Eeschema 简介

December 22, 2020

Contents

参考手册

Copyright

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 翻译: https://gitlab.com/kicad/code/kicad-i18n/issues

出版日期和软件版本

发布于 2015 年 5 月 30 日。

Chapter 1

Eeschema 简介

1.1 描述

Eeschema 是一个原理图设计软件，作为 KiCad 的一部分分发，可在以下操作系统下使用：

- Linux
- Apple OS X
- Windows

无论操作系统如何，所有 Eeschema 文件都可以从一个操作系统 100％兼容到另一个操作系统。

Eeschema 是一个集成的应用程序，其中绘图，控制，布局，库管理和访问 PCB 设计软件的所有功能都在 Eeschema 本身内执行。

Eeschema 打算与 PcbNew 合作，后者是 KiCad 的印刷电路设计软件。它还可以导出网表文件，其中列出了其他软件包的所有电气连接。

Eeschema 包含一个符号库编辑器，可以创建和编辑符号并管理库。它还集成了现代原理图捕获软件所需的以下附加但必不可少的功能：

- 电气规则检查（ERC），用于自动控制错误和缺失的连接
- 以多种格式导出绘图文件（Postscript，PDF，HPGL 和 SVG）
- 物料清单生成（通过 Python 或 XSLT 脚本，允许许多灵活的格式）。

1.2 技术概述

Eeschema 仅受可用内存的限制。因此，对元件、元件引脚，连接或板的数量没有实际限制。在多张图表的情况下，表示是分层的。

Eeschema 可以通过以下几种方式使用多表格图表：

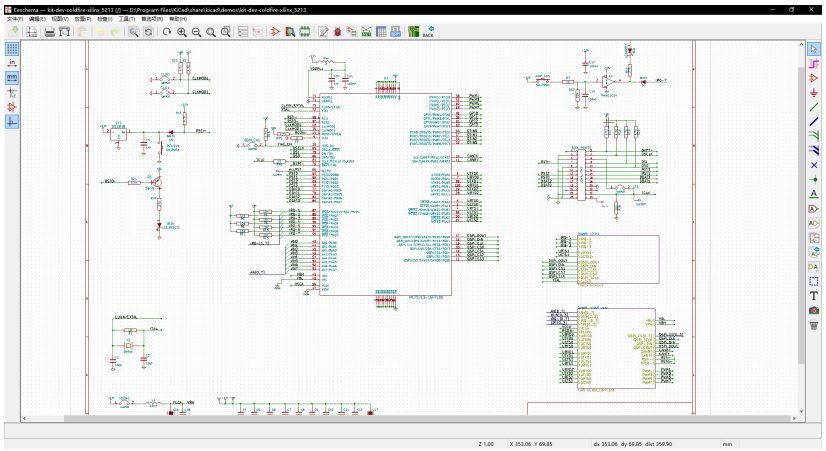
- 简单的层次结构（每个原理图只使用一次）。
- 复杂的层次结构（一些原理图在多个实例中不止一次使用）。
- 扁平层次结构（原理图未在主图中明确连接）。

Chapter 2

通用 Eeschema 命令

命令可以通过以下方式执行:

- 单击菜单栏（屏幕顶部)。
- 单击屏幕顶部的图标 (常规命令)。
- 点击屏幕右侧的图标 (特定命令或“工具”)。
- 单击屏幕左侧的图标 (显示选项)。
- 按下鼠标按钮（重要的补充命令)。特别是右键单击打开光标下元素的上下文菜单（缩放，网格和元素编辑)。
- 功能键（F1，F2，F3，F4，Insert 和 Space 键)。具体来说：Esc 键取消正在进行的命令。Insert 键允许复制最后创建的元素。
- 按热键，通常执行选择工具命令并在当前光标位置开始工具操作。有关热键列表，请参阅“帮助 → 列出热键”菜单项或按“Ctrl+F1”键。



2.1 鼠标命令

2.1.1 基本命令

左键

- 单击：在状态栏中显示光标下的符号或文本的特征。
- 双击：编辑（如果元素可编辑）符号或文本。

右键

- 打开弹出菜单。

2.1.2 阻止操作

您可以在所有 Eeschema 菜单中移动，拖动，复制和删除所选区域。

通过使用鼠标左键在项目周围绘制一个框来选择区域。

在选择期间按住“Shift”，“Ctrl”或“Shift + Ctrl”分别执行复制，拖动和删除：

鼠标左键	移动选择。
Shift + 鼠标左键	复制选择。
Ctrl + 鼠标左键	拖动选择。
Ctrl + Shift + 鼠标左键	删除选择。

拖动或复制时，您可以：

- 再次单击以放置元素。
- 单击右键或按 Esc 键取消。

如果已启动块移动命令，则可以使用右键单击弹出菜单选择另一个命令。

2.2 热键

- “Ctrl+F1”键显示当前热键列表。
- 可以在“原理图编辑器选项”对话框的“控件”选项卡中重新定义热键（菜单“首选项”→ “常规选项”）

这是默认的热键列表：

帮助（此窗口）	Ctrl+F1
放大	F1
缩小	F2
缩放重绘	F3
缩放中心	F4
适合屏幕	Home
缩放到选择	@
重置本地坐标	Space
编辑项目	E
删除项目	Del
旋转项目	R
拖动项目	G
撤消	Ctrl+Z
重做	Ctrl+Y
鼠标左键单击	Return
鼠标左键双击	End

保存原理图	Ctrl+S
加载原理图	Ctrl+O
查找项目	Ctrl+F
查找下一个项目	F5
查找下一个 DRC 标记	Shift+F5
查找和替换	Ctrl+Alt+F
重复最后一项	Ins
移动块 → 拖动块	Tab
复制块	Ctrl+C
粘贴块	Ctrl+V
切块	Ctrl+X
移动原理图项目	M
重复的符号或标签	C
添加符号	A
添加电源	P
镜像 X	X
镜像 Y	Y
定向普通符号	N
编辑符号值	V
编辑符号参考	U
编辑符号封装	F
使用符号编辑器编辑	Ctrl+E
开始画线	W
开始总线	B
终端线路总线	K
添加标签	L
添加分层标签	H
添加全局标签	Ctrl+L
添加连接点	J
添加无连接标志	Q
添加表	S
添加电线入口	Z
添加总线入口	/
添加图形折线	I
添加图形文字	T
从原理图更新到 PCB	F8
自置域	O
留下表	Alt+BkSp
删除节点	BkSp
突出显示连接	Ctrl+X

可以使用热键编辑器重新定义所有热键（菜单首选项 → 常规选项 → 《首选项-控件，控件》）。

可以使用菜单 (首选项 → 导入和导出 → 导入/导出热键) 导入/导出热键设置。

2.3 格点

在 Eeschema 中，光标始终在网格上移动。网格可以自定义：

- 可以使用弹出菜单或使用“首选项/选项”菜单更改大小。
- 可以在“原理图编辑器选项”对话框的“颜色”选项卡中更改颜色（菜单首选项 → 常规选项）。
- 可以使用左侧工具栏按钮切换可见性。

默认网格尺寸为 50mil（0.050”）或 1.27mm。

这是在符号编辑器中设计符号时将符号和电线放置在原理图中以及放置引脚的首选网格。

人们还可以使用 25mil 到 10mil 的较小网格。这仅用于设计符号体或放置文本和注释，不建议用于放置引脚和电线。

2.4 缩放选择

要更改缩放级别:

- 右键单击以打开弹出菜单，然后选择所需的缩放。
- 或使用功能键:
  - F1：放大
  - F2：缩小
  - F4 或只需单击鼠标中键（不移动鼠标）：围绕光标指针位置居中
- 窗口缩放:
  - 鼠标滚轮：放大/缩小
  - Shift + 鼠标滚轮：向上/向下平移
  - Ctrl + 鼠标滚轮：向左/向右平移

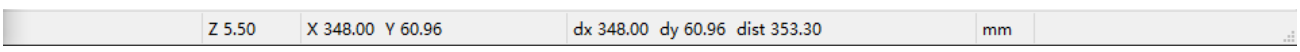
2.5 显示光标坐标

显示单位为英寸或毫米。但是，Eeschema 总是使用 0.001 英寸（mil/thou）作为其内部单元。

窗口右下角显示以下信息：

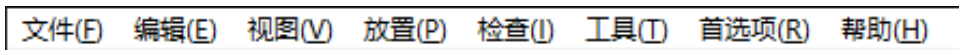
- 缩放系数
- 光标的绝对位置
- 光标的相对位置

按空格可以将相对坐标重置为零。这对于测量两点之间的距离或对齐对象很有用。



2.6 顶级菜单栏

顶部菜单栏允许打开和保存原理图，程序配置和查看文档。



2.7 上方工具栏

此工具栏可以访问 Eeschema 的主要功能。

如果 Eeschema 以独立模式运行，则这是可用的工具集：

请注意，当 KiCad 在项目模式下运行时，前两个图标不可用，因为它们可以处理单个文件。

	创建新原理图（仅在独立模式下）。
	打开原理图（仅在独立模式下）。
	保存完整的原理图项目。
	选择图纸尺寸并编辑标题栏。
	打开打印对话框。
	将复制/剪切的项目或块粘贴到当前工作表。
	撤消：还原最后一次更改。
	重做：还原最后一次撤消操作。
	显示在原理图中搜索符号和文本的对话框。
	显示用于搜索和替换原理图中文本的对话框。
	刷新屏幕; 缩放以适应。
	放大和缩小。
	查看和导航层次结构树。
	保留当前工作表并进入层次结构中。
	调用符号库编辑器以查看和修改库和符号。
	浏览符号库。
	注释符号。

	电气规则检查器（ERC），自动验证电气连接。
	调用 CvPcb 为符号分配封装。
	导出网表（Pcbnew，SPICE 和其他格式）。
	编辑符号字段。
	生成物料清单（BOM）。
	调用 Pcbnew 执行 PCB 布局。
	返回导入封装分配（使用 CvPcb 或 Pcbnew 选择）到“封装”字段中。

2.8 右侧工具栏图标

此工具栏包含以下工具：

- 放置符号，电线，总线，交叉点，标签，文本等。
- 创建分层子表和连接符号。

	取消激活的命令或工具。
	通过使用不同颜色标记其电线和网络标签来突出显示网络。如果 KiCad 以工程模式运行，那么对应于所选网络的铜将在 Pcbnew 中也会高亮。
	显示符号选择器对话框以选择要放置的新符号。
	显示电源符号选择器对话框以选择要放置的电源符号。
	画一根电线。
	画一根总线
	绘制线对总入口点。这些元素仅是图形化的，不会创建连接，因此它们不应用于将电线连接在一起。
	绘制总线到总线的入口点。
	放置“无连接”标志。这些标志应放在符号引脚上意味着没有连接。这样做是为了通知电气规则检查器特定引脚缺少连接是故意的，应该不报告。
	放置一个交叉点。这连接两根交叉线或一根线和一个引脚，当它可能是模糊的（即，如果线端或引脚不是直接的连接到另一个线端）。
	放置一个本地标签。本地标签连接 **位于同一张纸**中的物品。对于两个不同工作表之间的连接，您必须使用全局或分层标签。
	放置一个全局标签。即使具有相同名称的所有全局标签也已连接位于不同的纸张上。
	放置分层标签。分层标签用于创建子表与包含它的父表之间的连接。
	放置分层子表。您必须指定此子表的文件名。
	从子表单导入分层引脚。该命令只能在执行时执行分层子表。它将创建对应于分层的分层引脚放置在目标子表中的标签。
	在子表中放置分层引脚。该命令只能在执行时执行分层子表。它会创建任意分层引脚，即使它们也是如此目标子表中不存在。
	划一条线。这些只是图形化的，不能连接任何东西。
	放置文本注释。

	放置位图图像。
	删除所选元素。

2.9 左工具栏图标

此工具栏管理显示选项：

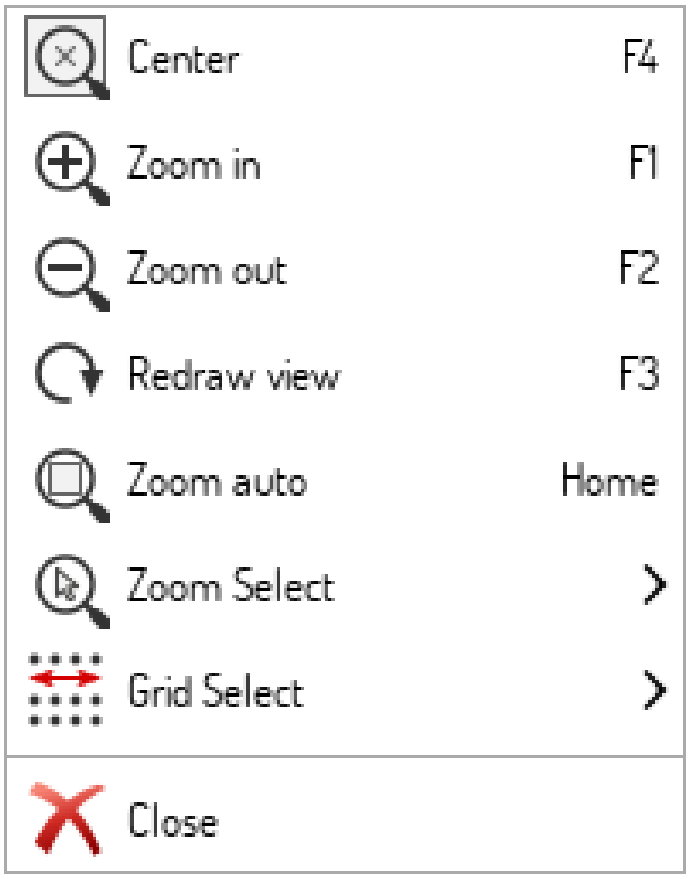
	Toggle grid visibility.
	Switch units to inches.
	Switch units to millimeters.
	Choose the cursor shape (full screen/small).
	Toggle visibility of ”invisible” pins.
	Toggle free angle/90 degrees wires and buses placement.

2.10 弹出菜单和快速编辑

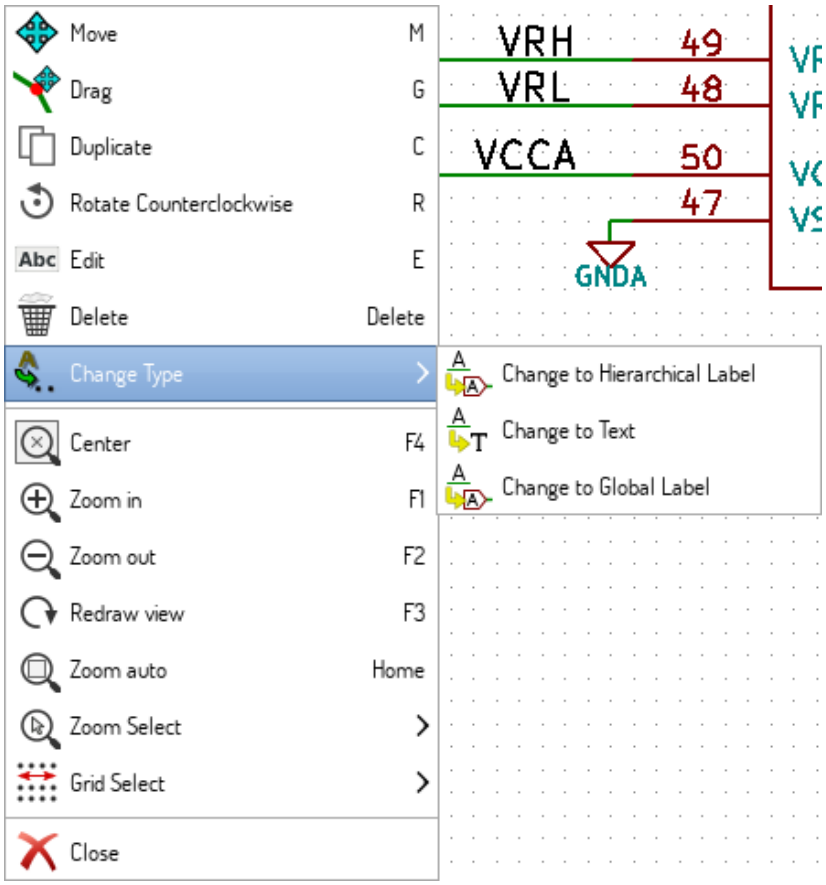
右键单击可打开所选元素的上下文菜单。这包含：

- 缩放系数。
- 网格调整。
- 通常编辑所选元素的参数。

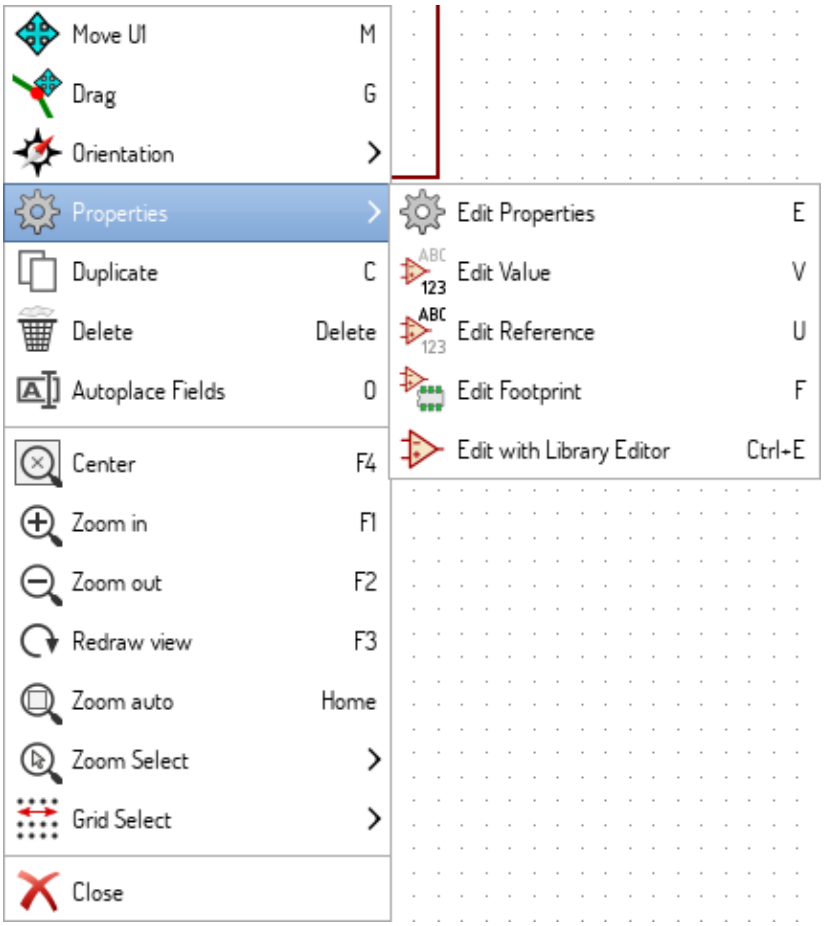
没有选定元素的弹出窗口。


Center F4
Zoom in F1
Zoom out F2
Redraw view F3
Zoom auto Home
Zoom Select
Grid Select
Close

