI)

X 229.870000 Y 8

编辑焊盘的上下文菜单:

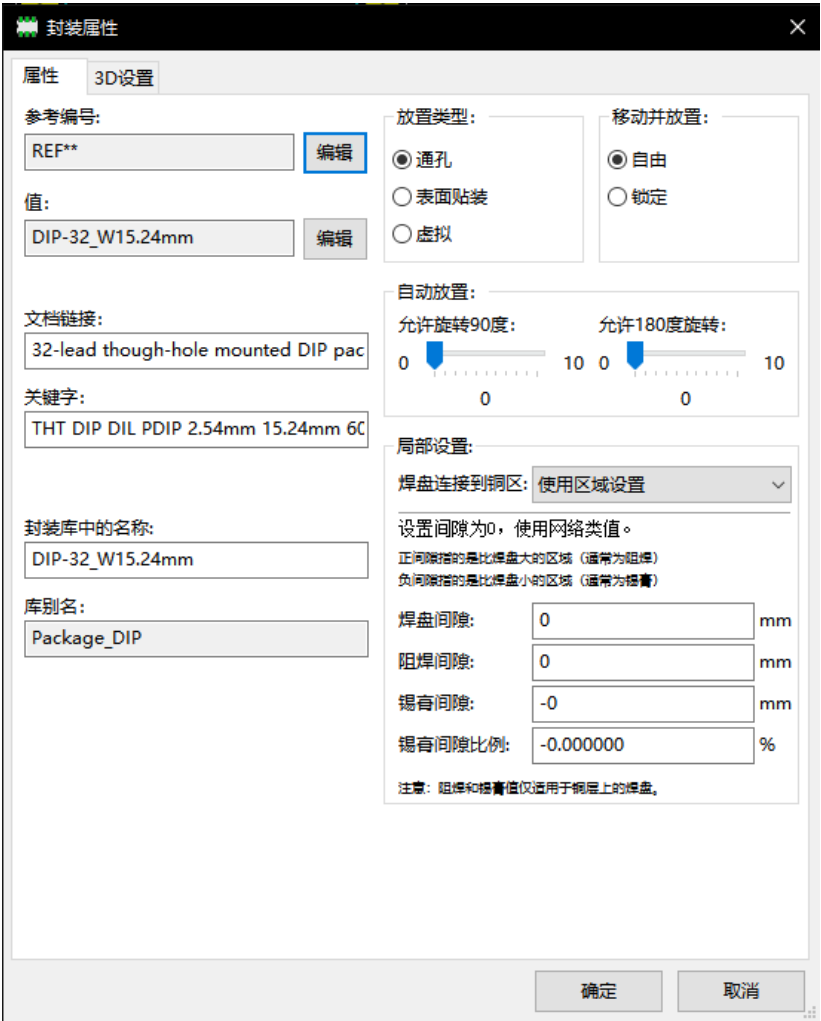


用于编辑图形元素的上下文菜单:




13.6 封装属性对话框

当光标位于封装上时，可以通过单击鼠标右键然后选择“编辑封装”来启动此对话框。



该对话框可用于定义主要封装参数。

13.7 创建新的封装




可以通过按钮图像创建新的封装：。将要求新封装的名称。这将在库中标识封装的名称。该文本还用作封装值，最终由实际值（100 F_16V，100Ω_0.5W，…）代替。

新的封装将需要：

- 边框（可能还有图形文字）。
- 焊盘。
- 一个值（隐藏文本在使用时由真值替换）。

替代方法：


当新的封装类似于库或电路板中的现有封装时，创建新封装的另一种更快的方法如下：

- 加载类似的封装 (,  或 )。
- 修改“库中的封装”字段以生成新标识符 (名称)。
- 编辑并保存新的封装。

13.8 添加和编辑焊盘

创建封装后，可以添加，删除或修改焊盘。焊盘的修改可以是局部的，仅影响光标下的焊盘或全局，影响封装的所有焊盘。


13.8.1 添加焊盘

从右侧工具栏中选择图像： 图标。可以使用鼠标左键单击所需位置来添加焊盘。焊盘属性在焊盘属性菜单中预定义。

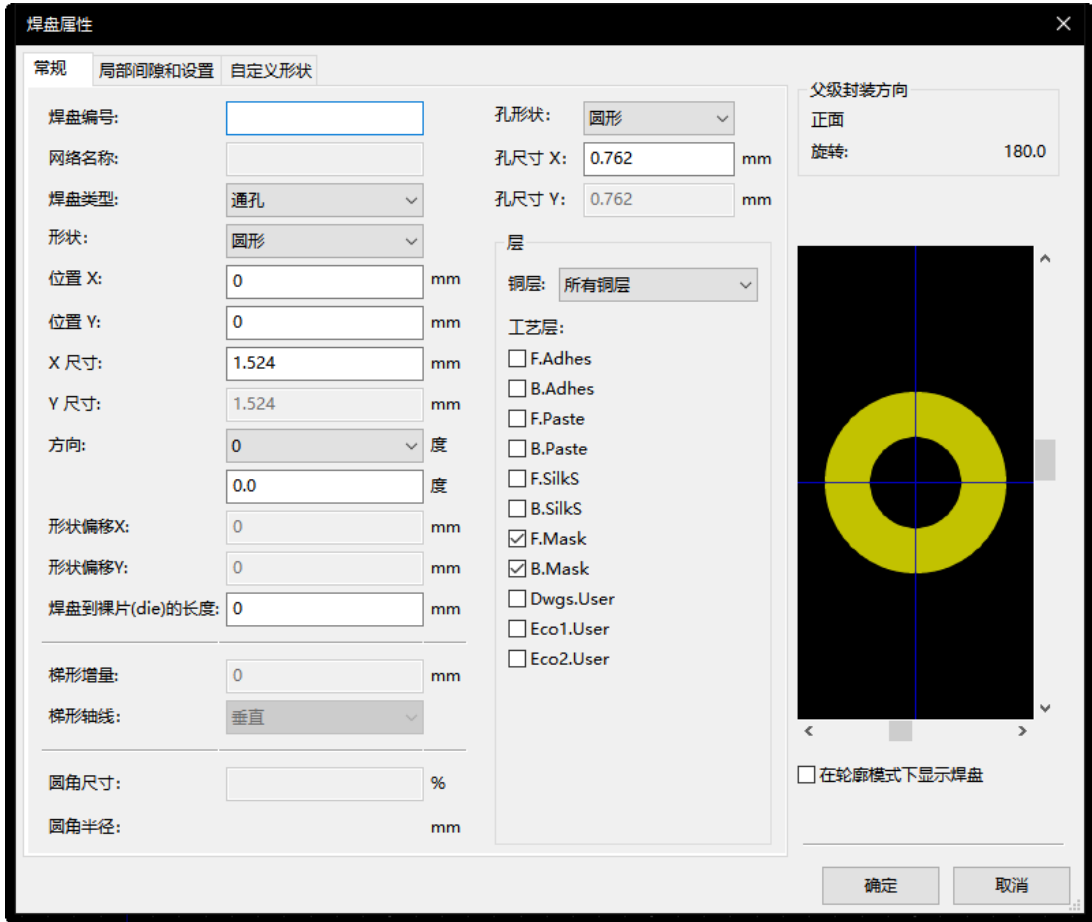
不要忘记输入焊盘编号。

13.8.2 设置焊盘属性

这可以通过三种不同的方式来实现：

- 从水平工具栏中选择图像： 。
- 单击现有焊盘并选择“编辑焊盘”。然后可以编辑焊盘的设置。
- 单击现有焊盘并选择“导出焊盘设置”。在这种情况下，所选焊盘的几何属性将成为默认焊盘属性。