编辑标签。


Move M
Drag G
Duplicate C
Rotate Counterclockwise R
Edit E
Delete Delete
Change Type
Center F4
Zoom in F1
Zoom out F2
Redraw view F3
Zoom auto Home
Zoom Select
Grid Select
Close
Change to Hierarchical Label
Change to Text
Change to Global Label
VRH 49
VRL 48
VCCA 50
47
GNDA

编辑符号。


Move UI M
Drag G
Orientation
Properties
Duplicate C
Delete Delete
Autoplace Fields 0
Center F4
Zoom in F1
Zoom out F2
Redraw view F3
Zoom auto Home
Zoom Select
Grid Select
Close
Edit Properties E
Edit Value V
Edit Reference U
Edit Footprint F
Edit with Library Editor Ctrl+E

Chapter 3

主菜单

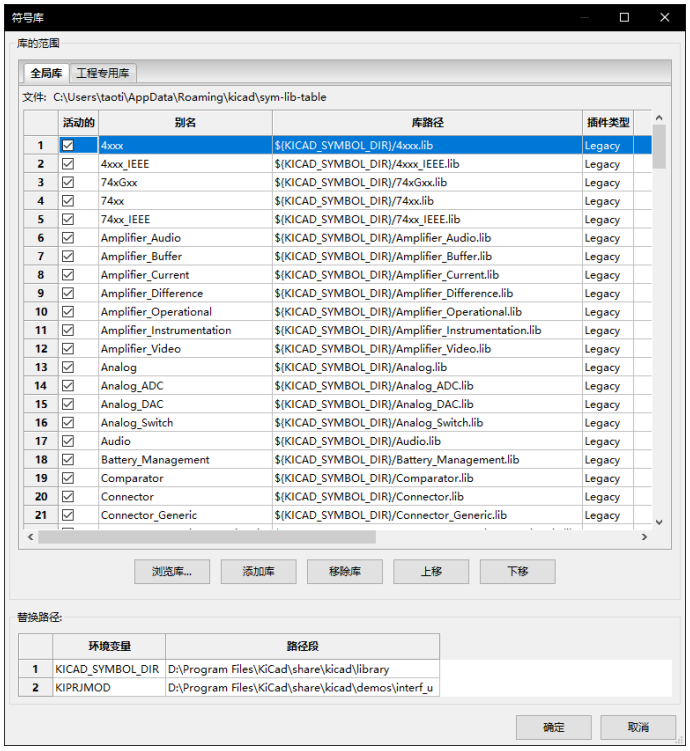
3.1 文件菜单

新建	关闭当前原理图并启动一个新原理图（仅在独立模式下）。
打开	加载原理图项目（仅在独立模式下）。
打开最近	从最近打开的文件列表中打开原理图项目（仅在独立模式下）。
附加示意图	将另一个工作表的内容插入当前工作表。
导入非 Kicad 原理图文件	导入以其他文件格式保存的原理图项目。
保存	保存当前工作表及其所有子工作表。
保存当前工作表	仅保存当前工作表，而不保存项目中的其他工作表。
将当前工作表另存为…	以新名称保存当前工作表。
页面设置	配置页面尺寸和标题栏。
打印	打印原理图项目（另见“打印和打印，打印和打印”一章）。
绘图	导出为 PDF，PostScript，HPGL 或 SVG 格式（请参见“绘图和打印，绘图和打印”一章）。
关闭	终止应用程序。

3.2 首选项菜单

管理符号库表	添加/删除符号库。
配置路径	设置默认搜索路径。
常规选项	首选项（单位，网格大小，字段名称等）。
设置语言	选择界面语言。
图标选项	图标可见性设置。
导入和导出	将首选项传输到/从文件传输。

3.2.1 管理符号库表



Eeschema 使用两个库表来存储可用符号库列表，这些符号库因范围而异：

- 全局库

每个项目都可以使用全局库表中列出的库。它们保存在您的主目录中的 **sym-lib-table** 中（确切路径取决于操作系统; 请检查表上方的路径）。

- 项目专用库

项目专用库中列出的库可用于当前打开的项目。它们保存在项目目录中的 **sym-lib-table** 文件中（检查表格上方的路径）。

您可以通过单击库表下方的 *全局库* 或 *项目专用库* 选项卡来查看任一列表。

3.2.1.1 添加一个新库

通过单击 **浏览库…** 按钮并选择文件或单击“附加库”并键入库文件的路径来添加库。选定的库将添加到当前打开的库表（全局/项目专用）中。

3.2.1.2 删除库

通过选择一个或多个库并单击 **删除库** 按钮来删除库。

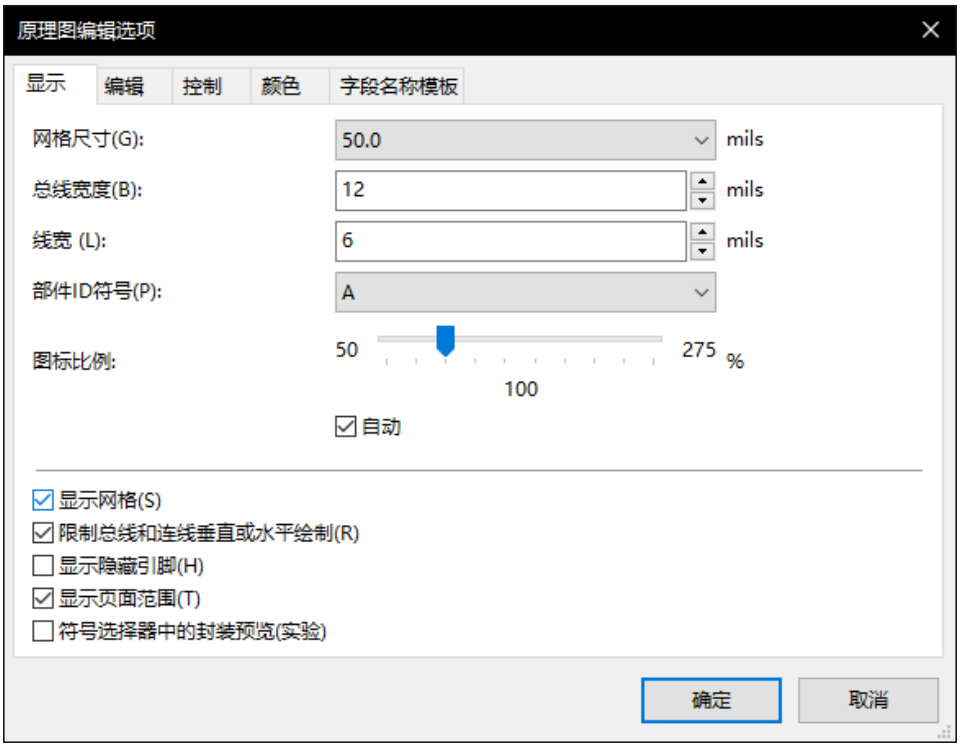
3.2.1.3 库属性

表中的每一行都存储了几个描述库的字段：

活动	启用/禁用库。暂时减少加载的库集很有用。
昵称	昵称是用于将符号分配给元件的简短唯一标识符。符号由'<Library Nickname>：<Symbol Name>' 字符串表示。
库路径	路径指向库位置。
插件类型	确定库文件格式。
选项	存储库特定选项（如果插件使用）。
说明	简要描述库内容。

3.2.2 常规选项

3.2.2.1 显示



网格尺寸	网格大小选择。 **建议**使用普通网格（0.050 英寸或 1,27 毫米）。较小网格用于元件构建。
总线厚度	用于绘制总线的笔大小。
线条粗细	用于绘制没有对象的对象的笔大小指定的笔大小。
元件 ID 表示法	用于表示符号单元的后缀样式（U1A，U1.A，U1-1 等）
图标比例	调整工具栏图标大小。
显示网格	网格可见性设置。
将总线和电线限制为 H 和 V 方向	如果检查，总线和电线仅用垂直或水平线绘制。否则，可以在任何方向放置总线和电线。
显示隐藏的引脚：	通常显示不可见（或 隐藏）引脚电源引脚。
显示页面限制	如果选中，则在屏幕上显示页面边界。

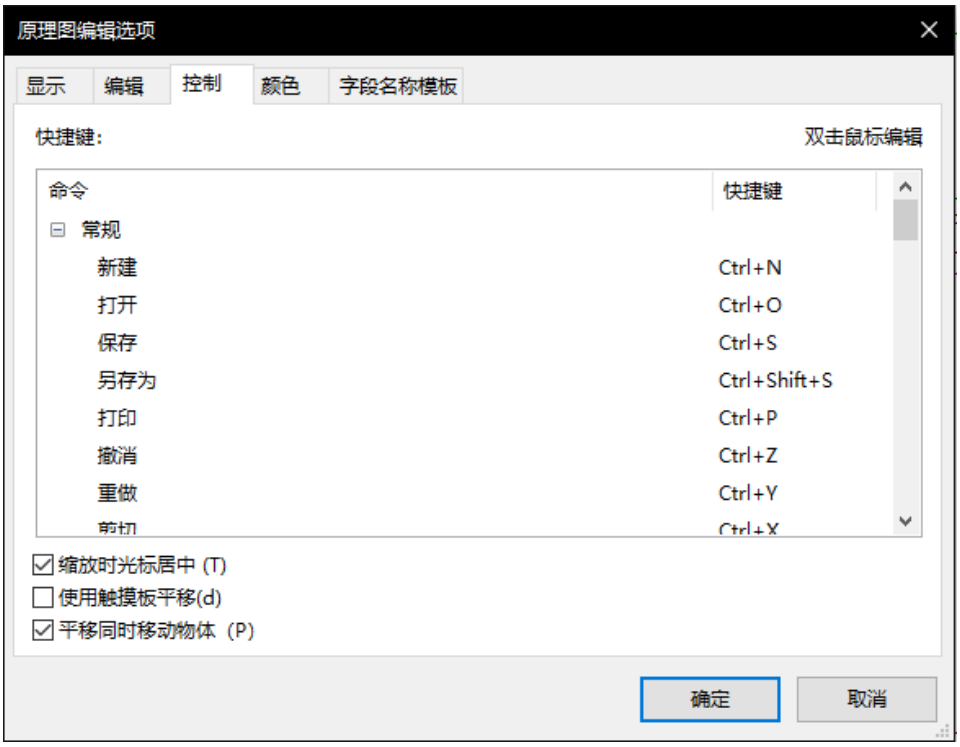
符号选择器中的封装预览	显示封装预览框和放置新符号时的封装选择器。 **注意：**可能会导致问题或延迟，使用风险自负。

3.2.2.2 编辑

测量单位	选择显示和光标坐标单位（英寸或毫米）。
重复项目的水平间距	元素复制期间 X 轴上的增量（默认值：0）（在放置符号，标签或电线等物品后，*Insert* 键复制）
重复项目的垂直间距	在 Y 轴期间增量元素复制（默认值：0.100 英寸或 2,54 毫米）。
重复标签的增量	在复制文本结束期间标签值的增量在一个数字中，例如总线成员（通常值 1 或-1）。
默认文本大小	创建新文本项或标签时使用的文本大小。
自动保存时间间隔	保存备份之间的时间（分钟）。
自动放置符号字段	如果选中，则为符号字段（例如，值和可能会移动新放置的符号以避免与其发生冲突其他项目。
允许字段自动放置更改对齐	扩展“自动放置符号字段”的选项。放置时启用符号字段的文本对齐调整一个新的部分。
始终将自动播放的字段与 50mil 网格对齐	自动扩展放置符号字段的选项。如果选中，则使用 50mil 自动放置字段网格，否则它们是自由放置的。

3.2.2.3 控制

重新定义热键并设置用户界面行为。



通过双击操作选择新的热键，或右键单击操作以显示弹出菜单：

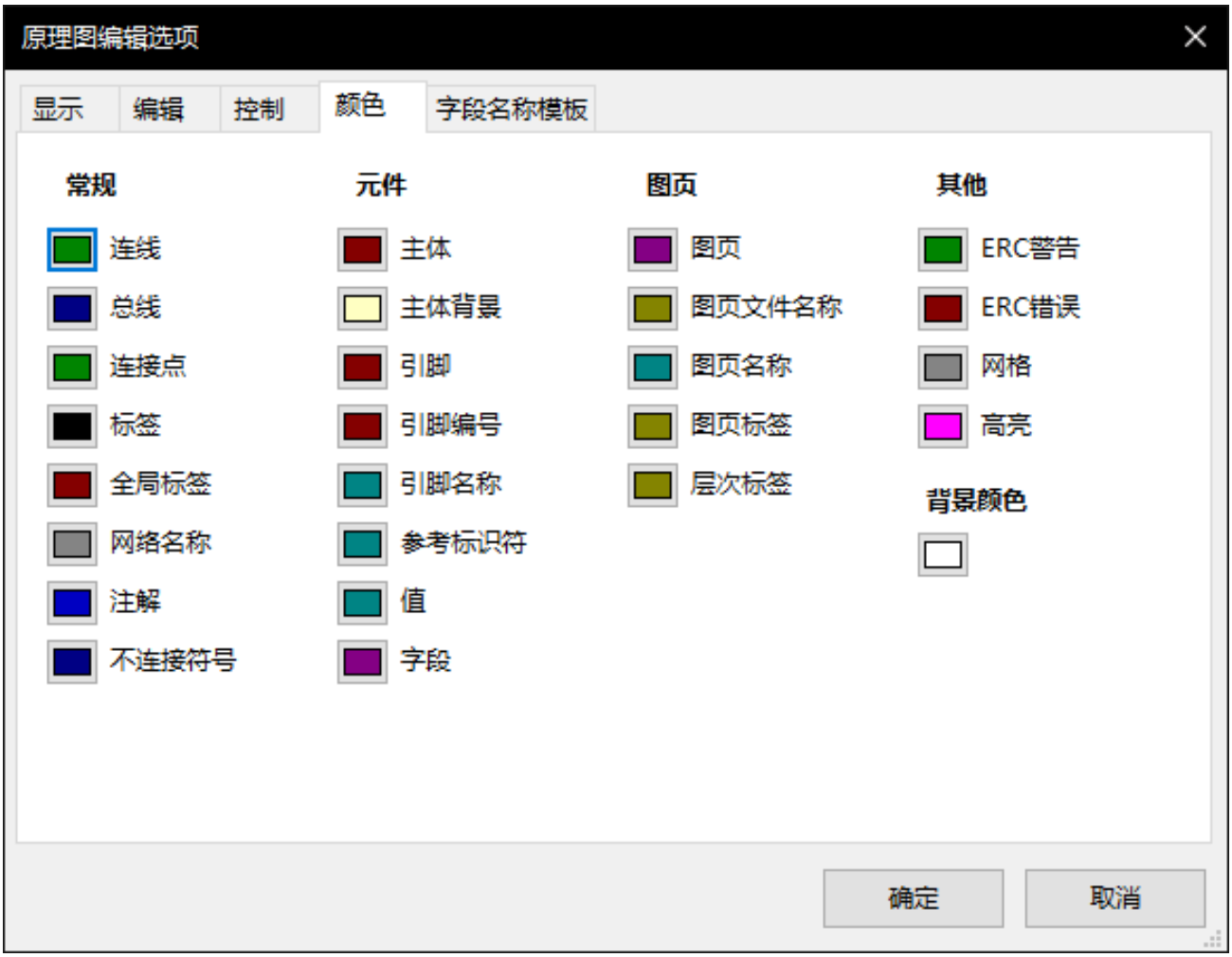
编辑	为操作定义新的热键（与双击相同）。
撤消更改	还原操作的最近热键更改。
恢复默认值	将操作热键设置为其默认值。
撤消所有更改	还原操作的所有最近热键更改。
全部恢复为默认值	将所有操作热键设置为其默认值。

选项说明:

中心和变形光标变焦	如果选中，则指向的位置会变形放大/缩小时到屏幕中心。
使用触摸板进行平移	启用时，使用滚轮（或。）平移视图触摸板手势）和缩放一个需要按住 Ctrl 键。否则滚轮放大/缩小和 Ctrl/Shift 是平移修改器。
移动对象时平移	如果选中，则自动平移窗口如果光标在绘图或移动过程中离开窗口。