在前两种情况下，将显示以下对话框窗口：



应注意正确定义焊盘所属的层。特别是，尽管铜层易于定义，但非铜层（阻焊层，焊盘……）的管理对于电路制造和文档记录同样重要。

焊盘类型选择器触发通常足够的图层自动选择。

13.8.2.1 矩形焊盘

对于 VQFP/PQFP 类型的 SMD 封装，其四边都有矩形焊盘（水平和垂直），建议仅使用一种形状（例如，水平矩形）并将其放置在不同的方向（水平为 0 和 90 度垂直）。然后可以在单个操作中完成焊盘的全局尺寸调整。

13.8.2.2 旋转焊盘

微波封装中使用的梯形焊盘只需要 -90 或 -180 的旋转。

13.8.2.3 无镀通孔焊盘

焊盘可以定义为非镀通孔焊盘（NPTH 焊盘）。

这些焊盘必须在一个或所有铜层上定义（显然，所有铜层上都有孔）。

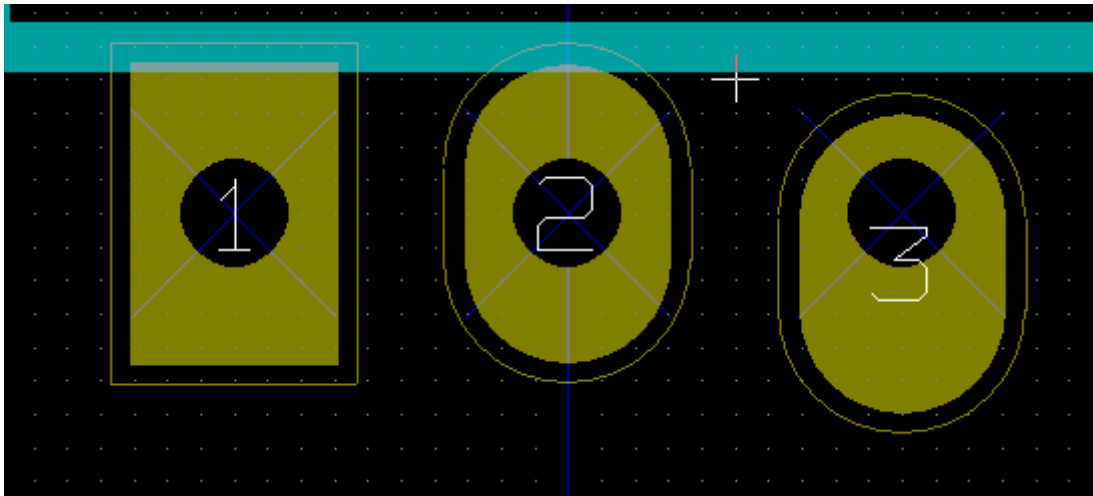
此要求允许您定义特定的间隙参数（例如螺钉的间隙）。

当焊盘通孔尺寸与焊盘尺寸相同时，对于圆形或椭圆形焊盘，此焊盘不会在 GERBER 文件中的铜层上绘制。

这些焊盘用于机械目的，因此不允许使用焊盘名称或网名。无法连接到网络。

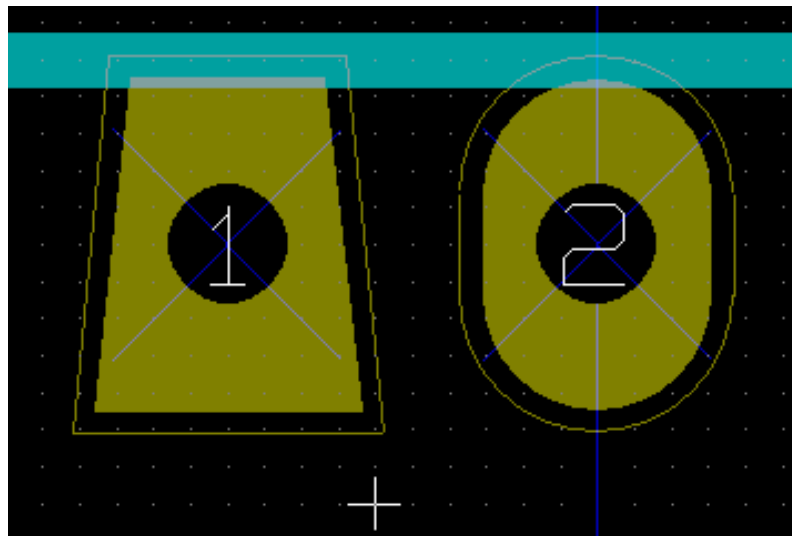
13.8.2.4 偏移参数

焊盘 3 的偏移 $Y = 15$ mils:



13.8.2.5 Delta (希腊字母: 得尔塔) 参数 (梯形焊盘)

焊盘 1 的参数 Delta (希腊字母: 得尔塔) $X = 10$ mils



13.8.3 设置阻焊层和焊膏层的间隙

在定义包含铜层的焊盘时, KiCad 会根据固定间隙和/或焊盘几何形状的比例创建阻焊层和焊膏层。用于计算最终焊盘尺寸的非零设置基于以下优先顺序:

- 焊盘设置
- 封装设置
- 全局设置

Note

阻焊层焊盘的形状通常比焊盘本身大。所以间隙值是正的。锡膏层焊盘的形状通常小于焊盘本身。所以间隙值是负的。

13.8.3.1 焊膏层设置

两个设置用于计算焊膏孔径：

- 固定间隙设置。
- 焊盘尺寸的百分比。

最终值是比率设置和间隙设置的乘积。

13.8.4 焊盘不在铜层上

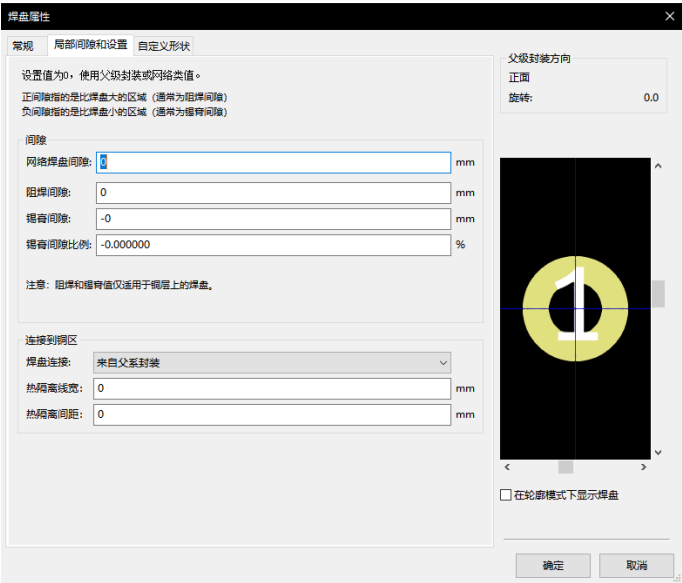
存在第二种用于创建没有定义任何铜层的焊盘的方法。这些焊盘通常被称为孔焊盘，并且可用于创建不基于铜焊盘几何形状边框的定制孔。此方法是在 5.0.0-rc2 版本中引入的。没有任何铜层定义的焊盘忽略全局和占位级别设置，仅使用焊盘级别设置。