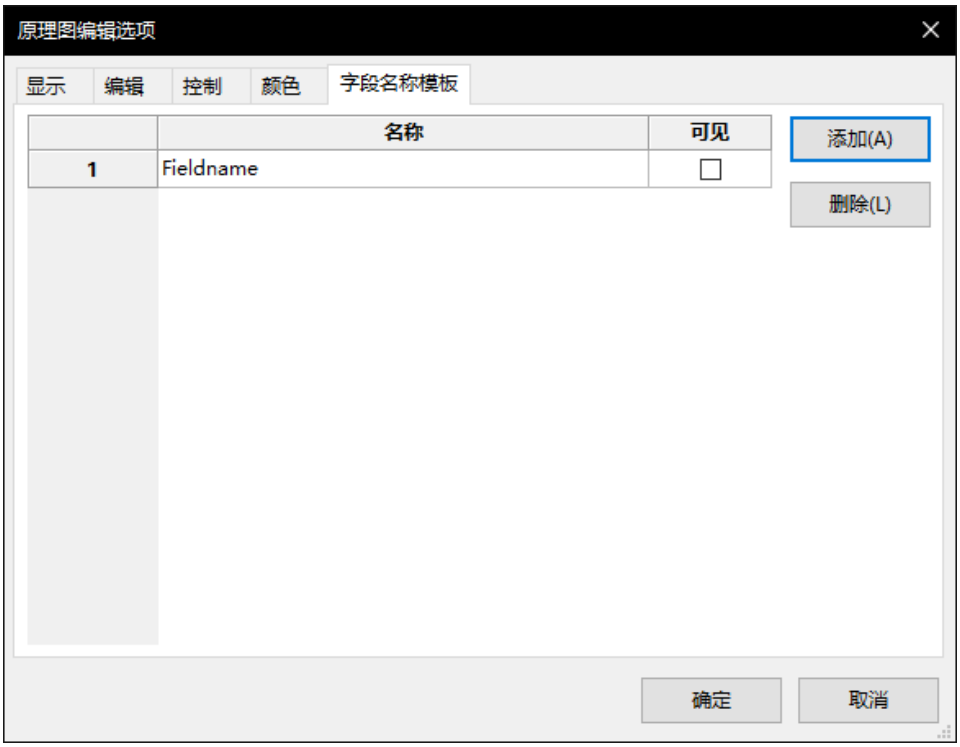
3.2.2.4 颜色

各种图形元素的配色方案。单击任何颜色样本以选择特定元素的新颜色。



3.2.2.5 默认字段

定义将在新放置的符号中显示的其他自定义字段和相应的值。



3.3 帮助菜单

访问在线帮助（本文档），获取有关 KiCad 的广泛教程。

在提交错误报告时使用“复制版本信息”来识别您的构建和系统。

Chapter 4

通用顶部工具栏

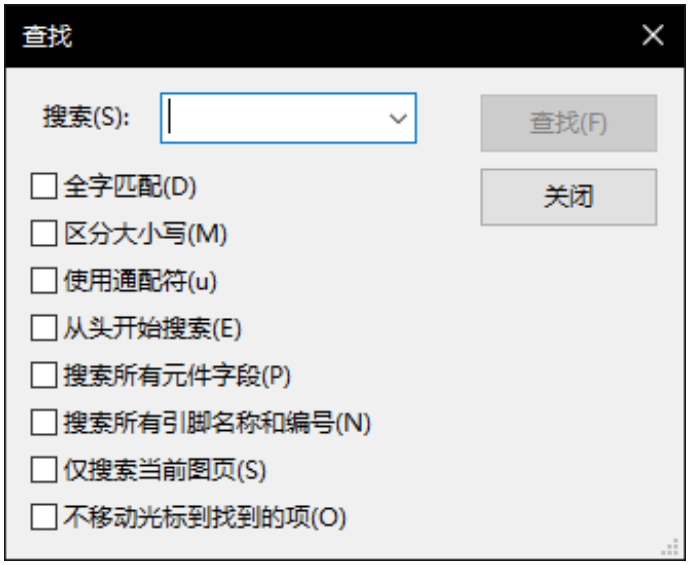
4.1 表格管理

“图纸设置”图标（  ）允许您定义图纸尺寸和标题栏的内容。

工作表编号会自动更新。您可以通过按“发布日期”按左箭头按钮将日期设置为今天，但不会自动更改。

4.2 搜索工具

“查找”图标（  ）可用于访问搜索工具。



您可以在当前工作表或整个层次结构中搜索引用，值或文本字符串。找到后，光标将定位在相关子表中的找到元素上。

4.3 网表工具

网表图标（  ）打开网表生成工具。

该工具创建一个文件，描述整个层次结构中的所有连接。

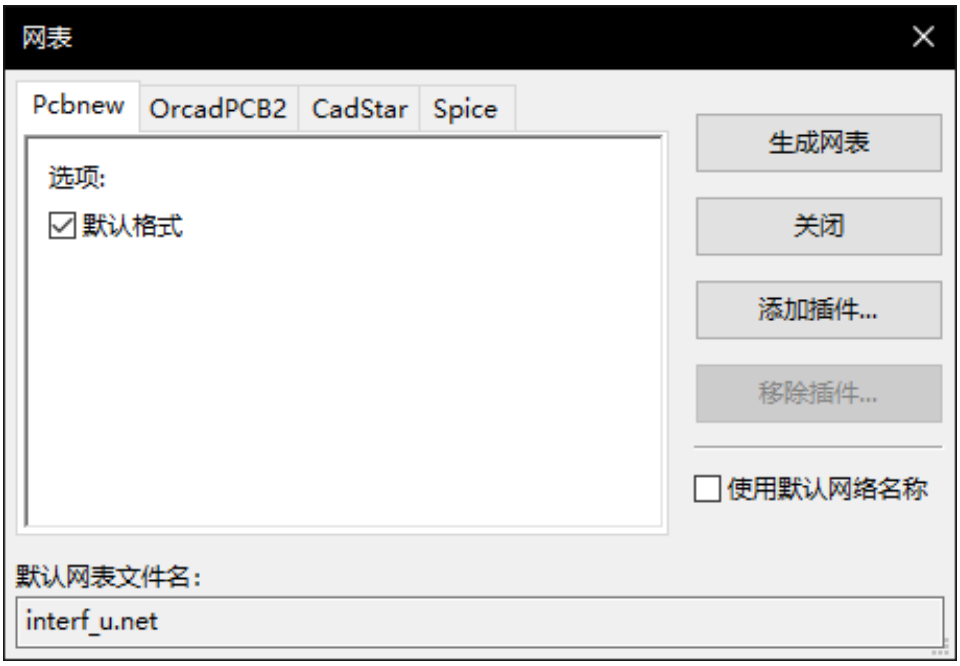
在多表层次结构中，任何本地标签仅在其所属的工作表内可见。例如：表 3 的标签 LABEL1 与表 5 的标签 LABEL1 不同（如果没有故意引入连接以连接它们）。这是因为工作表名称路径在内部与本地标签相关联。

Note

即使 Eeschema 中的标签没有文本长度限制，请考虑到读取生成的网表的其他程序可能存在此类限制。

Note

避免标签中的空格，因为它们将在生成的文件中显示为单独的单词。它不是 Eeschema 的限制，而是许多网表格式的限制，通常假设标签没有空格。



选项:

默认格式	选中以选择 Pcbnew 作为默认格式。

还可以生成其他格式:

- Orcad PCB2
- CadStar
- Spice (simulators)

可以添加外部插件来扩展网表格式列表（上图中添加了 PadsPcb 插件）。

有关在《create-a-netlist, Create a Netlist》一章中创建网表的更多信息。

4.4 注释工具

图标 启动注释工具。此工具分配对元件的引用。

对于多部件元件（例如包含 4 个门的 7400 TTL），还分配了多部件后缀（因此，指定为 U3 的 7400 TTL 将分为 U3A，U3B，U3C 和 U3D）。

您可以无条件地注释所有元件或仅注释新元件，即之前未注释的元件。

范围

使用整个原理图	所有工作表都重新注释（默认）。
仅使用当前页面	仅重新注释当前工作表（此选项仅在特殊情况下使用，例如评估当前表中的电阻数量。
保留现有注释	条件注释，只有新的元件将被重新注释（默认）。
重置现有注释	所有的无条件注释元件将被重新注释（此选项将在那里使用是重复的参考）。
重置，但不要交换任何带注释的多单元部件	保持当重新注释时，所有多个单元组（例如 U2A，U2B）在一起。

注释顺序

选择元件编号的顺序（水平或垂直）。

注释选择

选择指定的参考格式。

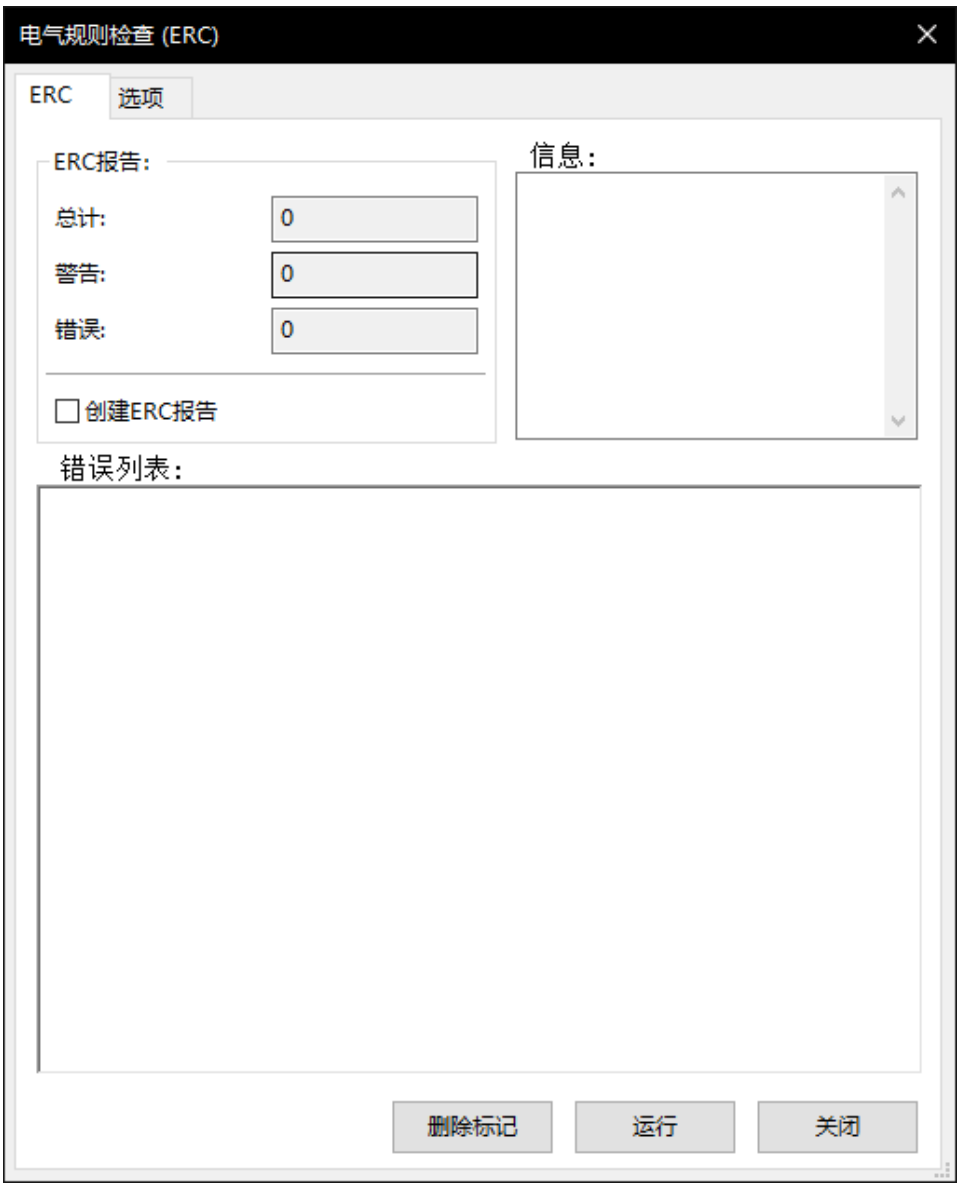
4.5 电气规则检查工具

图标  启动电子规则检查（ERC）工具。

该工具执行设计验证，能够检测被遗忘的连接和不一致。

运行 ERC 后，Eeschema 会放置标记以突出显示问题。左键单击标记后显示错误说明。还可以生成错误报告文件。

4.5.1 主要 ERC 对话框



错误显示在 Electrical Rules Checker 对话框中:

- 错误和警告的总数。
- 错误计数。
- 警告计数。

选项:

创建 ERC 文件报告	选中此选项可生成 ERC 报告文件。

命令:

删除标记	删除所有 ERC 错误/警告标记。
运行	启动电气规则检查。
关闭	关闭对话框。

- 单击错误消息将跳转到原理图中的相应标记。

4.5.2 ERC 选项对话框

此选项卡允许您定义引脚之间的连接规则; 您可以为每种情况选择 3 个选项:

- 无错误
- 警告
- 错误

可以通过单击修改单元格的每个方格。

选项:

测试类似标签	报告标签只有字母大小写（例如 lable/Lable/LaBeL）。网络名称区分大小写，因此这些标签被视为单独的网络。
测试独特的全局标签	报告仅出现一次的全局标签特别网。通常需要至少有两个连接。

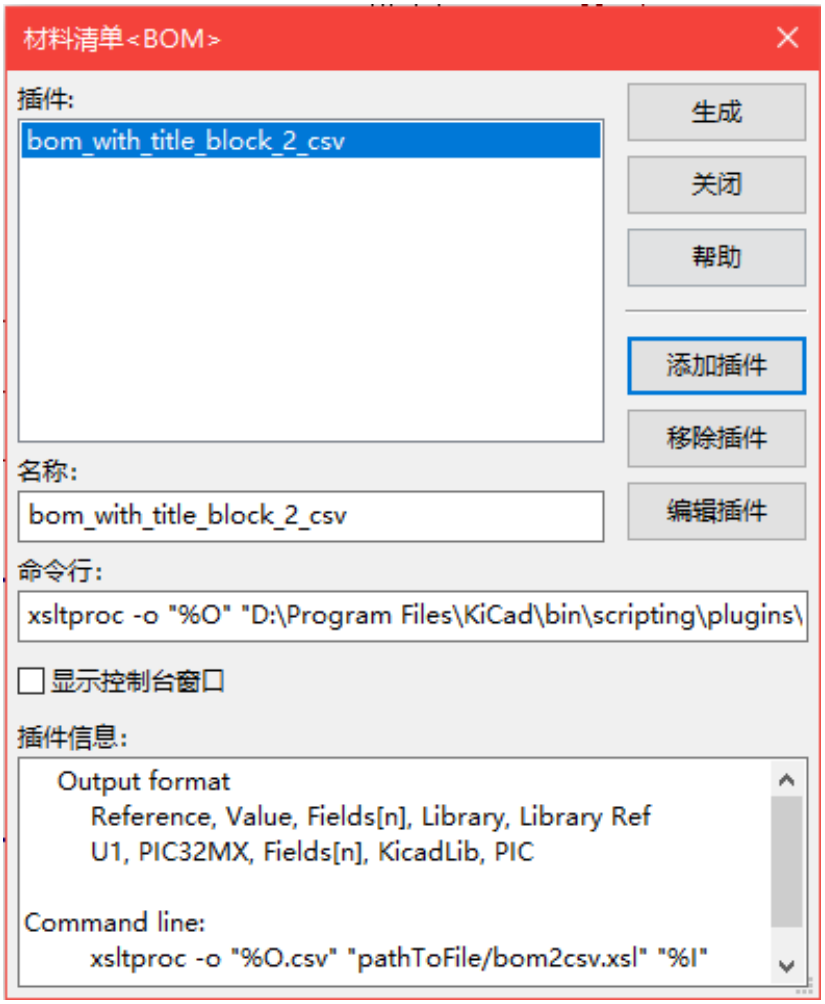
命令：

初始化为默认值	恢复原始设置。

4.6 物料清单工具

图标 

启动物料清单（BOM）生成器。此工具生成一个列出元件和/或分层连接（全局标签）的文件。

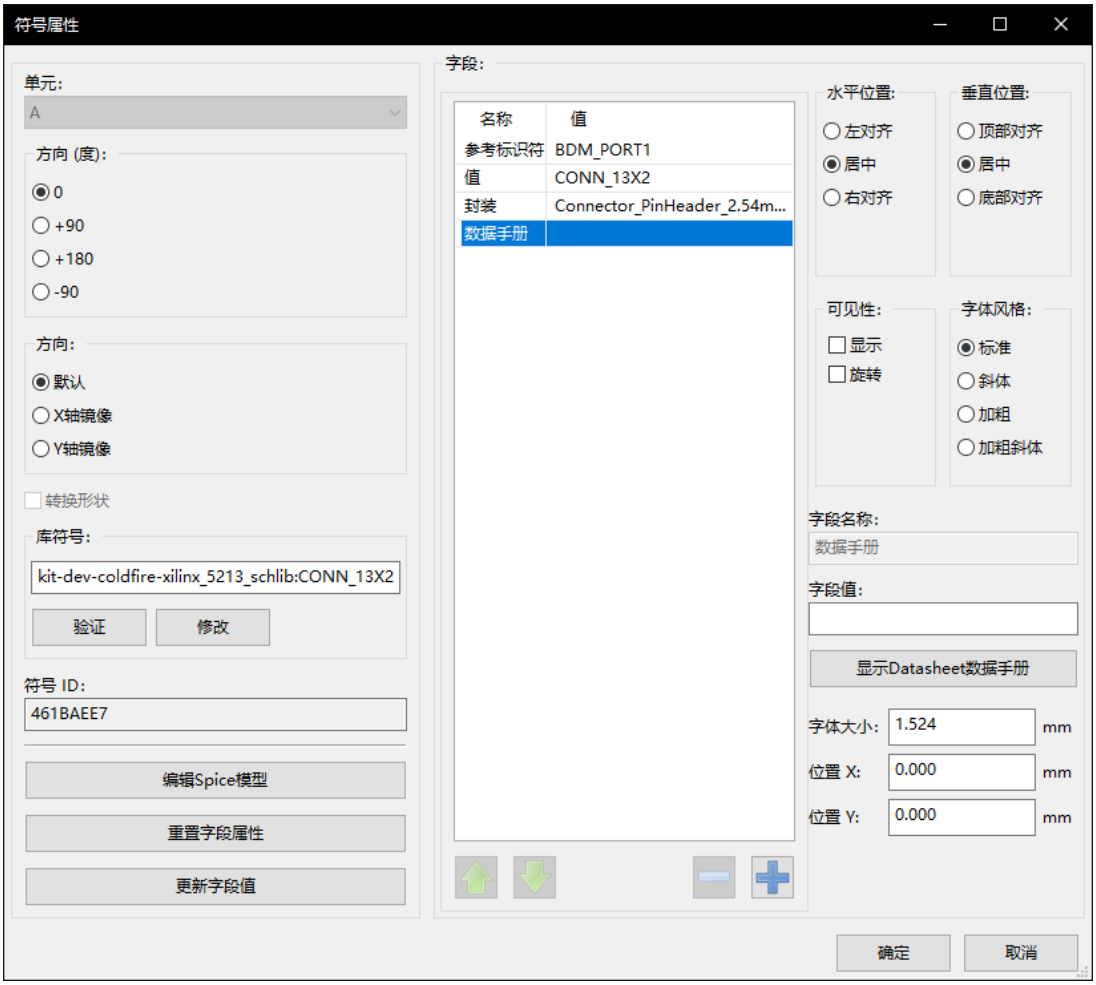


Eeschema 的 BOM 生成器使用外部插件，可以是 XSLT 或 Python 脚本。KiCad 程序文件目录中安装了一些示例。用于 BOM 的一组有用的元件属性包括：