Warning

使用版本 5.0.0-rc2 之前定义的没有铜层的焊盘使用上面定义的优先级和全局和占位设置进行绘制。必须对此版本之前设计的任何电路板进行调整，以获得相同的输出图。

封装级别设置：



焊盘级别设置：

设置间隙为0，使用网络类值。

正间隙指的是比焊盘大的区域（通常为阻焊）

负间隙指的是比焊盘小的区域（通常为锡膏）

焊盘间隙:	<input type="text" value="0"/>	mm
阻焊间隙:	<input type="text" value="0"/>	mm
锡膏间隙:	<input type="text" value="-0"/>	mm
锡膏间隙比例:	<input type="text" value="-0.000000"/>	%

注意：阻焊和锡膏值仅适用于铜层上的焊盘。

13.9 字段属性

至少有两个字段：引用和值。

必须更新它们的参数（属性，大小，宽度）。您可以通过双击该字段或封装属性对话框从弹出菜单访问该对话框：

封装文本属性

封装 REF** (C_Axial_L22.0mm_D9.5mm_P27.50mm_Horizontal) 方向 0.0

值:

宽度: (mm):

高度: (mm):

线宽: (mm):

偏移 X: (mm):

偏移 Y: (mm):

层:

F.Fab

显示:
☒ 可见
☐ 不可见

样式:
☒ 标准
☐ 斜体

方向:
☒ 0.0
☐ +90.0
☐ -90.0
☐ 180.0
☐ 其它

旋转 (-180.0度 至 180.0度)

☐ 解锁文本方向

确定

取消

13.10 自动放置封装

如果用户希望利用自动放置功能的全部功能，则必须定义封装的允许方向（“封装属性”对话框）。



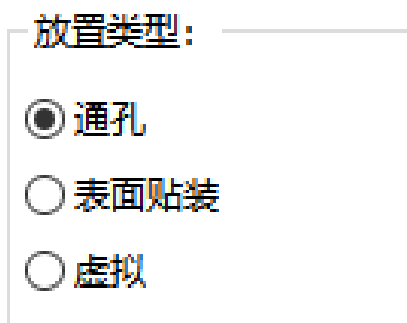
通常，电阻器，非极化电容器和其他对称元件允许旋转 180 度。

一些封装（例如小晶体管）通常允许旋转 +/- 90 或 180 度。默认情况下，新的封装将其旋转权限设置为零。这可以根据以下规则进行调整：

值 0 表示无法旋转，10 表示完全旋转，任何中间值表示旋转受限。例如，电阻器可能具有 10 度旋转 180 度（不受限制）的许可，以及 5 度允许旋转 +/- 90 度（允许但不鼓励）。

13.11 属性

属性窗口如下：




- 普通是标准属性。
- 普通 + 插入表示封装必须出现在自动插入文件中（对于自动插入机器）。此属性对表面贴装元件（SMD）最有用。
- 虚拟表示元件由电路板直接形成。例子是由特定布线形状（有时在微波封装中看到）创建的边缘连接器或电感器。

13.12 库封装的文档

强烈建议记录新创建的封装，以便于快速准确地检索。谁能够回忆起 TO92 封装的多种引脚变体？

“封装属性”对话框提供了一种简单而强大的文档生成方法。

 封装属性

属性

3D设置

参考编号:

REF**

编辑

值:

DIP-32_W15.24mm

编辑

文档链接:

32-lead though-hole mounted DIP pac

关键字:

THT DIP DIL PDIP 2.54mm 15.24mm 6C

封装库中的名称:

DIP-32_W15.24mm

库别名:

Package_DIP

此菜单允许：

- 注释行的输入（描述）。
- 多个关键字。

注释行与 CvPcb 中的元件列表和 Pcbnew 中的封装选择菜单一起显示。关键字可用于限制搜索具有给定关键字的那些部分。

因此，在使用加载封装命令（Pcbnew 右侧工具栏中的图标）时，可以在对话框中键入文本 “= TO220”，让 Pcbnew 显示拥有关键字 “TO220” 的封装列表

13.13 三维可视化

封装可以与包含元件的三维表示的文件相关联。要将封装与模型文件相关联，请选择 *3D 设置* 选项卡，如下所示。

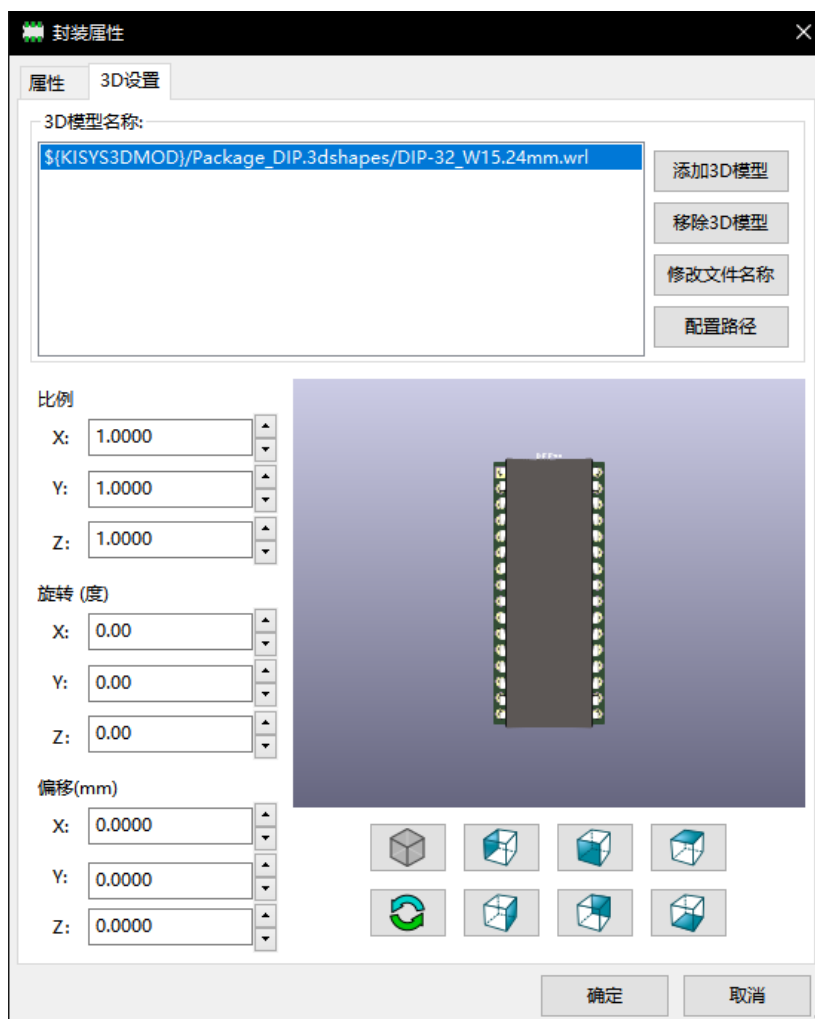


Figure 13.1: 3D 模型选择界面

右侧的按钮具有以下功能：

- **添加 3D 形状**显示 3D 文件选择对话框并创建一个元件的新模型条目。
- **删除 3D 形状**删除所选的模型条目。
- **编辑文件名**显示用于手动输入的文本编辑器模型文件名。
- **配置路径**显示一个配置对话框允许用户编辑路径别名列表和别名值。

3D 设置 选项卡包含一个面板，其中包含所选模型的预览以及模型的比例，偏移和旋转数据。

缩放值对于可视化格式（如 VRML1，VRML2 和 X3D）非常有用。由于模型可能由任意数量的 VRML/X3D 编辑器或导出器生成，并且 VRML 不强制模型的长度单位，因此用户可以输入适当的比例值以确保模型在 3D 查看器中显

示。一些用户使用简单的 VRML 框作为元件的通用模型并选择比例值，以便框具有表示组件的正确大小。对于机械 CAD (MCAD) 模型，比例值应保持为 1。MCAD 格式始终指定单位长度，任何使用 MCAD 数据格式的导出器都将忽略缩放值。但是 3D 查看器将始终应用比例值；如果除了单位以外的比例值与 MCAD 模型一起使用，则 3D 查看器的输出将与任何导出的 MCAD 模型（如 IDF）不同。

通常需要偏移和旋转值以使 3D 模型与封装对齐。由于 3D 建模软件的差异以及用户构建模型的方式不同，在绝大多数情况下，用户必须输入偏移和旋转值才能实现 3D 模型的所需定位。旋转值以度为单位，并以 ZYX 的顺序连续应用；所使用的惯例是当从轴的正位置朝向原点观察时，正角度导致部件顺时针旋转。

KiCad 通过插件系统支持 3D 模型格式，并为视觉模型格式 VRML1, VRML2 和 X3D 以及 MCAD 格式 IDF 提供支持。MCAD 格式 IGES 和 STEP 通过 OCE 插件支持，该插件需要合适版本的 OpenCascade 或 OpenCascade Community Edition (OCE) 软件。

13.13.1 3D 模型路径

在过去，KiCad 使用固定路径到 3D 模型目录，后来依靠 *KISYS3DMOD* 环境变量来指定模型目录的位置。可以使用其他环境变量指定模型的其他基目录。当前版本的 KiCad 具有专门的 别名系统，用于处理 3D 模型名称。新文件名管理系统（文件名解析系统）的目的是提供一种与早期版本的 KiCad 兼容的方案，同时提供更灵活的机制来指定 3D 模型文件名并提高共享项目文件的能力。

由于需要支持先前的方案，同时提供用于查找 3D 模型的灵活的新方案，因此有两种不同的方法来指定 3D 模型的基本搜索路径。

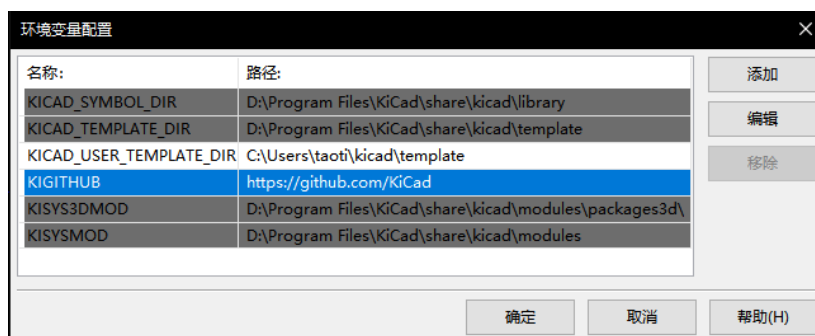


Figure 13.2: KiCad 路径配置对话框

支持缩短文件名的新方案是 别名系统。在这个系统中，一个路径以字符串 *:my alias* 开头：其中 *my alias* 是一个文本字符串，最好选择为短，同时对用户也很重要；例如，包含官方 KiCad 模型的目录的别名可能具有别名 *官方模型*，而您的个人模型集合可能具有别名 *My Models*。可以通过单击前面显示的 **3D 设置** 选项卡中的 **配置路径** 按钮来设置别名。别名配置对话框如下所示。

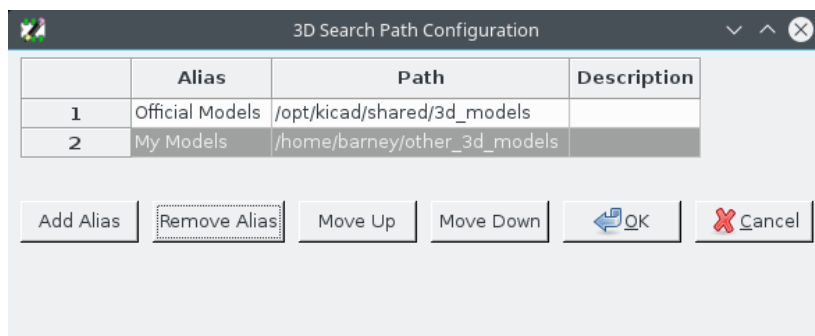


Figure 13.3: KiCad 别名配置对话框

单击 **添加 3D 形状** 可以选择 3D 模型文件以显示如下所示的 3D 模型浏览器。模型浏览器提供 3D 预览，文件过滤器和下拉路径选择器，其中包含通过环境变量或别名定义的当前搜索路径列表。根据型号尺寸和复杂程度，选择模型时可能需要几秒钟才能显示模型。在极端情况下，在测试期间使用的 BGA 封装模型花费大约 12 秒来显示。