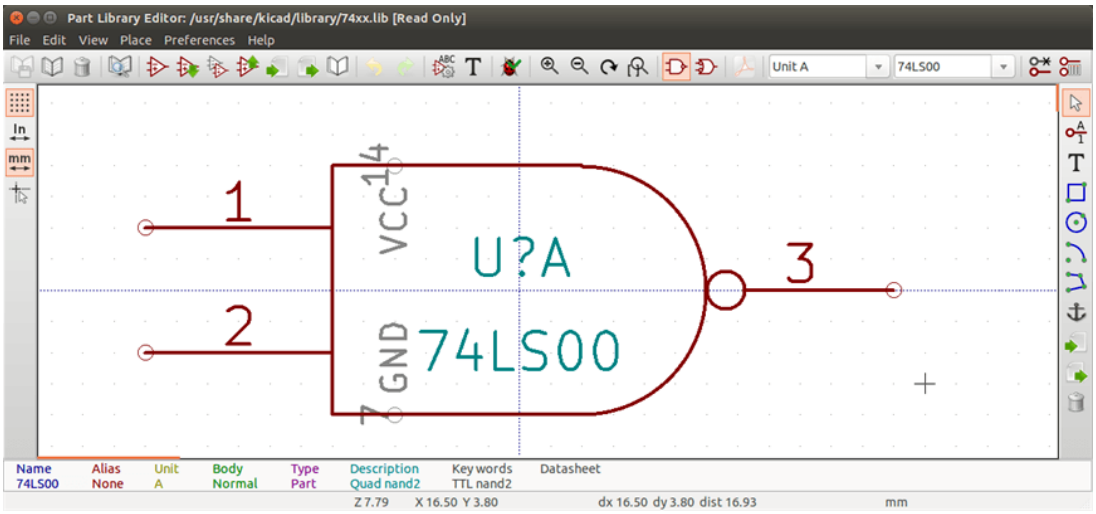
- 值 - 使用的每个部件的唯一名称。

- 封装 - 手动输入或反标注（见下文）。
- 字段 1 - 制造商的名称。
- 字段 2 - 制造商的元件号。
- 字段 3 - 分销商的元件号。

例如：

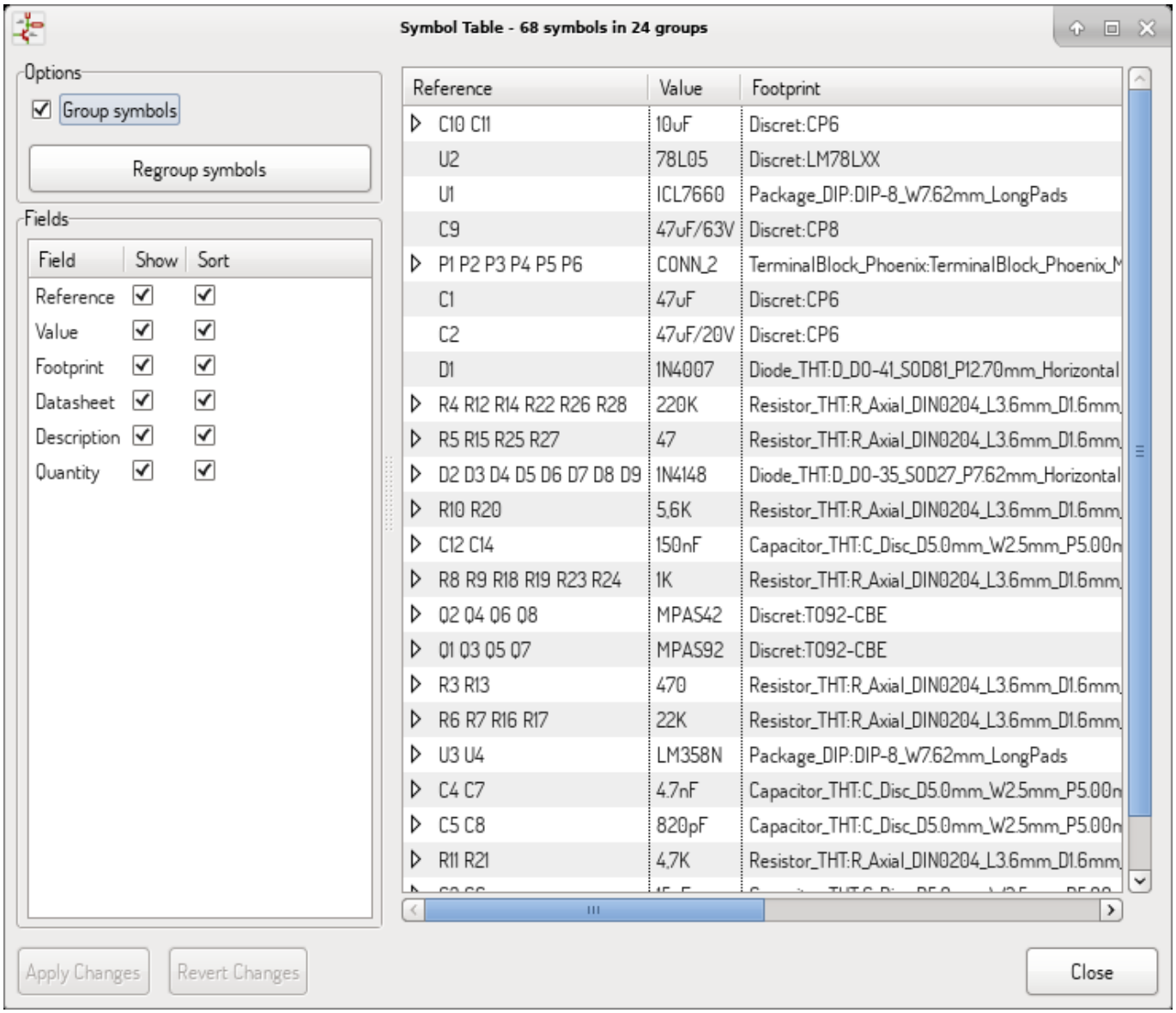


在 **MS Windows** 上，BOM 生成器对话框有一个特殊选项（由红色箭头指示），用于控制外部插件窗口的可见性。+ 默认情况下，BOM 生成器命令执行控制台窗口隐藏，输出重定向到 *Plugin info* 字段。设置此选项可显示正在运行的命令的窗口。如果插件提供了图形用户界面，则可能是必要的。



4.7 编辑字段工具

图标  打开电子表格以查看和修改所有符号的字段值。

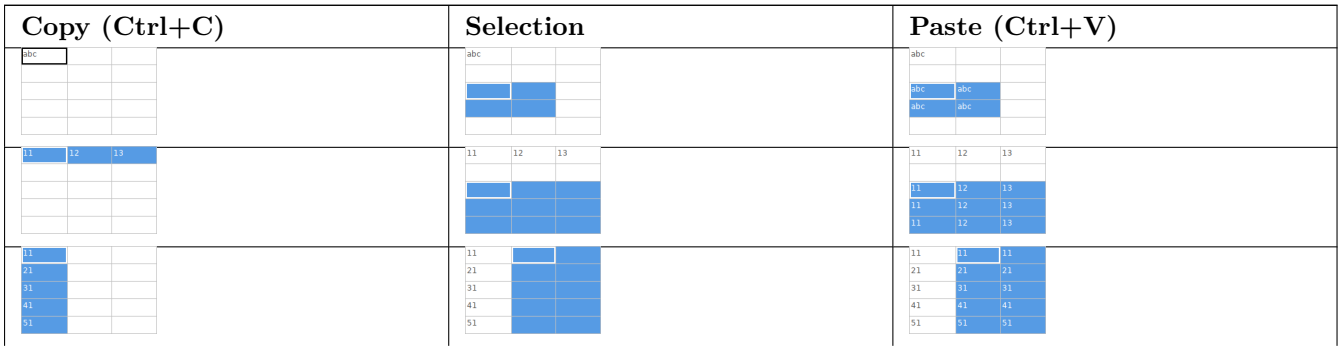


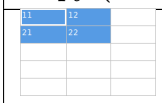
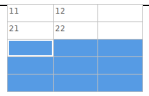
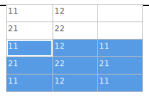
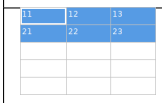
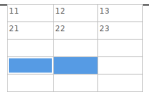
修改字段值后，您需要通过单击“应用”按钮接受更改，或通过单击 恢复按钮撤消更改。

4.7.1 简化字段填充的技巧

电子表格中有几种特殊的复制/粘贴方法。在输入在少数元件中重复的字段值时，它们可能很有用。

These methods are illustrated below.



Copy (Ctrl+C)	Selection	Paste (Ctrl+V)
		
		

Note

这些技术也可以在具有网格控制元素的其他对话框中使用。

4.8 用于封装分配的导入工具

4.8.1 访问：

图标 

启动反标注工具。

此工具允许将 PcbNew 中创建的封装更改导入 Eeschema 中的封装字段。

Chapter 5

管理符号库

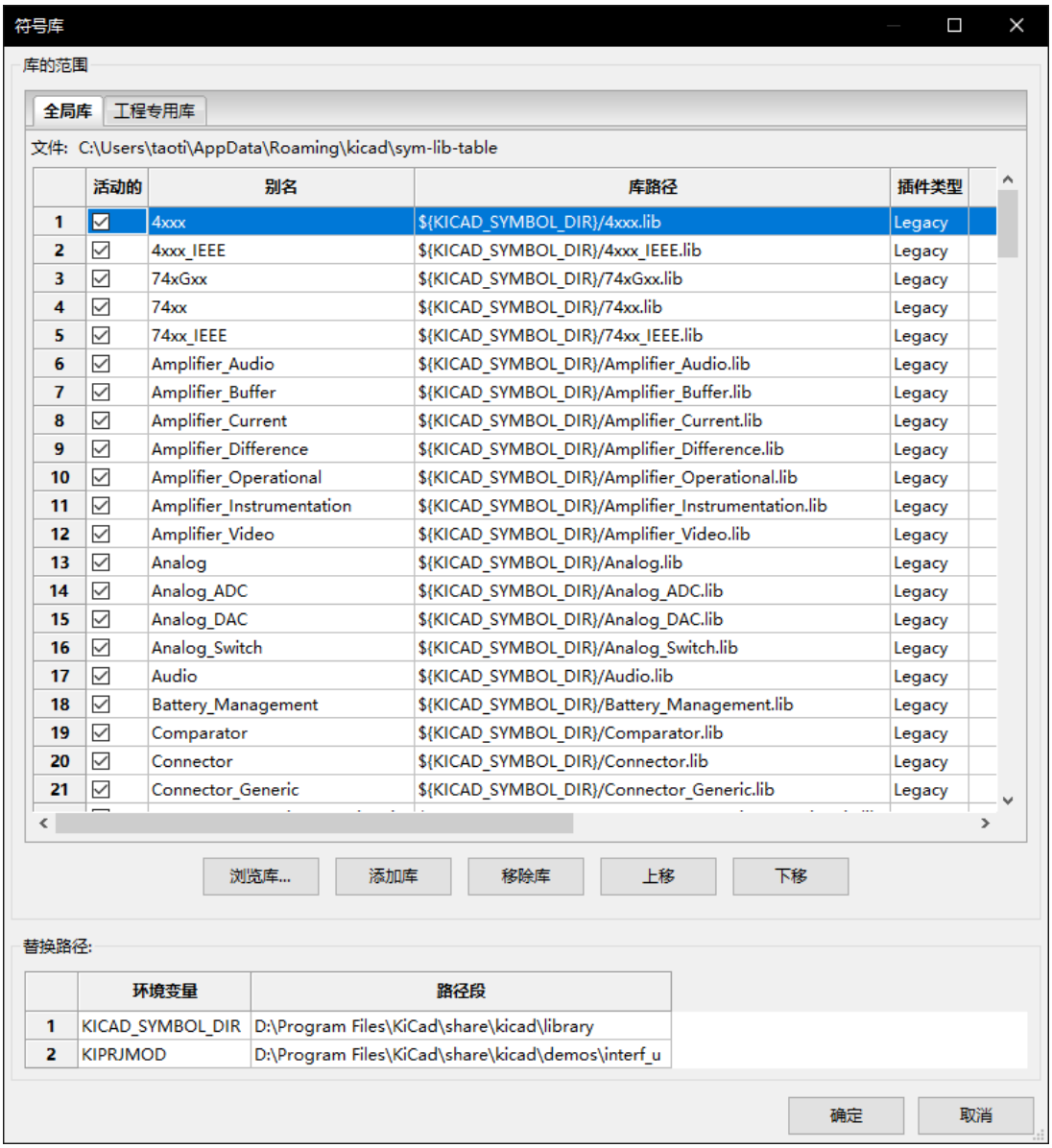
符号库包含创建原理图时使用的符号集合。原理图中的每个符号由一个全名唯一标识，该全名由库昵称和符号名称组成。一个例子是“音频：AD1853”。

5.1 符号库表

符号库表包含 KiCad 知道的所有库文件的列表。符号库表由全局符号库表文件和项目特定符号库表文件构成。

加载符号时，Eeschema 使用库昵称（在我们的示例中为“音频”）来查找符号库表中的库位置。

下图显示了符号库表编辑对话框，可以通过调用 首选项菜单中的 管理符号库表条目来打开该对话框。



5.1.1 全局符号库表

全局符号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的文件符号列表文件中。此文件夹的位置取决于所使用的操作系统。

5.1.2 项目特定符号库表

项目特定符号库表包含专门用于当前加载的项目文件的库列表。项目特定符号库表只能在与项目文件一起加载时进行编辑。如果未加载项目文件或当前项目路径中没有符号库表文件，则会创建一个空表，可以对其进行编辑，然后将其与项目文件一起保存。

5.1.3 初始配置

第一次运行 Eeschema 并且在用户的主文件夹中找不到全局符号表文件 **sym-lib-table** 时，Eeschema 将尝试复制存储在系统的 KiCad 模板文件夹中的默认符号表文件 sym-lib-table 到用户主文件夹中的文件 sym-lib-table。如果找不到默认模板 sym-lib-table 文件，则会出现一个对话框，提示输入 sym-lib-table 文件的备用位置。如果未找到 sym-lib-table 或解除对话框，则将在用户的主文件夹中创建空符号库表。如果发生这种情况，用户可以手动复制 sym-lib-table 或手动配置表。

Note

默认符号库表包括作为 KiCad 的一部分安装的所有符号库。根据用途和系统的速度，这可能是也可能不是所希望的。加载符号库所需的时间与符号库表中的库数量成正比。如果符号库加载时间过长，请从全局库表中删除很少和/或从未使用过的库，并根据需要将它们添加到项目库表中。

5.1.4 添加表项

为了使用符号库, 必须首先将其添加到全局表或项目特定表中。特定于项目的表仅适用于打开项目文件的情况。

每个图书馆条目必须有一个独特的昵称。

这不必以任何方式与实际库文件名或路径相关。冒号“:”和“/”字符不能在库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（参见下面的部分）。

必须选择适当的插件类型才能正确读取库。KiCad 目前仅支持旧版符号库文件插件。

还有一个描述字段用于添加库条目的描述。此时不使用选项字段，因此在加载库时添加选项将不起作用。

- 请注意，您不能在同一个表中包含重复的库昵称。但是，您可以在全局和项目特定的符号库表中包含重复的库昵称。
- 当出现重复的昵称时，项目特定的表条目将优先于全局表条目。
- 在项目特定表中定义条目时，包含这些条目的 sym-lib-table 文件将写入当前打开的项目文件的文件夹中。

5.1.5 环境变量替代

符号库表的最强大功能之一是环境变量替换。这允许定义符号库存储在环境变量中的自定义路径。使用库路径中的语法 `${ENV_VAR_NAME}` 支持环境变量替换。

默认情况下，在运行时 KiCad 定义 **两个环境变量**:

- `KIPRJMOD` 环境变量，始终指向当前打开的项目目录。`KIPRJMOD` 无法修改。
- `KICAD_SYMBOL_DIR` 环境变量。这指向使用 KiCad 安装的默认符号库的路径。

您可以重写 `KICAD_SYMBOL_DIR`, 方法是在“首选项/配置路径”中自己定义它, 该路径允许您替换自己的库, 以取代默认的 KiCad 符号库。

`KIPRJMOD` 允许您在没有项目路径的情况下存储库必须定义绝对路径（并不总是已知）项目特定符号库表中的库。

5.1.6 使用模式

符号库可以全局定义，也可以专门定义到当前加载的项目。用户全局表中定义的符号库始终可用，并存储在用户主文件夹的 sym-lib-table 文件中。项目特定符号库表仅对当前打开的项目文件有效。

每种方法都有优点和缺点。在全局表中定义所有库意味着它们将在需要时始终可用。这样做的缺点是加载时间会增加。

在项目特定的基础上定义所有符号库意味着您只有项目所需的库，这会减少符号库加载时间。缺点是您必须始终记住添加每个项目所需的每个符号库。

一种使用模式是全局定义常用库，而库只需要项目特定库表中的项目。对如何定义库没有限制。

5.1.7 遗留项目重新映射

加载在符号库表实现之前创建的原理图时，Eeschema 将尝试将原理图中的符号库链接重新映射到相应的库表符号。这一过程的成功取决于几个因素：

- 原理图中使用的原始库仍然可用，并且在符号添加到原理图时保持不变。
- 在检测到所有恢复行动时，执行所有恢复行动以创建恢复库或使现有恢复库保持最新状态。
- 项目符号缓存库的完整性尚未损坏。

Warning

重新映射将备份在重新映射期间在项目文件夹中的 rescue-backup 文件夹中更改的所有文件。在重新映射之前，请务必备份项目以防万一出错。

Warning

即使已禁用恢复操作以执行恢复操作以确保正确的符号可用于重新映射。请勿取消此操作，否则重映射将无法正确重新映射原理图符号。任何损坏的符号链接都必须手动修复。

Note

如果已删除原始库并且未执行恢复，则可以将缓存库用作恢复库作为最后的手段。将缓存库复制到新文件名，并在符号库表实现之前使用 Eeschema 版本将新库文件添加到库列表的顶部。

Chapter 6

原理图创建和编辑

6.1 简介

原理图可以用单张纸表示，但是，如果足够大，则需要多张纸。

由几张纸表示的示意图是分层的，并且其所有纸张（每个纸张由其自己的文件表示）构成 Eeschema 项目。分层原理图的操作将在《hierarchical-schematics，Hierarchical Schematics》章节中描述。

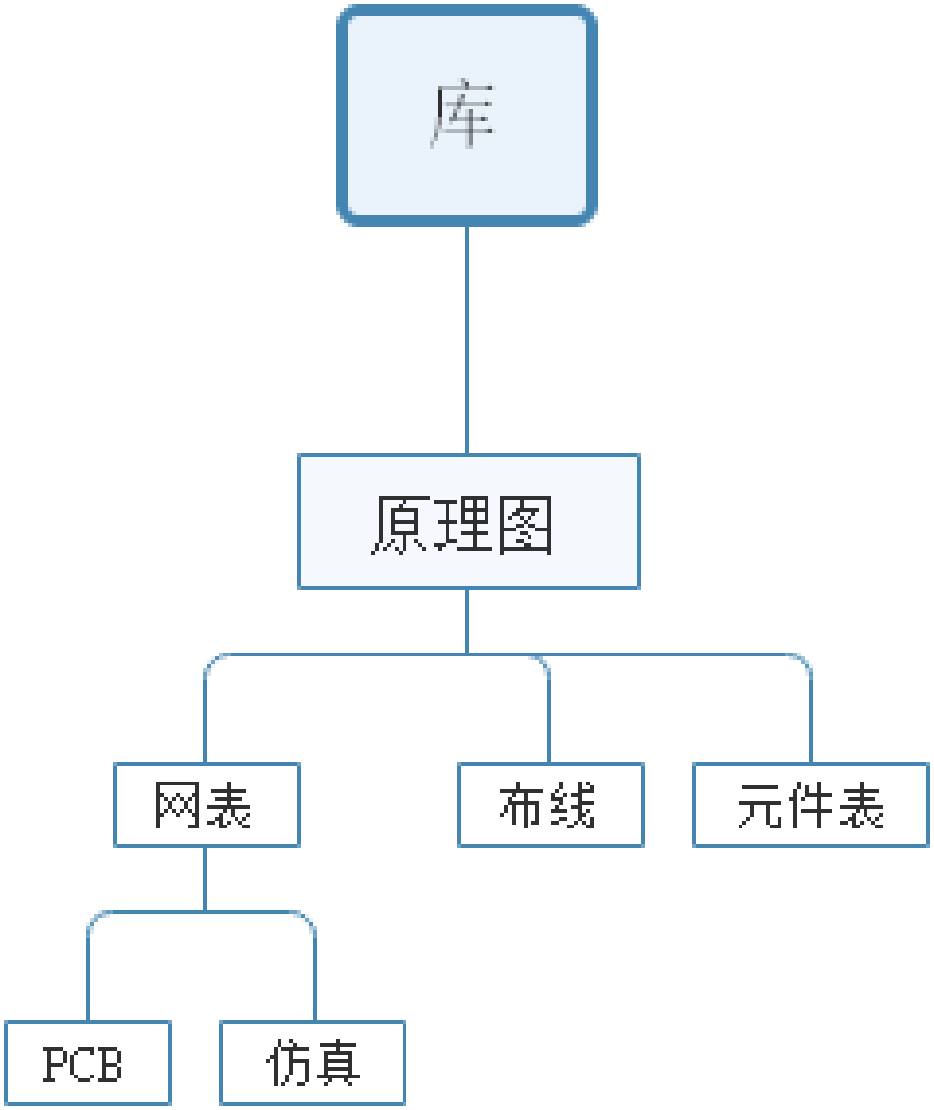
6.2 一般考虑

使用 Eeschema 设计的原理图不仅仅是电子设备的简单图形表示。它通常是开发链的入口点，允许：

- 验证一组规则（ERC，电气规则检查）以检测错误和遗漏。
- 自动生成物料清单（BOM）。
- 用于仿真软件（如 SPICE）的（创建 - 定制 - 网表和文件 - 文件，生成网表）。
- （创建 - 定制 - 网表和网络文件，生成网表），用于传输到 PCB 布局。

原理图主要由符号，电线，标签，连接点，总线和电源端口组成。为了清晰起见，您可以放置纯粹的图形元素，如总线条目，注释和折线。

6.3 开发链

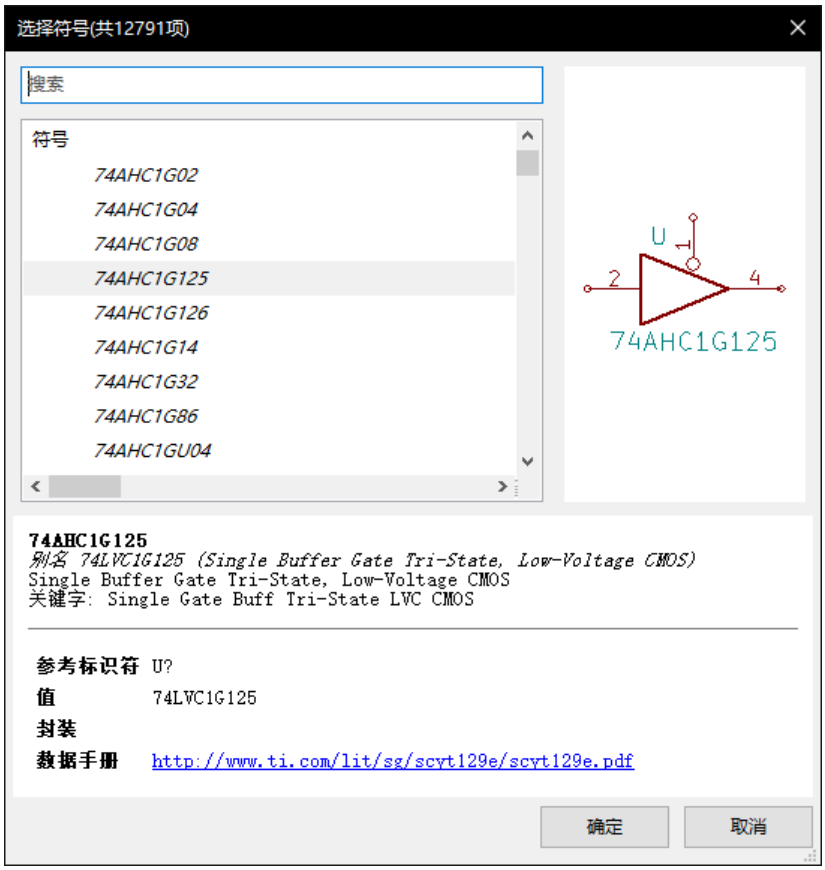


符号从符号库添加到原理图中。在制作原理图之后，生成一个网表，稍后用于将连接和封装集导入 PcbNew。

6.4 符号放置和编辑

6.4.1 找到并放置一个符号

要将符号加载到原理图中，可以使用图标 。使用对话框可以键入要加载的符号的名称。



“选择符号”对话框将根据您在搜索字段中键入的内容按名称，关键字和说明过滤符号。只需输入高级过滤器即可使用它们：

- **通配符：**分别使用字符“?”和“*”表示“任意字符”和“任意数量的字符”。
- **关联：**如果库部分的描述或关键字包含标签格式为“Key：123”，您可以通过键入相对于该匹配“Key> 123”（大于），“Key <123”（小于）等。数字可能包括以下不区分大小写的后缀之一：

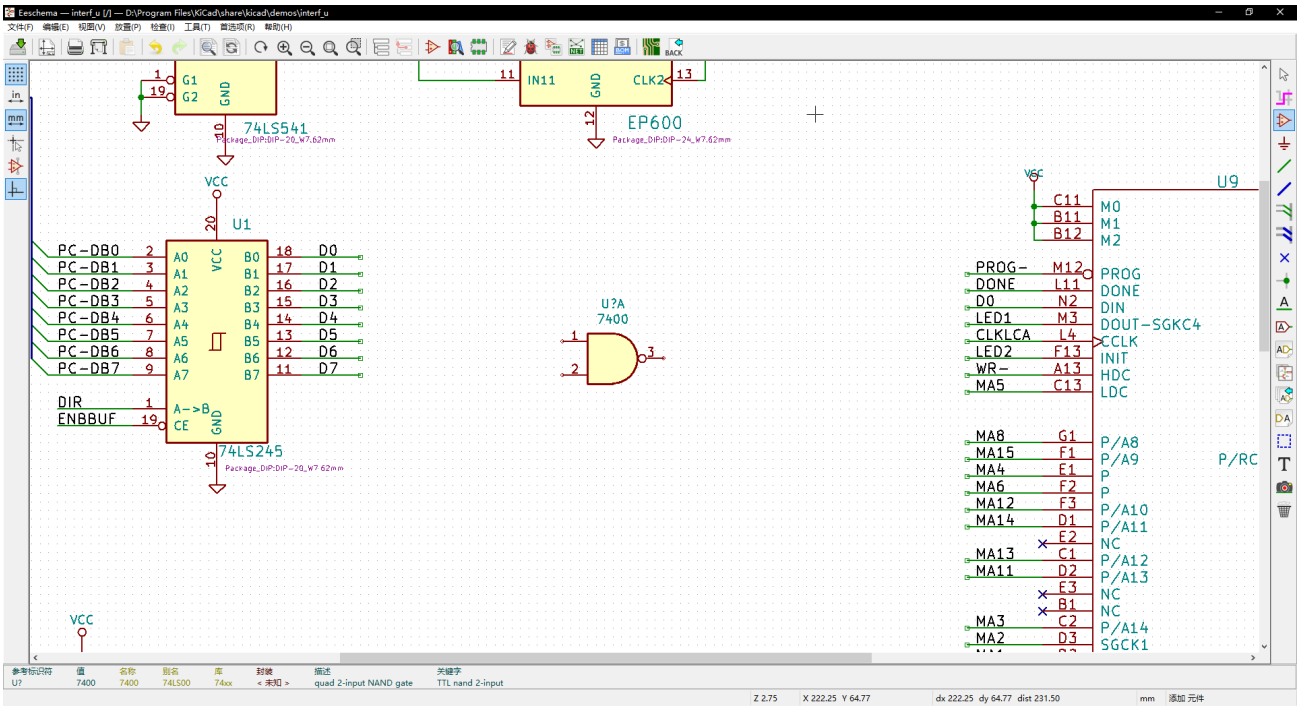
p	n	u	m	k	meg	g	t
10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{3}	10^{6}	10^{9}	10^{12}

ki	mi	gi	ti
2^{10}	2^{20}	2^{30}	2^{40}

- **正则表达式：**如果你熟悉正则表达式，这些也可以用。使用的正则表达式风味是 wxWidgets 高级正则表达式 ，类似于 Perl 常规表达式。

在将符号放置在原理图中之前，您可以使用热键或右键单击上下文菜单对其进行旋转，镜像和编辑其字段。这可以在放置后以相同的方式完成。

这是放置期间的符号：



6.4.2 电源端口

电源端口符号是符号（符号在 电源库中分组），因此可以使用符号选择器放置它们。但是，由于电源放置频繁，因此可以使用  工具。这个工具很相似，只是搜索直接在 电源库中完成。

6.4.3 符号编辑和修改（已放置的元件）

编辑符号有两种方法：

- 符号本身的修改：多单元符号上的位置，方向，单位选择。
- 修改符号的其中一个字段：引用，值，覆盖区等。

刚刚放置符号时，您可能需要修改其值（特别是电阻器，电容器等），但是立即为其分配参考编号或选择单元是没有用的（除了元件之外）锁定单位，您必须手动分配）。这可以通过注释功能自动完成。

6.4.3.1 符号修改

要修改符号的某些功能，请将光标放在符号上，然后执行以下任一操作：

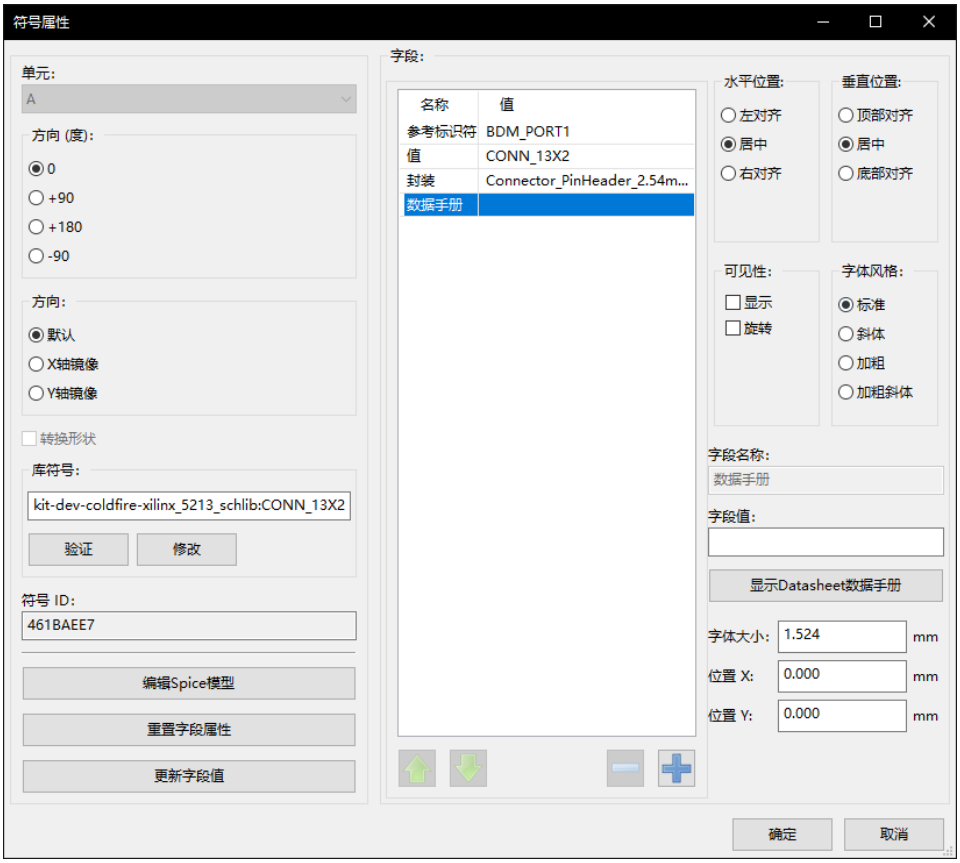
- 双击符号以打开完整的编辑对话框。
- 右键单击以打开上下文菜单并使用以下命令之一：移动，方向，编辑，删除等。

6.4.3.2 文本字段修改

您可以修改字段的参考，值，位置，方向，文本大小和可见性：

- 双击文本字段进行修改。
- 右键单击以打开上下文菜单并使用以下命令之一：移动，旋转，编辑，删除等。

要获得更多选项，或者要创建字段，请双击该符号以打开 符号属性对话框。



每个字段都可以是可见的或隐藏的，并且可以水平或垂直显示。始终为正常显示的符号（无旋转或镜像）指示显示的位置，并且相对于符号的锚点。

重置为库默认值选项将符号设置为原始方向，并重置每个字段的选项，大小和位置。但是，文本字段不会被修改，因为这可能会破坏原理图。

6.5 电线，总线，标签，电源端口

6.5.1 简介

所有这些绘图元素也可以与垂直右侧工具栏上的工具一起放置。

这些元素是:

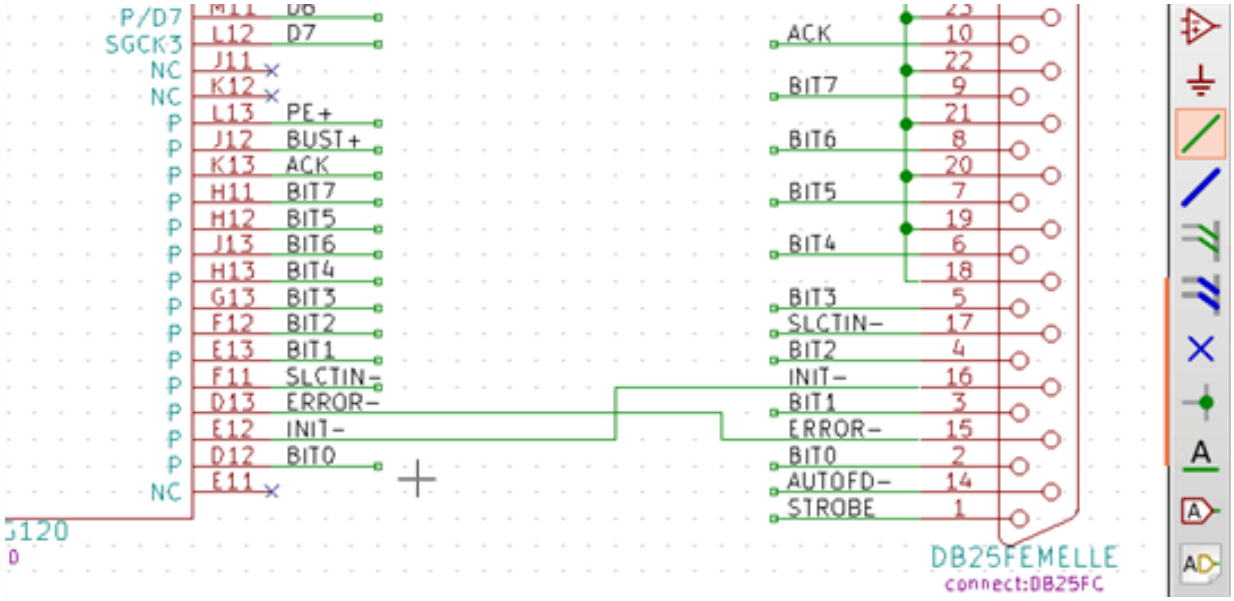
- **电线：**符号之间的大多数连接。
- **总线：**以图形方式加入总线标签
- **折线：**用于图形演示。
- **连接点：**用于在交叉线或总线之间建立连接。
- **总线入口：**显示电线和总线之间的连接。
- **标签：**用于标记或创建连接。
- **全局标签：**用于表格之间的连接。
- **文本：**用于评论和注释。
- **“无连接”标志：**终止不需要任何连接的引脚。
- **分层表**及其连接引脚。

6.5.2 连接（电线和标签）

有两种方法可以建立连接：

- 引脚到引脚的电线。
- 标签。

下图显示了这两种方法：



注 1：

标签的 接触点是标签第一个字母的左下角。未连接时，此点显示为小滑块。

因此，该点必须与导线接触，或者叠加在销的末端，以使标签看起来是连接的。

注 2:

要建立连接，必须将一段导线的两端连接到另一个段或一个引脚。

如果有重叠（如果导线通过引脚，但没有连接到引脚端）则没有连接。

注 3:

交叉的电线不是隐式连接的。如果需要连接，则必须将它们与连接点连接。

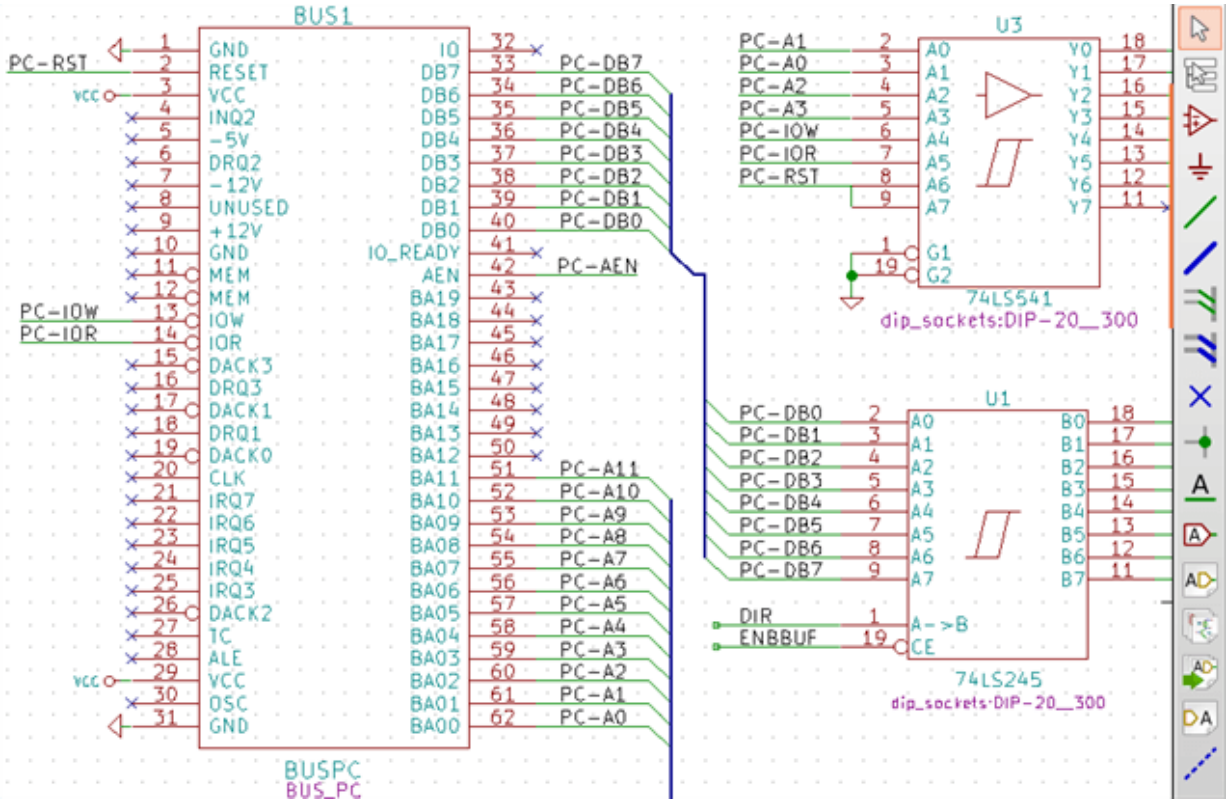
前面的图（连接到 DB25FEMALE 引脚 22,21,20,19 的导线）显示了使用结符号的连接情况。

注 4:

如果两个不同的标签放在同一根电线上，它们会连接在一起并变得相同：连接到一个或另一个标签的所有其他元件然后连接到所有标签。

6.5.3 连接（总线）

在下面的原理图中，许多引脚连接到总线。



6.5.3.1 总线编号

从示意图的角度来看，总线是一组信号，以公共前缀开头，以数字结尾。例如，PCA0，PCA1 和 PCA2 是 PCA 总线的成员。

完整的总线名为 PCA[N..m]，其中 N 和 m 是该总线的第一个和最后一个线号。因此，如果 PCA 有 20 个成员，从 0 到 19，整个总线记为 PCA[0..19]。总线中不能包含 PCA0，PCA1，PCA2，WRITE，READ 等信号的集合。

6.5.3.2 总线成员之间的连接

总线成员之间的连接连接在总线的相同成员之间的连接必须通过标签连接。无法将引脚直接连接到总线; Eeschema 将忽略这种类型的连接。

在上面的示例中，连接是通过放置在连接到引脚的导线上的标签进行的。到总线的总线入口（45 度线段）仅是图形化的，并不是形成逻辑连接所必需的。

实际上，使用重复命令（*Insert* 键），如果元件引脚按递增顺序排列，则可以通过以下方式快速建立连接（实际上在存储器，微处理器等元件上的常见情况）：

- 放置第一个标签（例如 PCA0）
- 尽可能多地使用重复命令来放置成员。Eeschema 将自动创建垂直对齐的下一个标签（PCA1，PCA2 ……），理论上是在其他引脚的位置上。
- 在第一个标签下画线。然后使用重复命令将其他导线放在标签下。
- 如果需要，以相同的方式放置总线条目（放置第一个条目，然后使用重复命令）。

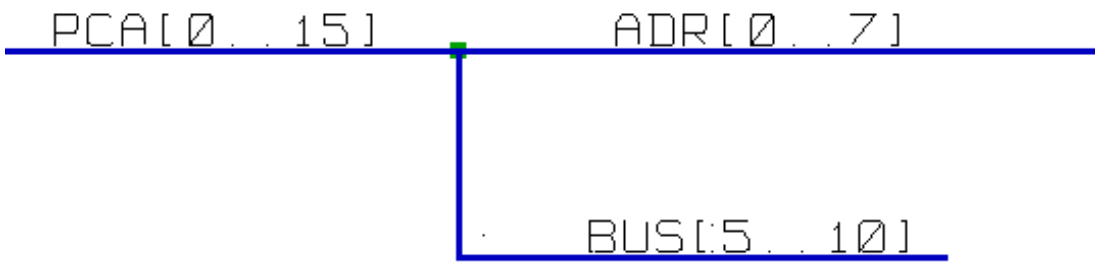
Note

在“首选项/选项”菜单中，您可以设置重复参数：

- **垂直步骤**。
- **水平步骤**。
- **标签增量（因此可以递增 2,3 或递减）**。

6.5.3.3 总线之间的全局连接

您可能需要总线之间的连接，以便链接具有不同名称的两个总线，或者在层次结构的情况下，以在不同的表之间创建连接。您可以通过以下方式建立这些连接。



总线 PCA[0..15]，ADR[0..7] 和 BUS[5..10] 连接在一起（注意这里的连接点，因为垂直总线连接到水平总线段的中间）。

更确切地说，相应的成员连接在一起：连接 PCA0，ADR0（与 PCA1 和 ADR1 …PCA7 和 ADR7 相同）。

此外，PCA5，BUS5 和 ADR5 连接（就像 PCA6，BUS6 和 ADR6 一样，如 PCA7，BUS7 和 ADR7）。

PCA8 和 BUS8 也连接（就像 PCA9 和 BUS9，PCA10 和 BUS10 一样）

6.5.4 电源端口连接

当符号的电源引脚可见时，它们必须连接，就像任何其他信号一样。

门和触发器等符号可能有不可见的电源引脚。必须小心这些因为：

- 由于它们不可见，你无法连接电线。
- 你不知道他们的名字。

此外，将它们视为可见并将它们像其他引脚一样连接将是一个坏主意，因为原理图将变得不可读并且不符合通常的惯例。

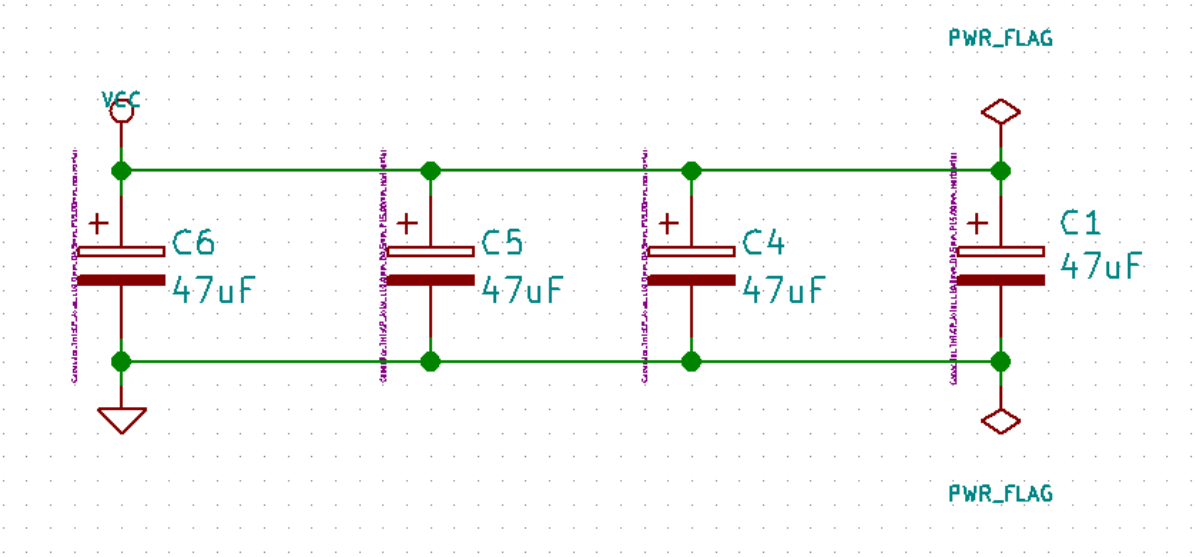
Note

如果要强制显示这些不可见的电源引脚，必须在主菜单的 首选项/选项对话框中选中 显示不可见的电源引脚选项，或者图标  **左侧（选项）工具栏上的。**

Eeschema 自动将同名的隐形电源引脚连接到该名称的电源网。可能需要连接不同名称的电源网（例如，TTL 元件中的 *GND* 和 MOS 元件中的 *VSS* ）；为此使用电源端口。

建议不要使用标签进行电源连接。它们只有一个 _ 本地 _ 连接范围，不会连接不可见的电源引脚。

下图显示了电源端口连接的示例。



在该示例中，地（GND）连接到电源端口 VSS，电源端口 VCC 连接到 VDD。

可以看到两个 PWR_FLAG 符号。它们表明两个电源端口 VCC 和 GND 确实连接到电源。如果没有这两个标志，ERC 工具将诊断出：警告：电源端口未通电 _。

所有这些符号都可以在“电源”符号库中找到。

6.5.5 “无连接”标志

这些符号对于避免意外的 ERC 警告非常有用。电气规则检查确保没有连接意外地保持未连接状态。

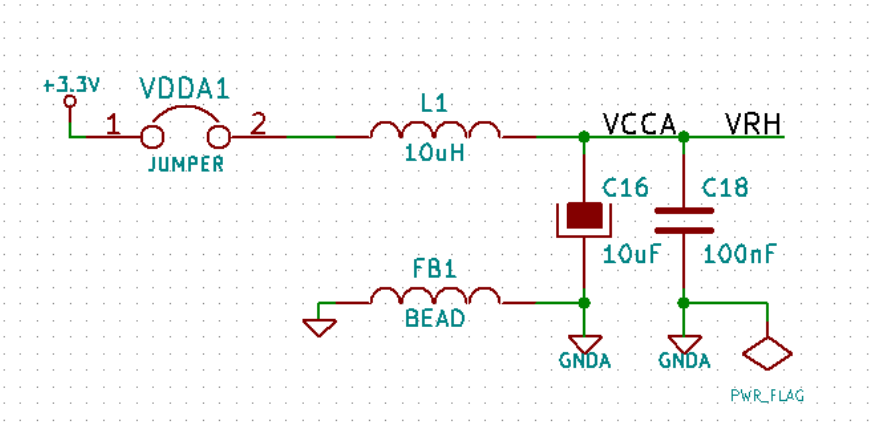
如果引脚必须保持未连接状态，则必须在这些引脚上放置“无连接”标志（工具  ）。这些符号对生成的网表没有任何影响。

6.6 绘图补充

6.6.1 文本注释

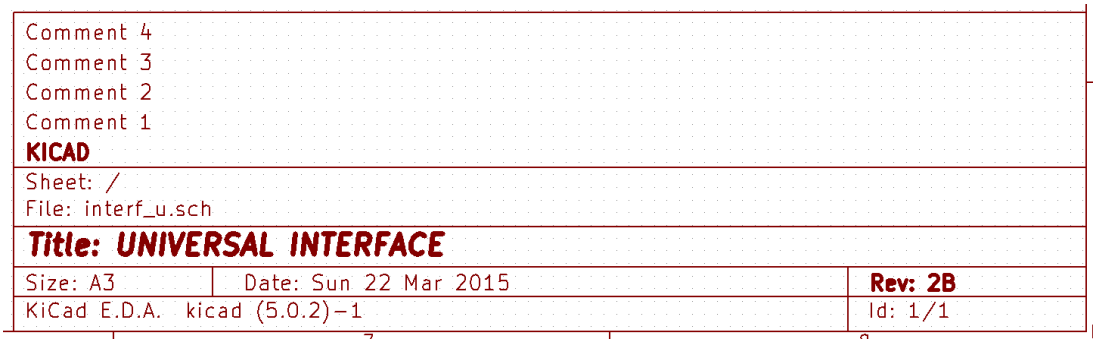
放置注释 (例如文本字段和框架) 可能很有用 (有助于理解原理图)。文本字段 (工具 ) 和多段线 (工具 ) 用于此用途，而标签和导线是连接元素。

在这里，您可以找到带有文本注释的框架示例。



6.6.2 表格标题栏

使用工具图像编辑标题栏：  。

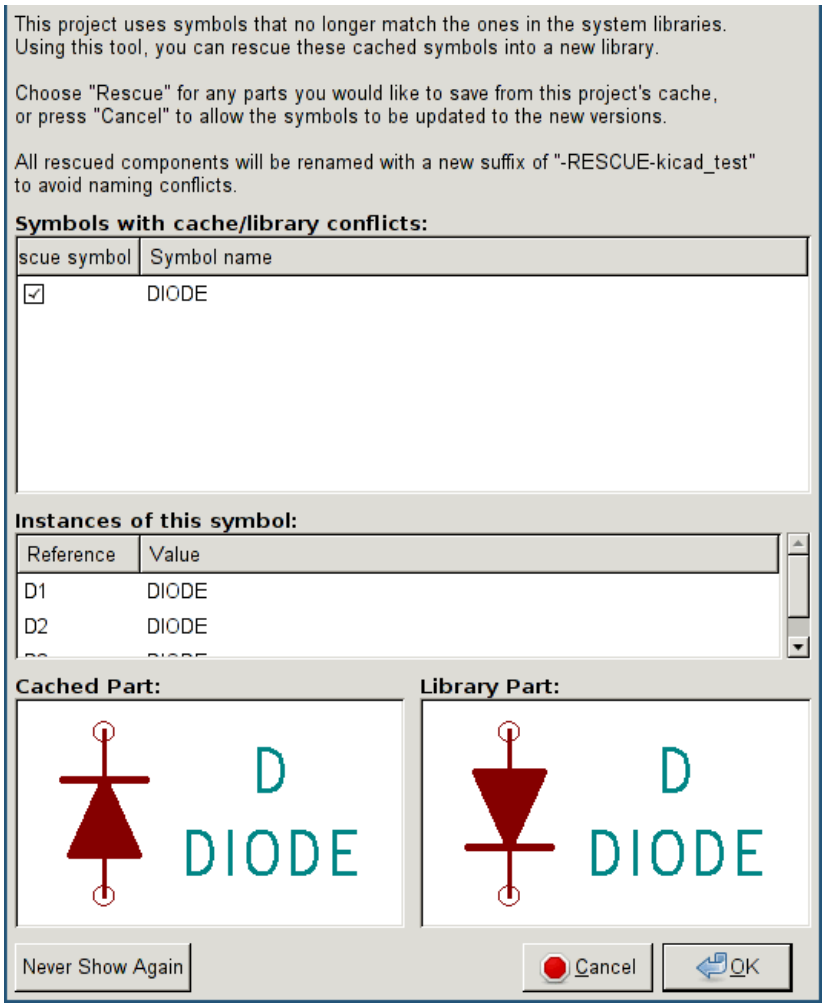


工作表编号（工作表 X / Y）会自动更新。

6.7 抢救缓存的符号

默认情况下，Eeschema 根据设置的路径和从项目库加载符号库顺序。这在加载非常旧的项目时可能会导致问题：如果库已更改或已被删除，或者库自使用后已不存在。在项目中，项目中的项目将自动替换为新版本。新版本可能没有正确对齐，或者可能方向不同，导致出现。破损的原理图。

保存项目时，将包含具有当前库符号内容的缓存库以及原理图。这允许在没有完整库的情况下分发项目。如果加载其缓存和系统库中存在符号的项目，Eeschema 将扫描库以查找冲突。找到的任何冲突都将在以下对话框中列出：



您可以在此示例中看到该项目最初使用的是阴极朝上的二极管，但现在库中包含阴极朝下的二极管。这种改变会打破原理图！在此处按 OK 将使符号缓存库保存到特殊的 恢复库中，并重命名所有符号以避免命名冲突。

如果按 取消，则不会进行任何恢复，因此 Eeschema 默认会加载所有新元件。如果此时保存原理图，则将覆盖缓存并且旧符号将无法恢复。如果已保存原理图，则仍可以通过选择“工具”菜单中的“恢复缓存元件”再次调用恢复对话框，再次运行恢复功能。

如果您不想看到此对话框，可以按 从不再显示。默认设置是不执行任何操作并允许加载新元件。可以在 库首选项中更改此选项。

Chapter 7

分层原理图

7.1 简介

对于大于几张的项目，分层表示通常是一个很好的解决方案。如果要管理此类项目，则需要：

- 使用大纸张会导致打印和处理问题。
- 使用多个工作表，这将引导您进入层次结构。

然后，完整的原理图包含一个主要的原理图表，称为根表，以及构成该层次结构的子表。此外，巧妙地将设计细分为单独的表格通常会提高其可读性。

从根表中，您必须能够找到所有子表。借助可通过顶部工具栏的图标  访问的集成 *层次结构导航器*，Eeschema 可以轻松实现分层原理图管理。

有两种类型的层次结构可以同时存在：第一种层次结构刚刚被唤起并且具有普遍用途。第二个包括在库中创建符号，这些符号在原理图中看起来像传统符号，但实际上对应于描述其内部结构的示意图。

第二种类型用于开发集成电路，因为在这种情况下，您必须在绘制的原理图中使用函数库。

Eeschema 目前不会处理第二种情况。

层次结构可以是:

- 简单: 给定的工作表只使用一次
- 复杂: 给定的工作表被多次使用 (倍数实例)
- 平面: 这是一个简单的层次结构, 但不会绘制工作表之间的连接。

Eeschema 可以处理所有这些层次结构。

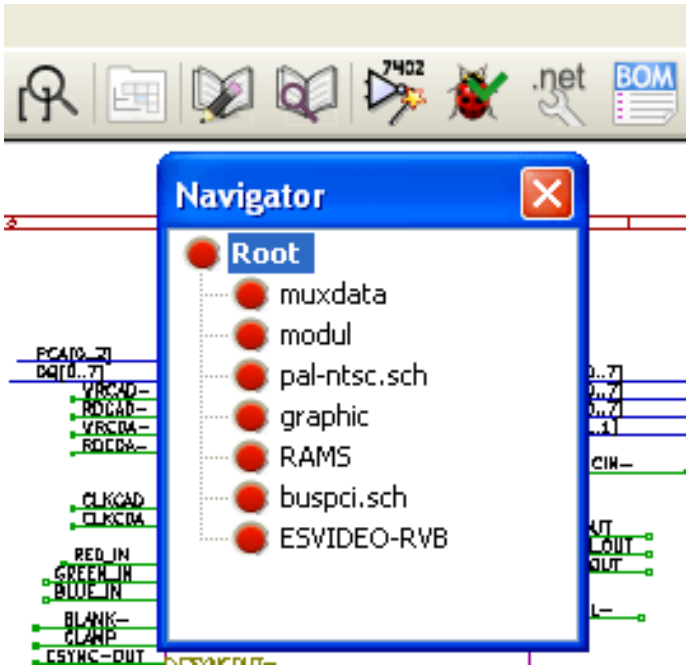
创建分层原理图很容易，整个层次结构从根原理图开始处理，就像您只有一个原理图一样。