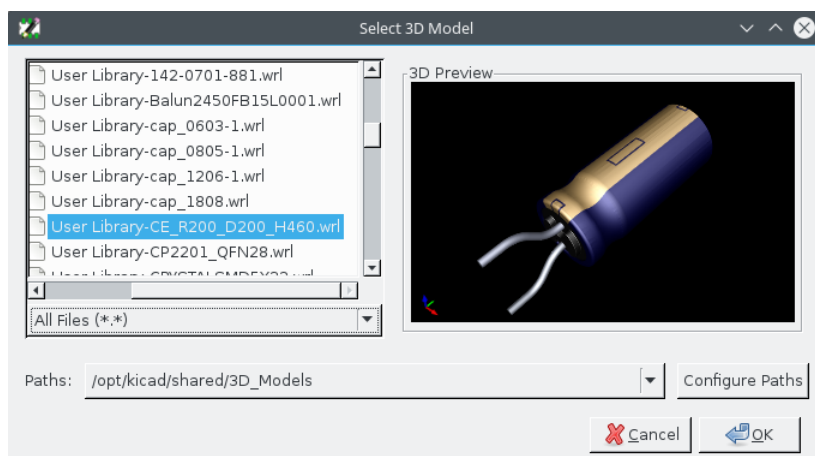


Figure 13.4: KiCad 3D 文件浏览器


13.14 将封装保存到活动库中

保存命令（修改活动库的文件）由图像激活： 按钮。

如果存在相同名称的封装（旧版本），则将覆盖该封装。因为能够对库的封装有信心非常重要，所以在保存之前，有必要仔细检查错误的封装。

在保存之前，还建议将封装的引用或值更改为等于封装的库名称。

13.15 保存封装到电路板

如果编辑的封装来自当前板，则按钮图像： 将在板上更新此封装。


Chapter 14

高级的 PCB 编辑工具

Pcbnew 和封装编辑器中提供了一些更高级的编辑工具，可以帮助您在画布上高效布局元件。

14.1 复制项目

复制是一种克隆项目并在同一操作中拾取它的方法。它大致类似于复制和粘贴，但它允许您在 PCB 上“洒”元件，并允许您使用“移动精确”工具（见下文）更轻松的手动布局元件。

通过使用热键（默认为 Ctrl-D）或上下文菜单中的重复项选项，完成复制图标图像：。

14.2 精确移动项目

“移动精确”工具允许您将项目（或项目组）移动一定量，可以以笛卡尔或极坐标格式输入，并且可以在任何支持的单位中输入。当切换到不同的网格或者根据任何现有网格没有间隔特征时，这将是有益的。

要使用此工具，请选择要移动的项目，然后使用热键（默认为 Ctrl-M）或上下文菜单项来调用对话框。您还可以在移动或复制项目时使用热键调用对话框，这样可以轻松地将偏移重复应用于多个元件。

使用笛卡尔移动矢量条目精确移动

×

相对位置

锚点位置是转换坐标的起点。

☐ 使用极坐标

锚点坐标 X:

0

mm

重置

锚点坐标 Y:

0

mm

重置

旋转项:

0

度

重置

锚点位置 X:

0

Y:

0

选择锚点位置

确定

取消

使用极移动矢量条目精确移动

×

相对位置

锚点位置是转换坐标的起点。

☒ 使用极坐标

距离:

5

mm

重置

角度:

0

度

重置

旋转项:

0

度

重置

锚点位置 X:

0

Y:

0

选择锚点位置

确定

取消

该复选框允许您在笛卡尔坐标系和极坐标系之间切换。表单中当前的任何内容都将自动转换为其他系统。

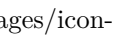
然后输入所需的移动矢量。您可以使用标签指示的单位（上图中的“mm”），也可以自己指定单位（例如，“1 in”表示一英寸，“2 rad”表示 2 弧度）。

按“确定”将转换应用于选择，取消将关闭对话框，不会移动项目。如果按下确定，则下次打开对话框时将保存并预填充移动矢量，这允许将相同的矢量重复应用于多个对象。

14.3 阵列工具

Pcbnew 和封装编辑器都有助于创建特征和元件阵列，可用于在 PCB 和封装上轻松准确地布置重复元素。

14.3.1 激活阵列工具

阵列工具作用于光标下的元件，或者在 GAL 模式下作用于选择。可以通过上下文菜单，图标  进行选择，也可以通过键盘快捷键（默认为 Ctrl-N）访问。

阵列工具显示为对话框窗口，其中包含数组类型的窗格。到目前为止，支持两种类型的数组：网格和圆形。

可以在各个窗格上完全指定每种类型的阵列。几何选项（如何布置网格）在左侧；右侧的编号选项（包括数字在网格中的进展情况）。

14.3.2 网格阵列

网格阵列是根据二维方格网格放置元件的阵列。这种阵列也可以通过仅布置单个行或列来生成线性阵列。

网格阵列的设置对话框如下所示：



The image shows the 'Create Array' (创建阵列) dialog box in Pcbnew. It has two tabs: 'Grid Array' (网格阵列) and 'Circular Array' (圆形阵列). The 'Grid Array' tab is active. The dialog is divided into two main sections: 'Geometry' (几何) on the left and 'Numbering' (编号) on the right.

Geometry Section (Left):

- Horizontal (X) direction count: 5
- Vertical (Y) direction count: 5
- Horizontal spacing: 5 mm
- Vertical spacing: 5 mm
- Horizontal offset: 0 mm
- Vertical offset: 0 mm
- Stagger: 1
- Stagger type: ☒ Column (列), ☐ Row (行)

Numbering Section (Right):

- Wiring board numbering direction: ☒ Horizontal, then vertical (水平, 然后垂直), ☐ Vertical, then horizontal (垂直, 然后水平)
- ☐ Alternate row or column reverse numbering (交替行或列反向编号)
- Initial wiring board number: ☒ Start value (起始值), ☐ Prefer to use unused numbers (优先使用未用编号)
- Wiring board numbering scheme: ☒ Coordinate (A1, A2, ... B1, ...) (坐标), ☐ Continuous (1, 2, 3, ...) (连续)
- Main coordinate axis number: 数值 (0, 1, 2, ..., 9, 10)
- Subordinate coordinate axis number: 数值 (0, 1, 2, ..., 9, 10)
- Wiring board numbering start: 1

Buttons at the bottom: 确定 (OK), 取消 (Cancel).

14.3.2.1 几何选项

几何选项如下：

- **水平计数：**网格中“列”的数量。
- **垂直计数：**网格中的“行”数。
- **水平间距：**从项目到同一行中项目的水平距离和下一栏。如果这是负数，则网格从右向左进行。

- **垂直间距**：从一个项目到项目的垂直距离列和下一行。如果这是负数，则网格进度最低最佳。
- **水平偏移**：从前一行的右边开始每一行
- **垂直偏移**：将每一列的距离从前一列开始