要理解的两个重要步骤是：

- 如何创建子表。
- 如何在子表之间建立电气连接。

7.2 在层次结构中导航

通过使用可通过顶部工具栏上的按钮 访问的导航工具来实现子表之间的导航。



单击其名称即可访问每个工作表。要快速访问，请右键单击工作表名称，然后选择“输入工作表”或双击工作表的范围。

要将当前工作表退出到父工作表，请右键单击原理图中没有对象的任何位置，然后在上下文菜单中选择“离开工作表”或按“Alt + Backspace”。

7.3 本地、分层和全局标签

7.3.1 属性

本地标签，工具 ，仅在工作表内连接信号。分层标签（工具 ）仅在工作表内连接信号，并连接到父工作表中的分层引脚。

全局标签（工具 ）连接所有层次结构中的信号。电源引脚（类型 电源输入和 电源输出）不可见就像全局标签一样，因为它们在所有层次结构中被视为连接在它们之间。

Note

在层次结构（简单或复杂）中，可以使用分层标签和/或全局标签。

7.4 层次结构创建摘要

您必须:

- 在根工作表中放置一个名为 工作表符号的层次结构符号。
- 使用导航器进入新原理图（子工作表）并绘制它，就像任何其他原理图一样。
- 通过将全局标签（HLabels）放在新的原理图（子表）中，并在根表中使用相同名称的标签（称为 SheetLabels）绘制两个原理图之间的电气连接。这些 SheetLabel 将连接到根表的工作表符号，连接到原理图的其他元素，如标准符号引脚。

7.5 工作表符号

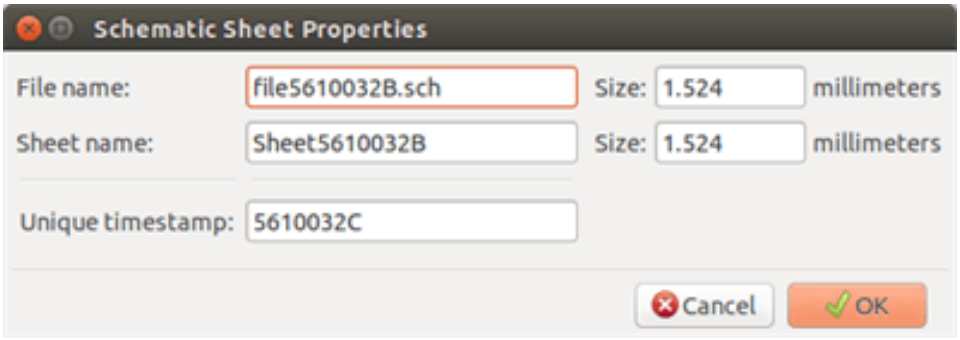
绘制一个由两个对角点定义的矩形，表示子表格。

此矩形的大小必须允许您放置以后特定标签，层次结构引脚，对应于子表中的全局标签（HLabels）。

这些标签类似于通常的符号引脚。选择工具  。

单击以放置矩形的左上角。再次单击以放置右下角，具有足够大的矩形。

然后，系统将提示您为此子表单键入文件名和表单名称（以便使用层次结构导航器访问相应的原理图）。



您必须至少提供一个文件名。如果没有工作表名称，则文件名将用作工作表名称（通常的方法）。

7.6 连接 - 分层引脚

您将在此处创建刚刚创建的符号的连接点（层次结构引脚）。

这些连接点类似于普通符号引脚，但是只需一个连接点就可以连接一个完整的总线。

有两种方法可以做到这一点:

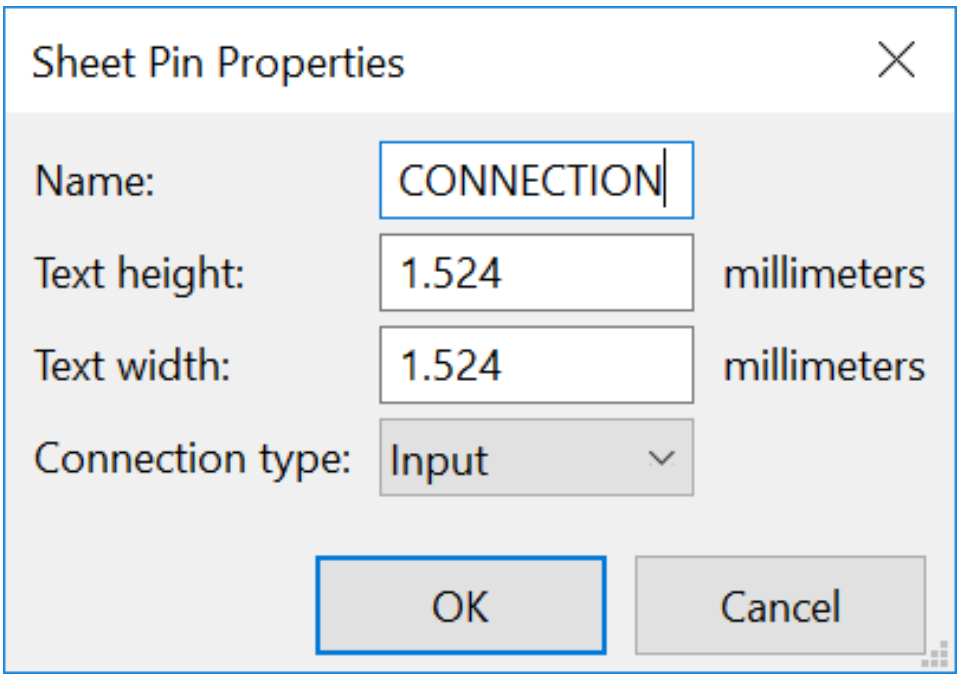
- 在绘制子表之前放置不同的引脚（手动放置）。
- 在绘制子表和全局标签（半自动放置）后放置不同的引脚。

第二种解决方案是非常优选的。

手动放置：

- 选择工具  。
- 单击要放置引脚的层次结构符号。

有关创建名为“连接”的分层引脚的示例，请参见下文：



您可以在创建期间或之后定义引脚的名称，大小和方向，方法是右键单击引脚并在弹出菜单中选择 *编辑图纸引脚*。

在工作表内部，必须使用与“分层引脚”相同的名称预设“分层标签”。注意正确匹配这些名称必须手动完成，这就是下面第二种方法首选的原因。

自动放置：

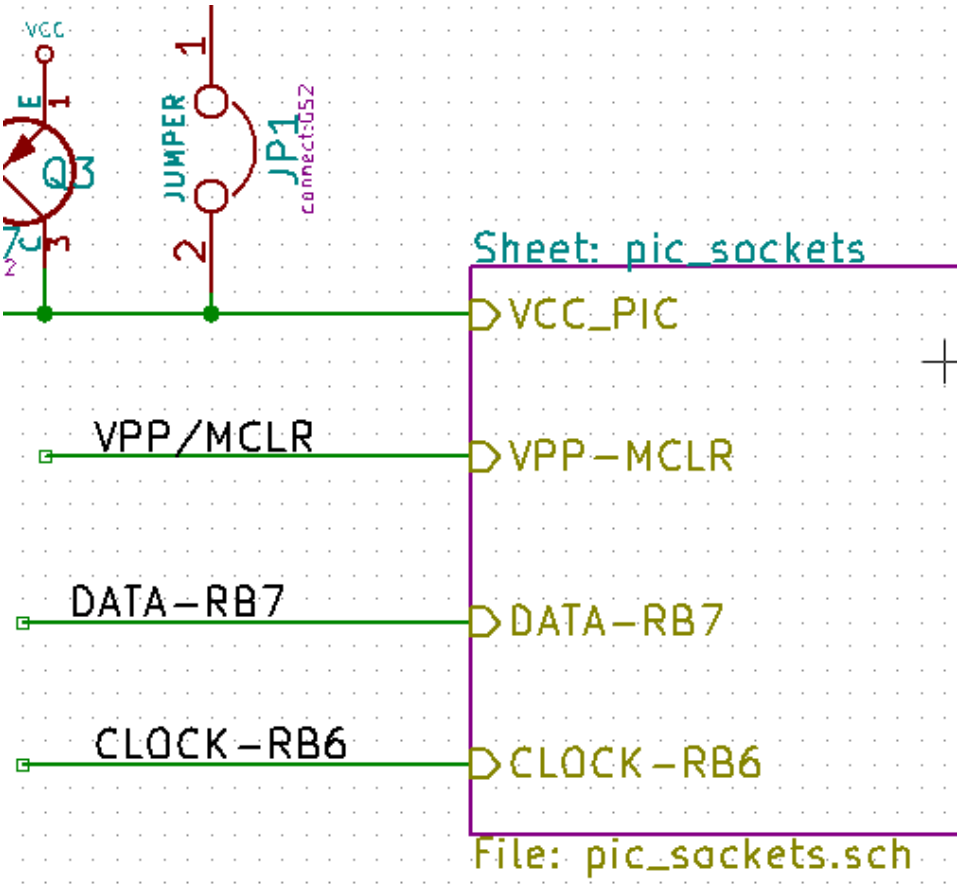
- 选择工具 image:images/icons/import_hierarchical_label.png[icons/import_hierarchical_label_png] 。
- 单击要从中导入与相应原理图中放置的全局标签对应的引脚的层次结构符号。如果存在新的全局标签，则出现分层引脚，即不对应于已经放置的引脚。
- 单击要放置此引脚的位置。

因此可以快速且无误地放置所有必需的引脚。他们的方面符合相应的全局标签。

7.7 连接 - 分层标签

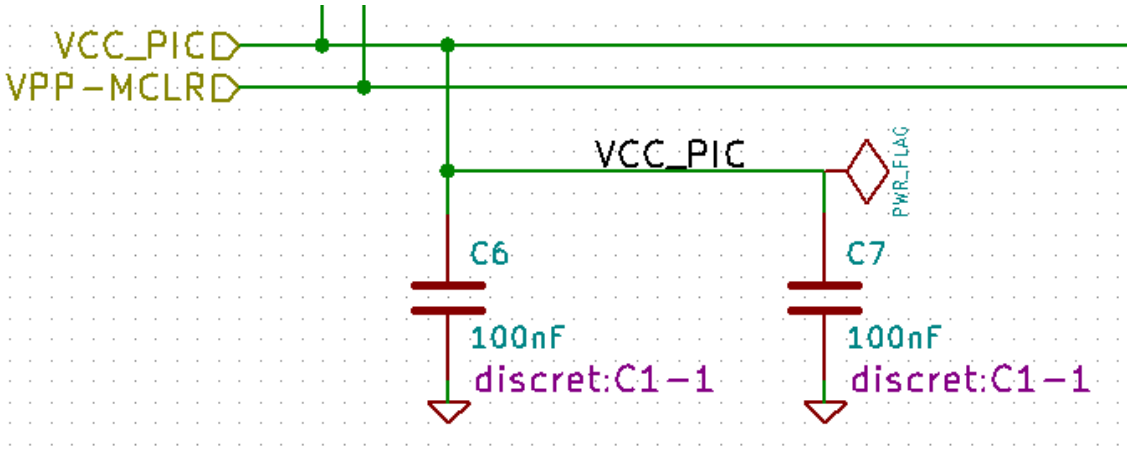
刚刚创建的工作表符号的每个引脚必须对应于子工作表中名为分层标签的标签。分层标签与标签类似，但它们提供子表和根表之间的连接。两个互补标签（pin 和 HLabel）的图形表示是类似的。使用工具图像创建分层标签：  。

请参阅下面的根表示例：



注意引脚 VCC_PIC，连接到连接器 JP1。

以下是子表中的相应连接：



您再次找到两个相应的分层标签，提供两个分层表之间的连接。

Note

根据前面描述的语法（Bus [N. .m]），您可以使用分层标签和层次结构引脚连接两条总线。

7.7.1 标签，分层标签，全局标签和隐形电源引脚

以下是有关提供连接的各种方法的一些注释，而不是有线连接。

7.7.1.1 简单的标签

简单标签具有局部连接能力，即限于放置它们的示意图。这是因为:

- 每张纸都有一个纸张编号。
- 此工作表编号与标签相关联。

因此，如果在标签 n°3 中放置标签 “TOTO”，实际上真正的标签是 “TOTO_3”。如果您还在工作表 n°1（根表）中放置了标签 “TOTO”，则实际上放置一个名为 “TOTO_1” 的标签，与 “TOTO_3” 不同。即使只有一张纸也是如此。

7.7.1.2 分层标签

对于简单标签而言，对于分层标签也是如此。

因此，在同一张纸中，分层标签 “TOTO” 被认为连接到本地标签 “TOTO”，但没有连接到另一张纸中称为 “TOTO” 的分层标签或标签。

分层标签被认为连接到放置在父表中的分层符号中的相应表引脚符号。

7.7.1.3 隐形电源引脚

可以看出，如果隐形电源引脚具有相同的名称，它们将连接在一起。因此，所有声称为 *隐形电源引脚*并命名为 VCC 的电源引脚都将所有符号 VCC 的电源引脚连接在它们所放置的工作表内。

这意味着如果将 VCC 标签放在子表中，它将不会连接到 VCC 引脚，因为该标签实际上是 VCC_n，其中 n 是表单号。

如果您希望此标签 VCC 真正连接到整个原理图的 VCC，则必须通过 VCC 电源符号将其明确连接到不可见的电源引脚。

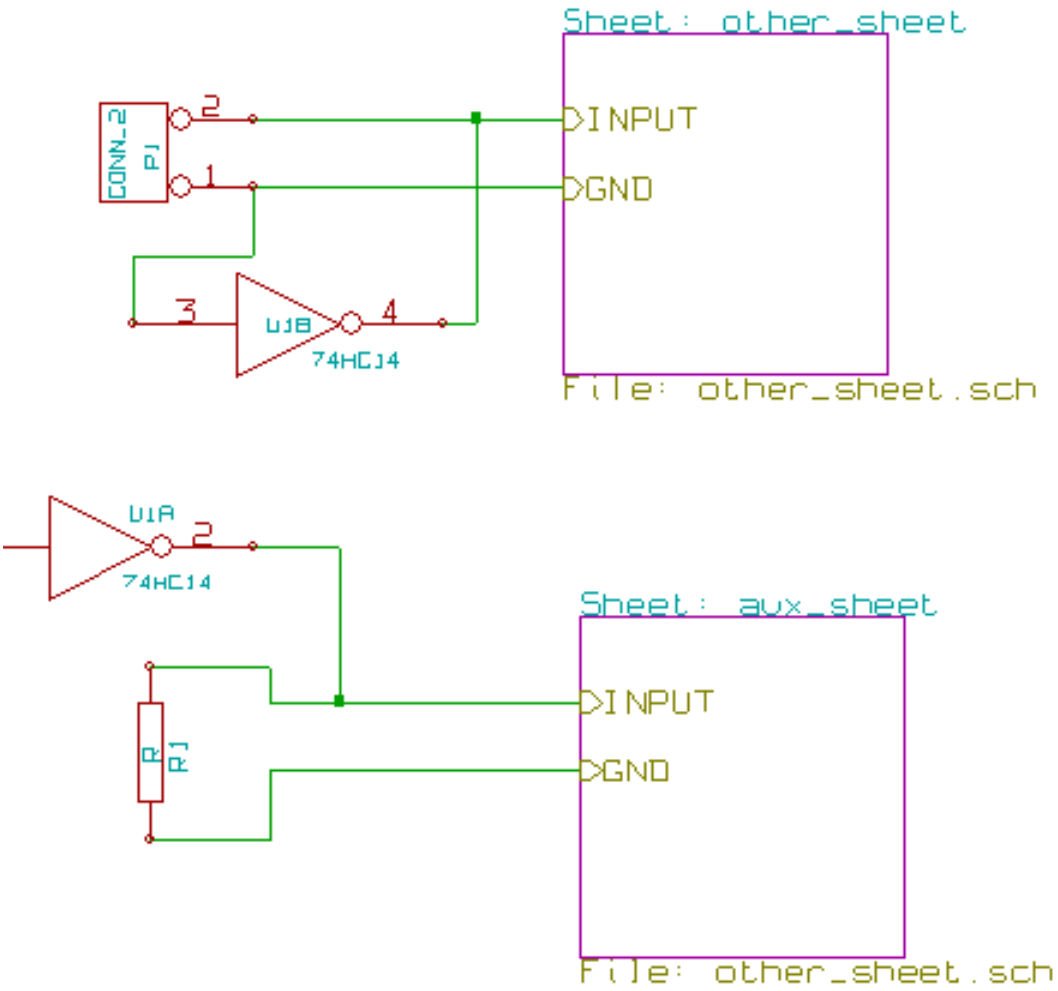
7.7.2 全局标签

具有相同名称的全局标签跨整个层次结构连接。

（像 vcc 这样的强力标签……是全局标签）

7.8 复杂层次结构

这是一个例子。相同的原理图使用两次（两个实例）。这两个工作表共享相同的原理图，因为两个工作表的文件名相同（*other_sheet.sch* ）。工作表名称必须是唯一的。