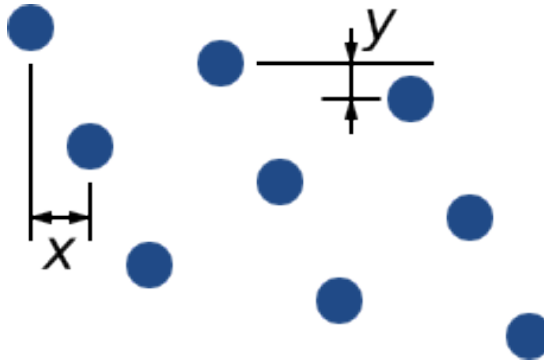


Figure 14.1: 具有 x 和 y 偏移的 3x3 网格

- **交错**：为每行 “n” 行/列添加一个偏移量进展相关间距尺寸的 $1/n$ ：

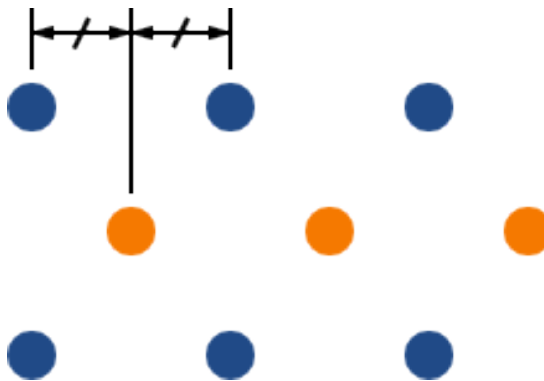


Figure 14.2: 3x3 网格，行交错 2

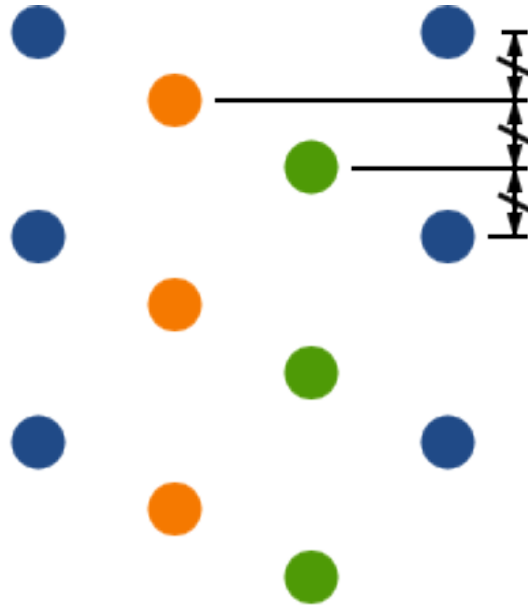


Figure 14.3: 4x3 网格，列错开 3

14.3.2.2 编号选项

- **编号方向**: 确定数字是否沿行继续移动到下一行，或向下移动到下一列，然后移动到下一列。注意编号的方向由间距的符号定义：负数间距将导致从右到左或从下到上的编号。
- **在备用行或列上反向编号**: 如果选择，则为编号顺序（在替代行或列上，从左到右或从右到左）。行或列是否交替取决于编号方向。这个选项对于 DIP 这样的包很有用，其中编号会增加一个一边倒另一边。
- **重新开始编号**: 如果使用已有数字的项目进行布局，重置为开始，否则从该项目的编号继续
- **编号方案**
 - **连续**: 编号只在行/列中断后继续 - 如果第一行中的最后一项编号为“7”，第二行中的第一项行将是“8”。
 - **坐标**: 编号使用双轴方案 number 由行和列索引组成。哪一个是第一位的（行或列）由编号方向确定。
- **轴编号**: 用于对轴进行编号的“字母”。选择是
 - **整数**对于普通整数索引
 - **十六进制**用于基数为 16 索引
 - **字母，减去 IOSQXZ**，电子元件的通用方案，ASME 推荐 Y14.35M-1997 sec. 5.2（以前的 MIL-STD-100 sec. 406.5）避免与数字混淆。
 - **全字母**来自 A-Z。

14.3.3 圆形阵列

圆形阵列围绕圆形或圆弧布置项目。圆由定义的位置（或所选组的中心）和指定的中心点定义。下面是循环阵列配置对话框：



14.3.3.1 几何选项

- **水平中心，垂直中心：**圆的中心。半径调整这些字段时，下面的字段会自动更新。
- **角度：**两个相邻项目之间的角度差异阵列。将此值设置为零可将圆圈均匀分配“技术”元素。
- **计数：**阵列中的项目数（包括原始项）
- **旋转：**围绕自己的位置旋转每个项目。否则，项目将被翻译但不会旋转（例如，方形焊盘如果未设置此选项，将始终保持直立）。

14.3.3.2 编号选项

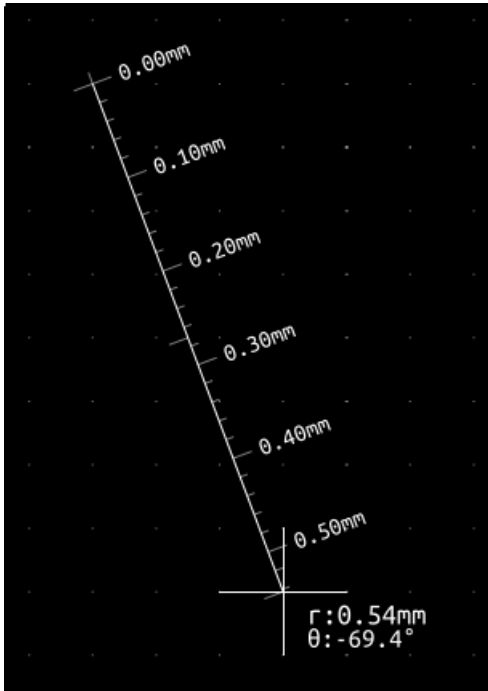
圆形阵列只有一个维度，比网格更简单。可用选项的含义与网格相同。项目顺时针编号 - 对于逆时针阵列，指定负角度。

14.4 测量（标尺）工具

测量工具是一个线性标尺，可用于目视检查 PCB 上的尺寸和间距。

可以通过卡尺图标图像访问它：右侧工具栏中的 `image: images/icons/measurement.png`， “尺寸” 菜单中的热键（默认为 Ctrl-Shift-M）。

激活时，可以在画布上绘制一个临时标尺，该标尺将标有当前单位。您可以通过按住 Ctrl 键捕捉到 45 度角。可以使用常用热键（默认情况下为 Ctrl-U）在不离开工具的情况下更改单位。



Chapter 15

KiCad 脚本参考

脚本允许您使用 **Python** 语言在 KiCad 中自动执行任务。

另请参阅 **Python 脚本参考** 中的 doxygen 文档。

您可以在终端上输入 ‘pydoc pcbnew’ 来查看 Python 模块帮助。

使用脚本可以创建：

- **插件**：KiCad 启动时会加载此类脚本。例子：
 - **封装向导**：帮助您轻松填充参数。请参阅下面的专用部分《Footprint_Wizards, Footprint Wizards》。
 - **文件 I/O** ” (计划)”：允许您编写插件以导出/导入其他文件类型
 - **行动** “(实验性)”：将事件与脚本操作相关联或注册新菜单或工具栏图标。
- **命令行脚本**：可以从命令行，加载板或库中使用的脚本，修改它们，以及渲染输出或新板。

需要注意的是，唯一支持脚本编写的 KiCad 应用程序是 Pcbnew。未来还计划为 Eeschema。

15.1 KiCad 对象

脚本 API 反映了 KiCad/pcbnew 中的内部对象结构。板 (BOARD) 是主要对象，它具有一组属性和一组组件 (MODULEs)，以及布线 (TRACKs) /过孔 (VIAs)，文本 _PCB (TEXTE_PCB)，尺寸 (DIMENSION)，绘制段 (DRAWSEGMENT)。然后元件有 D_ 焊盘 (D_PADs)，边缘 (EDGEs) 等。

- 请参阅下面的板 (BOARD) 部分。

15.2 基本 API 参考

所有 pcbnew API 都是从 Python 中的 “pcbnew” 模块提供的。GetBoard() 方法将在编辑器中返回当前打开的 PCB，对于从 pcbnew 或 action 插件中的集成脚本 shell 编写的命令非常有用。

15.3 加载和保存板 (Board)

- **LoadBoard(文件名):** 使用与文件扩展名匹配的文件格式从文件加载板 (BOARD) 对象的板。
- **SaveBoard(文件名, 板):** 使用与文件扩展名匹配的文件格式将板 (BOARD) 对象保存到文件。
- **board.Save(文件名):** 与上面相同, 但它是板 (BOARD) 对象的一种方法。

加载板, 隐藏所有值的示例显示所有引用

```
#!/usr/bin/env python2.7
import sys
from pcbnew import *

filename=sys.argv[1]

pcb = LoadBoard(filename)
for module in pcb.GetModules():
    print "* Module: %s"%module.GetReference()
    module.Value().SetVisible(False)      # set Value as Hidden
    module.Reference().SetVisible(True)    # set Reference as Visible

pcb.Save("mod_"+filename)
```

15.4 列出和加载库

枚举库, 枚举模块, 枚举焊盘

```
#!/usr/bin/python

from pcbnew import *

libpath = "/usr/share/kicad/modules/Socket.pretty"
print ">> enumerate footprints, pads of",libpath

# Load the suitable plugin to read/write the .pretty library
# (containing the .kicad_mod footprint files)
src_type = IO_MGR.GuessPluginTypeFromLibPath( libpath );
# Rem: we can force the plugin type by using IO_MGR.PluginFind( IO_MGR.KICAD )
plugin = IO_MGR.PluginFind( src_type )

# Print plugin type name: (Expecting "KiCad" for a .pretty library)
print( "Selected plugin type: %s" % plugin.PluginName() )

list_of_footprints = plugin.FootprintEnumerate(libpath)

for name in list_of_footprints:
```

```

fp = plugin.FootprintLoad(libpath,name)
# print the short name of the footprint
print name # this is the name inside the loaded library
# followed by ref field, value field, and decription string:
# Remember ref and value texts are dummy texts, replaced by the schematic values
# when reading a netlist.
print "  ->", fp.GetReference(), fp.GetValue(), fp.GetDescription()

# print pad info: GetPos0() is the pad position relative to the footprint position
for pad in fp.Pads():
    print "    pad [%s]" % pad.GetPadName(), "at", \
        "pos0", ToMM(pad.GetPos0().x), ToMM(pad.GetPos0().y), "mm", \
        "shape offset", ToMM(pad.GetOffset().x), ToMM(pad.GetOffset().y), "mm"
print ""

```

15.5 板 (BOARD)

板 (Board) 是 KiCad pcbnew 中的基本对象，它是文档。

板 (BOARD) 包含一组可以使用以下方法访问的对象列表，它们将返回可以使用 “for obj in list:” 迭代的可迭代列表：

- **board.GetModules ()**：此方法返回组件 (MODULE) 对象列表，板中可用的所有模块都将在此处公开。
- **board.GetDrawings ()**：返回属于电路板图纸的板项目 (BOARD_ITEMS) 列表
- **board.GetTracks()**：此方法返回板 (BOARD) 内的布线 (TRACK) 和过孔 (VIA) 列表
- **board.GetFullRatsnest()**：返回飞线 (ratsnest) 列表 (连接仍未路由)
- **board.GetNetClasses()**：返回网络类列表
- **board.GetCurrentNetClassName()**：返回当前的网络类
- **board.GetViasDimensionsList()**：返回电路板可用的过孔 (Via) 尺寸列表。
- **board.GetTrackWidthList()**：返回电路板可用的布线宽度 (Track Widths) 列表。

板 (Board) 检查示例

```

#!/usr/bin/env python
import sys
from pcbnew import *

filename=sys.argv[1]

pcb = LoadBoard(filename)

ToUnits = ToMM

```

```
FromUnits = FromMM
#ToUnits=ToMils
#FromUnits=FromMils

print "LISTING VIAS:"

for item in pcb.GetTracks():
    if type(item) is VIA:

        pos = item.GetPosition()
        drill = item.GetDrillValue()
        width = item.GetWidth()
        print " * Via:    %s - %f/%f"%(ToUnits(pos),ToUnits(drill),ToUnits(width))

    elif type(item) is TRACK:

        start = item.GetStart()
        end = item.GetEnd()
        width = item.GetWidth()

        print " * Track: %s to %s, width %f" % (ToUnits(start),ToUnits(end),ToUnits(width))

    else:
        print "Unknown type    %s" % type(item)

print ""
print "LIST DRAWINGS:"

for item in pcb.GetDrawings():
    if type(item) is TEXTE_PCB:
        print "* Text:    '%s' at %s"%(item.GetText(), item.GetPosition())
    elif type(item) is DRAWSEGMENT:
        print "* Drawing: %s"%item.GetShapeStr() # dir(item)
    else:
        print type(item)

print ""
print "LIST MODULES:"

for module in pcb.GetModules():
    print "* Module: %s at %s"%(module.GetReference(),ToUnits(module.GetPosition()))

print ""
print "Ratsnest cnt:",len(pcb.GetFullRatsnest())
print "track w cnt:",len(pcb.GetTrackWidthList())
print "via s cnt:",len(pcb.GetViasDimensionsList())

print ""
```



```
print "LIST ZONES:", pcb.GetAreaCount()

for idx in range(0, pcb.GetAreaCount()):
    zone=pcb.GetArea(idx)
    print "zone:", idx, "priority:", zone.GetPriority(), "netname", zone.GetNetname()

print ""
print "NetClasses:", pcb.GetNetClasses().GetCount(),
```

15.6 例子

15.6.1 更改元件引脚的焊膏层边距

我们只想将引脚从 1 改为 14,15 是一个必须保持原样的导热焊盘。

```
#!/usr/bin/env python2.7
import sys
from pcbnew import *

filename=sys.argv[1]
pcb = LoadBoard(filename)

# Find module U304
u304 = pcb.FindModuleByReference('U304')
pads = u304.Pads()

# Iterate over pads, printing solder paste margin
for p in pads:
    print p.GetPadName(), ToMM(p.GetLocalSolderPasteMargin())
    id = int(p.GetPadName())
    # Set margin to 0 for all but pad (pin) 15
    if id<15: p.SetLocalSolderPasteMargin(0)

pcb.Save("mod_"+filename)
```

15.7 封装向导

封装向导是可以从封装编辑器访问的 Python 脚本的集合。如果调用封装对话框，则选择一个给定的向导，该向导允许您查看渲染的封装，并且您可以编辑一些参数。

如果插件未正确分发到您的系统软件包，您可以在 KiCad 源代码树中的链接中找到最新版本：[gitlab](#)。

它们应位于例如 “C:\Program Files\KiCad\share\kicad\scripting\plugins” 中。

在 linux 上，您还可以将用户插件保存在 “\$HOME/.kicad_plugins” 中。

构建封装，轻松填写参数。

```

from __future__ import division
import pcbnew

import HelpfulFootprintWizardPlugin as HFPW

class FPC_FootprintWizard(HFPW.HelpfulFootprintWizardPlugin):

    def GetName(self):
        return "FPC (SMT connector)"

    def GetDescription(self):
        return "FPC (SMT connector) Footprint Wizard"

    def GetValue(self):
        pins = self.parameters["Pads"]["*n"]
        return "FPC_%d" % pins

    def GenerateParameterList(self):
        self.AddParam( "Pads", "n", self.uNatural, 40 )
        self.AddParam( "Pads", "pitch", self.uMM, 0.5 )
        self.AddParam( "Pads", "width", self.uMM, 0.25 )
        self.AddParam( "Pads", "height", self.uMM, 1.6)
        self.AddParam( "Shield", "shield_to_pad", self.uMM, 1.6 )
        self.AddParam( "Shield", "from_top", self.uMM, 1.3 )
        self.AddParam( "Shield", "width", self.uMM, 1.5 )
        self.AddParam( "Shield", "height", self.uMM, 2 )

    # build a rectangular pad
    def smdRectPad(self,module,size,pos,name):
        pad = pcbnew.D_PAD(module)
        pad.SetSize(size)
        pad.SetShape(pcbnew.PAD_SHAPE_RECT)
        pad.SetAttribute(pcbnew.PAD_ATTRIB_SMD)
        pad.SetLayerSet( pad.SMDMask() )
        pad.SetPos0(pos)
        pad.SetPosition(pos)
        pad.SetPadName(name)
        return pad

    def CheckParameters(self):
        p = self.parameters
        self.CheckParamInt( "Pads", "*n" ) # not internal units preceded by "*"

    def BuildThisFootprint(self):

```

```

p = self.parameters
pad_count      = int(p["Pads"]["*n"])
pad_width      = p["Pads"]["width"]
pad_height     = p["Pads"]["height"]
pad_pitch      = p["Pads"]["pitch"]
shl_width      = p["Shield"]["width"]
shl_height     = p["Shield"]["height"]
shl_to_pad     = p["Shield"]["shield_to_pad"]
shl_from_top   = p["Shield"]["from_top"]

offsetX        = pad_pitch * ( pad_count-1 ) / 2
size_pad = pcbnew.wxSize( pad_width, pad_height )
size_shld = pcbnew.wxSize(shl_width, shl_height)
size_text = self.GetTextSize()  # IPC nominal

# Gives a position and size to ref and value texts:
textposy = pad_height/2 + pcbnew.FromMM(1) + self.GetTextThickness()
self.draw.Reference( 0, textposy, size_text )

textposy = textposy + size_text + self.GetTextThickness()
self.draw.Value( 0, textposy, size_text )

# create a pad array and add it to the module
for n in range ( 0, pad_count ):
    xpos = pad_pitch*n - offsetX
    pad = self.smdRectPad(self.module,size_pad, pcbnew.wxPoint(xpos,0),str(n+1))
    self.module.Add(pad)

# Mechanical shield pads: left pad and right pad
xpos = -shl_to_pad-offsetX
pad_s0_pos = pcbnew.wxPoint(xpos,shl_from_top)
pad_s0 = self.smdRectPad(self.module, size_shld, pad_s0_pos, "0")
xpos = (pad_count-1) * pad_pitch+shl_to_pad - offsetX
pad_s1_pos = pcbnew.wxPoint(xpos,shl_from_top)
pad_s1 = self.smdRectPad(self.module, size_shld, pad_s1_pos, "0")

self.module.Add(pad_s0)
self.module.Add(pad_s1)

# add footprint outline
linewidth = self.draw.GetLineThickness()
margin = linewidth

# upper line
posy = -pad_height/2 - linewidth/2 - margin
xstart = - pad_pitch*0.5-offsetX
xend = pad_pitch * pad_count + xstart;

```

```
self.draw.Line( xstart, posy, xend, posy )

# lower line
posy = pad_height/2 + linewidth/2 + margin
self.draw.Line(xstart, posy, xend, posy)

# around left mechanical pad (the outline around right pad is mirrored/y axis)
yend = pad_s0_pos.y + shl_height/2 + margin
self.draw.Line(xstart, posy, xstart, yend)
self.draw.Line(-xstart, posy, -xstart, yend)

posy = yend
xend = pad_s0_pos.x - (shl_width/2 + linewidth + margin*2)
self.draw.Line(xstart, posy, xend, posy)

# right pad side
self.draw.Line(-xstart, posy, -xend, yend)

# set SMD attribute
self.module.SetAttributes(pcbnew.MOD_CMS)

# vertical segment at left of the pad
xstart = xend
yend = posy - (shl_height + linewidth + margin*2)
self.draw.Line(xstart, posy, xend, yend)

# right pad side
self.draw.Line(-xstart, posy, -xend, yend)

# horizontal segment above the pad
xstart = xend
xend = - pad_pitch*0.5-offsetX
posy = yend
self.draw.Line(xstart, posy, xend, yend)

# right pad side
self.draw.Line(-xstart, posy, -xend, yend)

# vertical segment above the pad
xstart = xend
yend = -pad_height/2 - linewidth/2 - margin
self.draw.Line(xstart, posy, xend, yend)

# right pad side
self.draw.Line(-xstart, posy, -xend, yend)

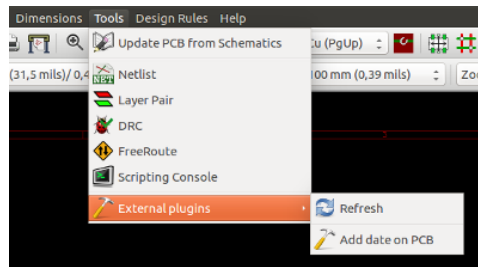
# register into pcbnew
```

```
FPC_FootprintWizard().register()
```

15.8 动作插件

动作插件将事件关联到脚本动作。目前只注册一个新菜单。

菜单内有新菜单 **工具 外部插件**。



- **刷新**：重新加载插件（如果需要，创建新菜单）
- **在 PCB 上添加日期**：示例插件。

警告：与所有其他 python 脚本一样，undo/redo 功能不起作用（还是!）。

动作插件示例：向内容为 “\$date\$” 的任何文本项添加日期

```
import pcbnew
import re
import datetime

class text_by_date(pcbnew.ActionPlugin):
    """
    test_by_date: A sample plugin as an example of ActionPlugin
    Add the date to any text field of the board where the content is '$date$'
    How to use:
    - Add a text on your board with the content '$date$'
    - Call the plugin
    - Automatically the date will be added to the text (format YYYY-MM-DD)
    """

    def defaults(self):
        """
        Method defaults must be redefined
        self.name should be the menu label to use
        self.category should be the category (not yet used)
        self.description should be a comprehensive description
        of the plugin
        """
        self.name = "Add date on PCB"
        self.category = "Modify PCB"
```

```
self.description = "Automatically add date on an existing PCB"

def Run(self):
    pcb = pcbnew.GetBoard()
    for draw in pcb.GetDrawings():
        if draw.GetClass() == 'PTEXT':
            txt = re.sub("\$date\$ [0-9]{4}-[0-9]{2}-[0-9]{2}",
                          "$date$", draw.GetText())
            if txt == "$date$":
                draw.SetText("$date$ %s"%datetime.date.today())

text_by_date().register()
```