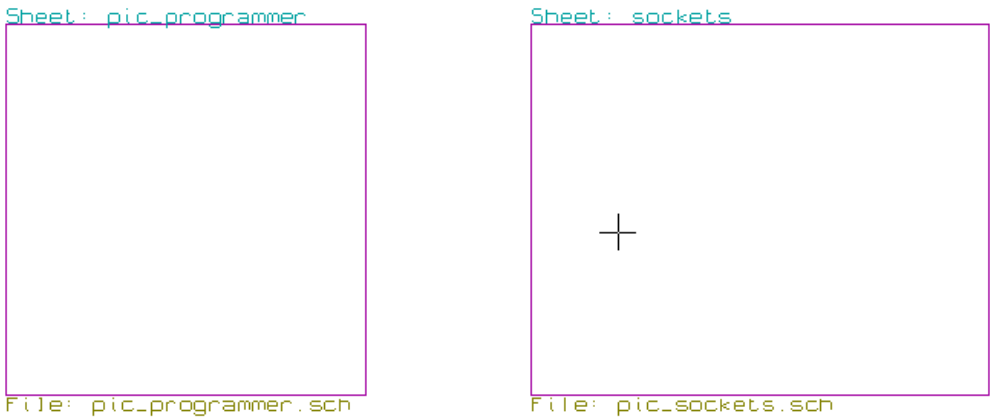


7.9 平面层次结构

如果遵守以下规则，则可以使用多个工作表创建项目，而无需在这些工作表（平面层次结构）之间创建连接：

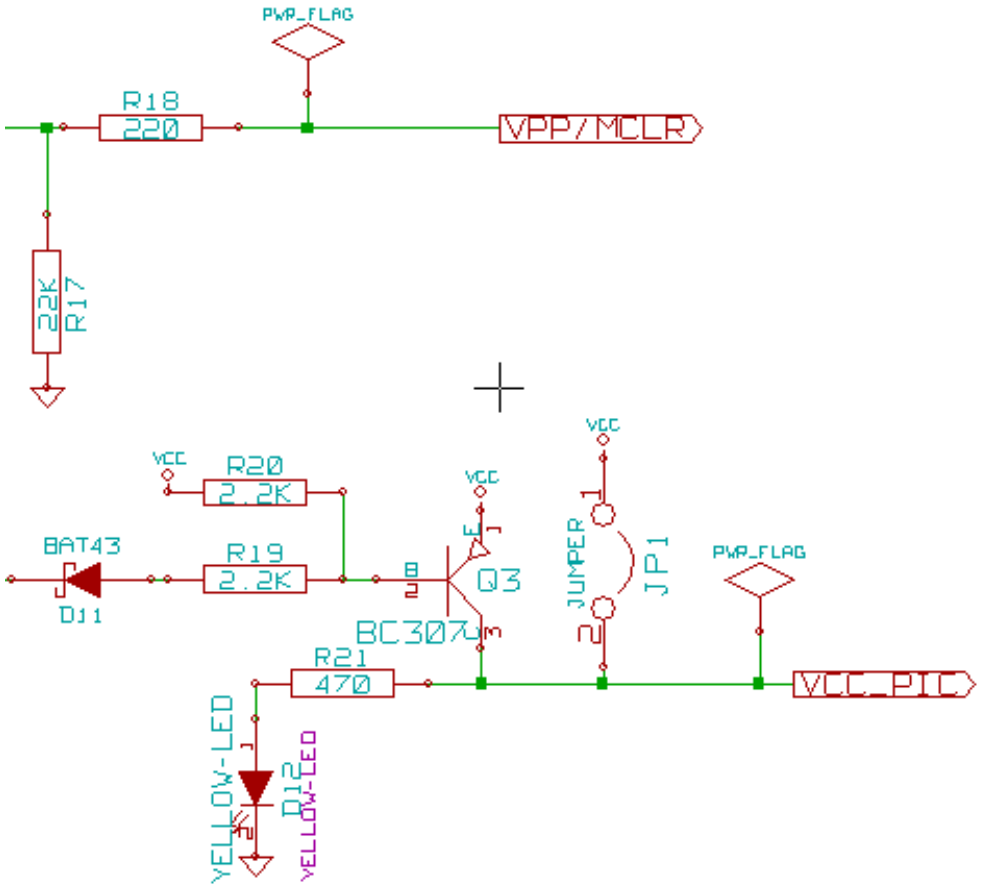
- 创建包含其他工作表的根工作表，这些工作表充当其他工作表之间的链接。
- 不需要明确的连接。
- 在所有工作表中使用全局标签而不是分层标签。

以下是根表的示例。

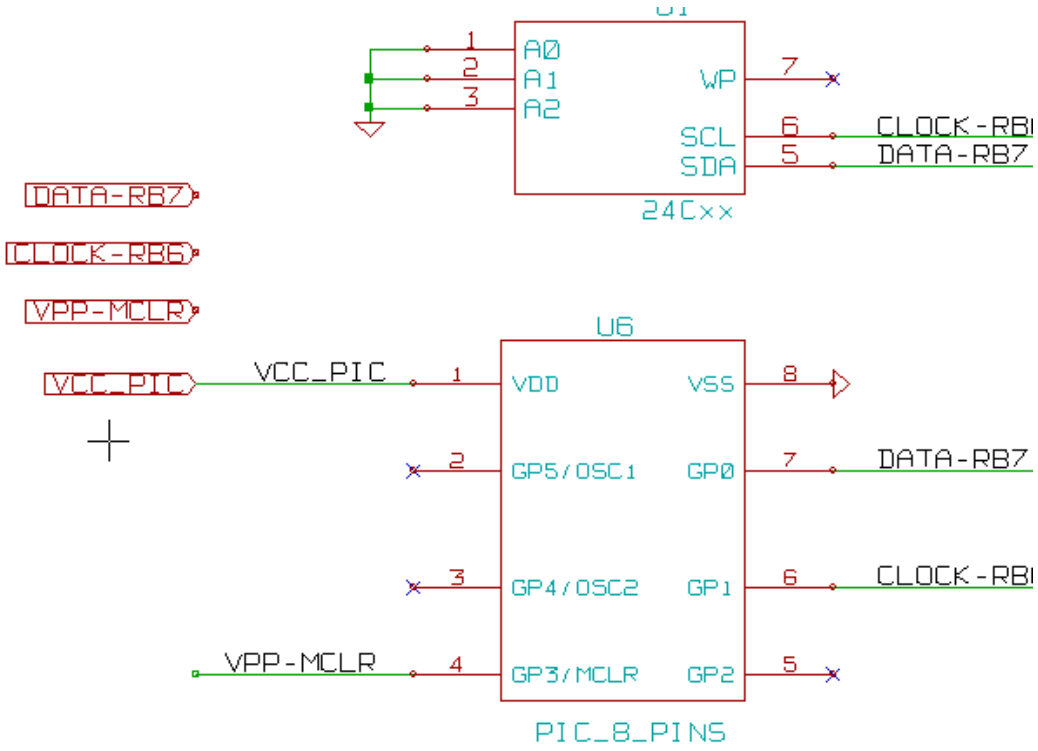


这是两页，由全局标签连接。

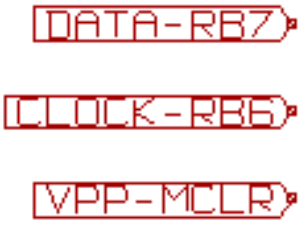
这是 pic_programmer.sch。



这是 pic_sockets.sch。


A0
A1
A2
WP
SCL
SDA
24Cxx
DATA-RB7
CLOCK-RB6
VPP-MCLR
VCC_PIC
U6
VDD
VSS
GP5/OSC1
GP0
GP4/OSC2
GP1
GP3/MCLR
GP2
PIC_8_PINS

查看全局标签。


DATA-RB7
CLOCK-RB6
VPP-MCLR

Chapter 8

符号注释工具

8.1 简介

注释工具允许您自动为原理图中的符号指定一个指示符。具有多个单位的符号的注释将分配一个唯一的后缀, 以最大限度地减少这些符号的数量。注释工具可通过图标图像访问: image:images/icons/annotate.png [icons_annotate_png]。在这里, 你可以找到它的主窗口。

可用的注释方案:

- 注释所有符号 (重置现有注释选项)
- 注释所有符号, 但不要交换任何以前批注的多单元元件。

- 仅注释当前未注释的符号。未批注的符号将具有以”? ”字符结尾的指示符。
- 注释整个层次结构 (使用整个原理图选项)。
- 仅注释当前工作表 (仅使用当前页选项)。

“重置, 但不交换任何带注释的多单元部件” 选项保留具有多单元的符号之间的所有现有关联。例如, U22 和 U2A 可能分别重新注释为 U1B 和 U1B, 但它们永远不会重新注释到 U1B 和 U2A, 也不会重新注释为 U2B 和 U2A。如果要确保保持引脚分组, 这很有用。

注释顺序选择提供了用于在层次结构的每个工作表内设置参考编号的方法。

除特定情况外, 如果您不想修改以前的批注, 则自动批注将应用于整个项目 (所有工作表) 和新元件。

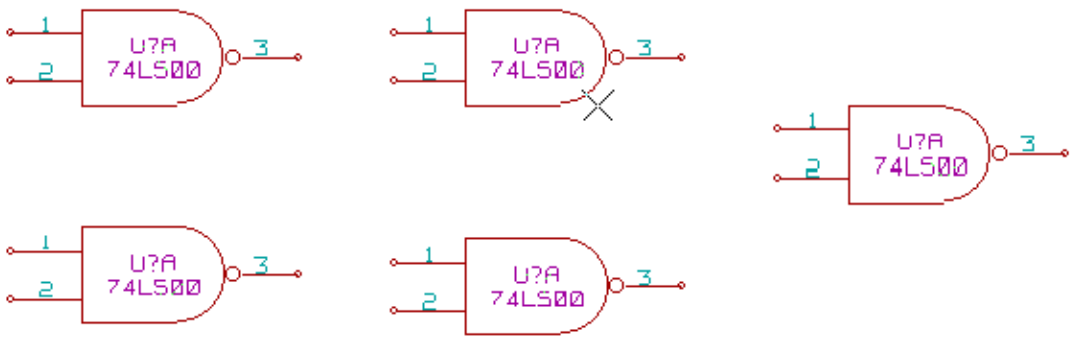
“注释选择” 给出了用于计算参考的方法:

- 在原理图中使用第一个空闲编号：元件从 1 开始注释（对于每个引用前缀）。如果存在先前的注释，则仅使用未使用的数字。
- 从纸张编号 *100 开始并使用第一个空闲编号：从纸张 1 的 101 开始注释，从纸张 2 的 201 开始，等等。如果在工作表内有超过 99 个具有相同参考前缀（U，R）的项目在图 1 中，注释工具使用数字 200 和更多，并且工作表 2 的注释将从下一个空闲编号开始。
- 从表格编号 *1000 开始并使用第一个空闲编号。对于纸张 1，注释从 1001 开始，从纸张 2 的 2001 开始。

8.2 一些例子

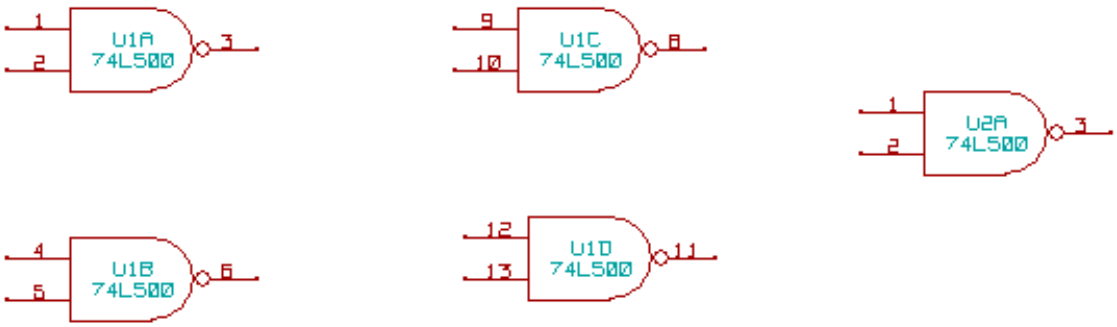
8.2.1 注释顺序

此示例显示放置了 5 个元素，但未注释。

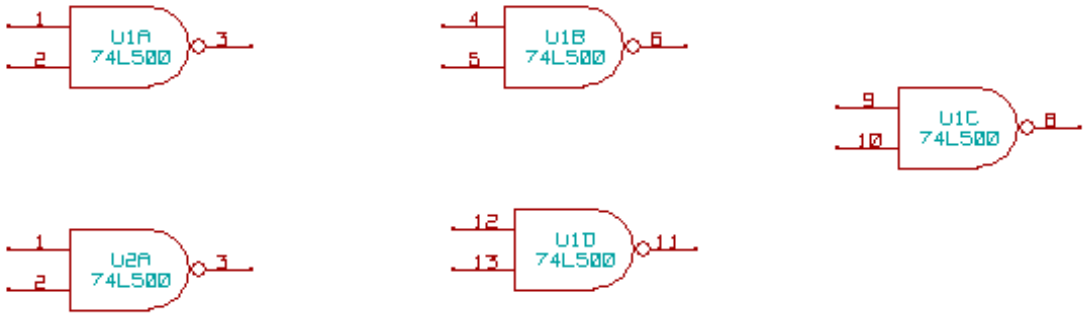


执行注释工具后，获得以下结果。

按 X 位置排序。



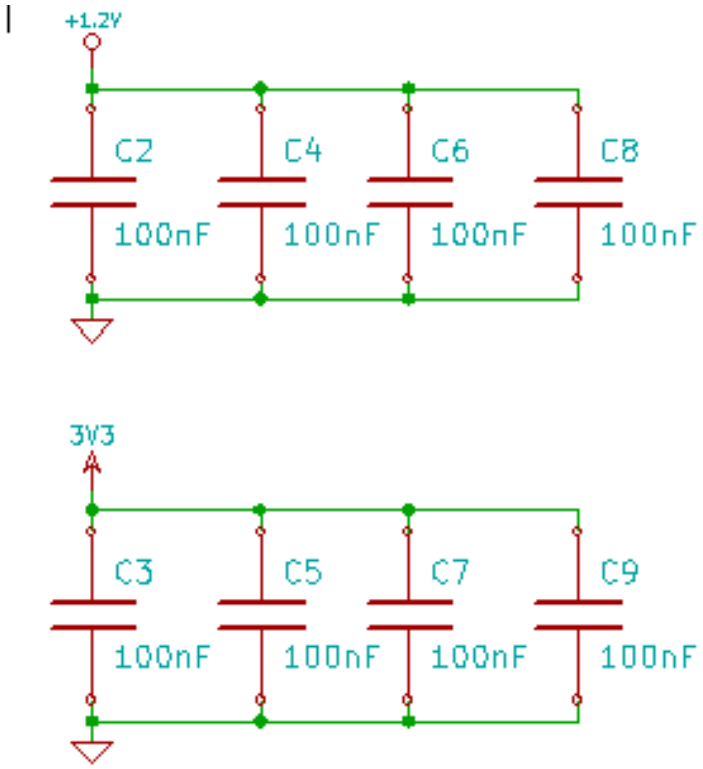
按 Y 位置排序。



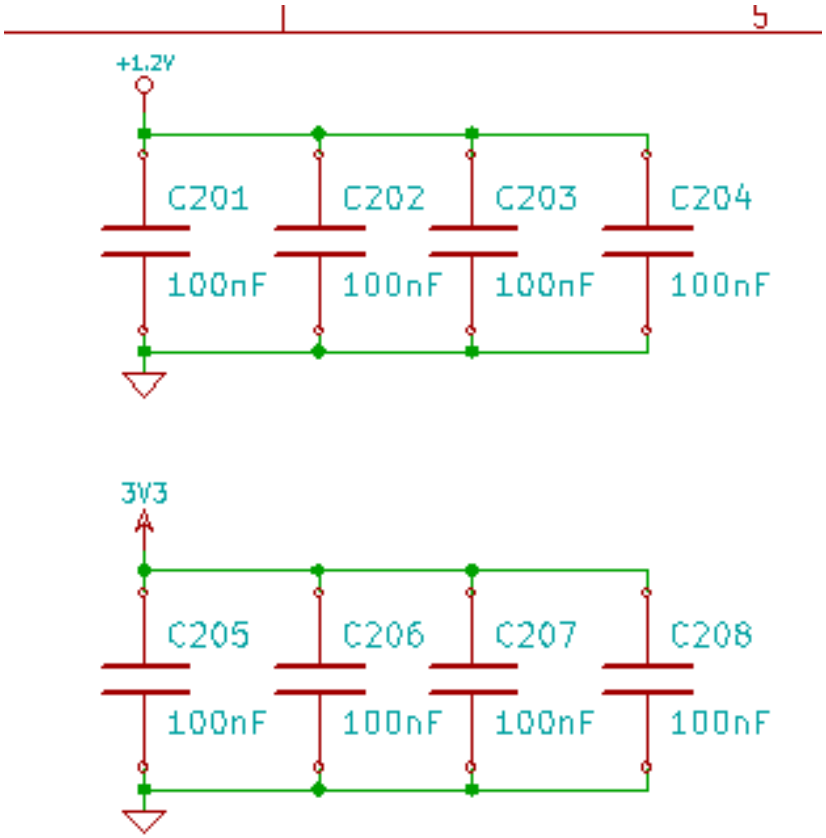
您可以看到四个 74LS00 门分布在 U1 包中，第五个 74LS00 已分配给下一个 U2。

8.2.2 注释选择

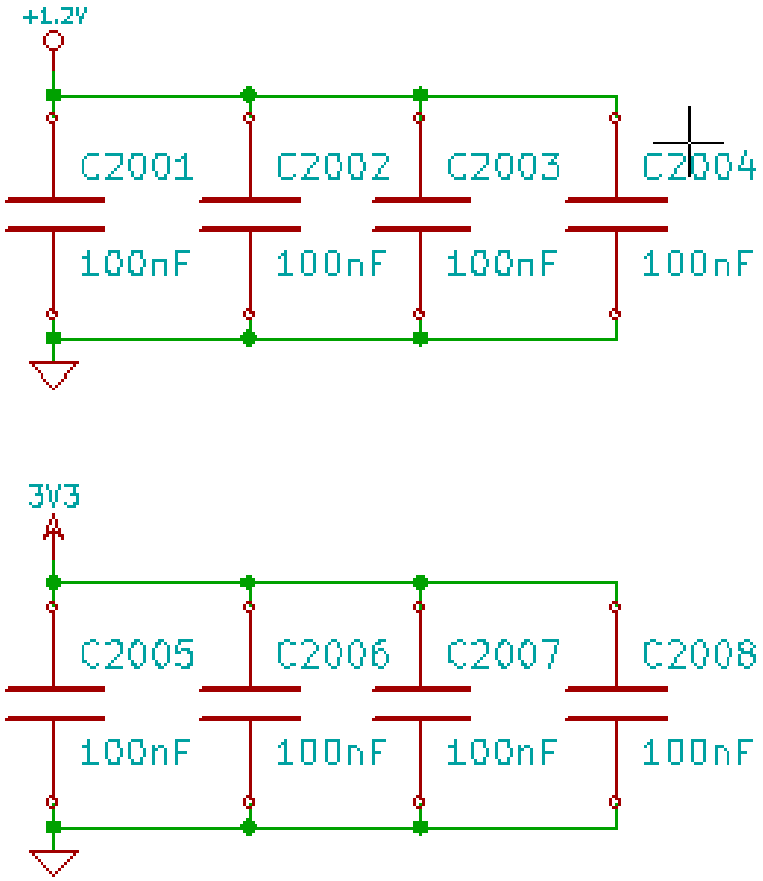
这是表 2 中的注释，其中选项使用原理图中的第一个空闲编号。



选项开始到工作表编号 *100 并使用第一个空闲编号给出以下结果。



选项开始到工作表编号 *1000 并使用第一个空闲编号给出以下结果。


+1.2V
C2001
C2002
C2003
C2004
100nF
100nF
100nF
100nF
3V3
C2005
C2006
C2007
C2008
100nF
100nF
100nF
100nF

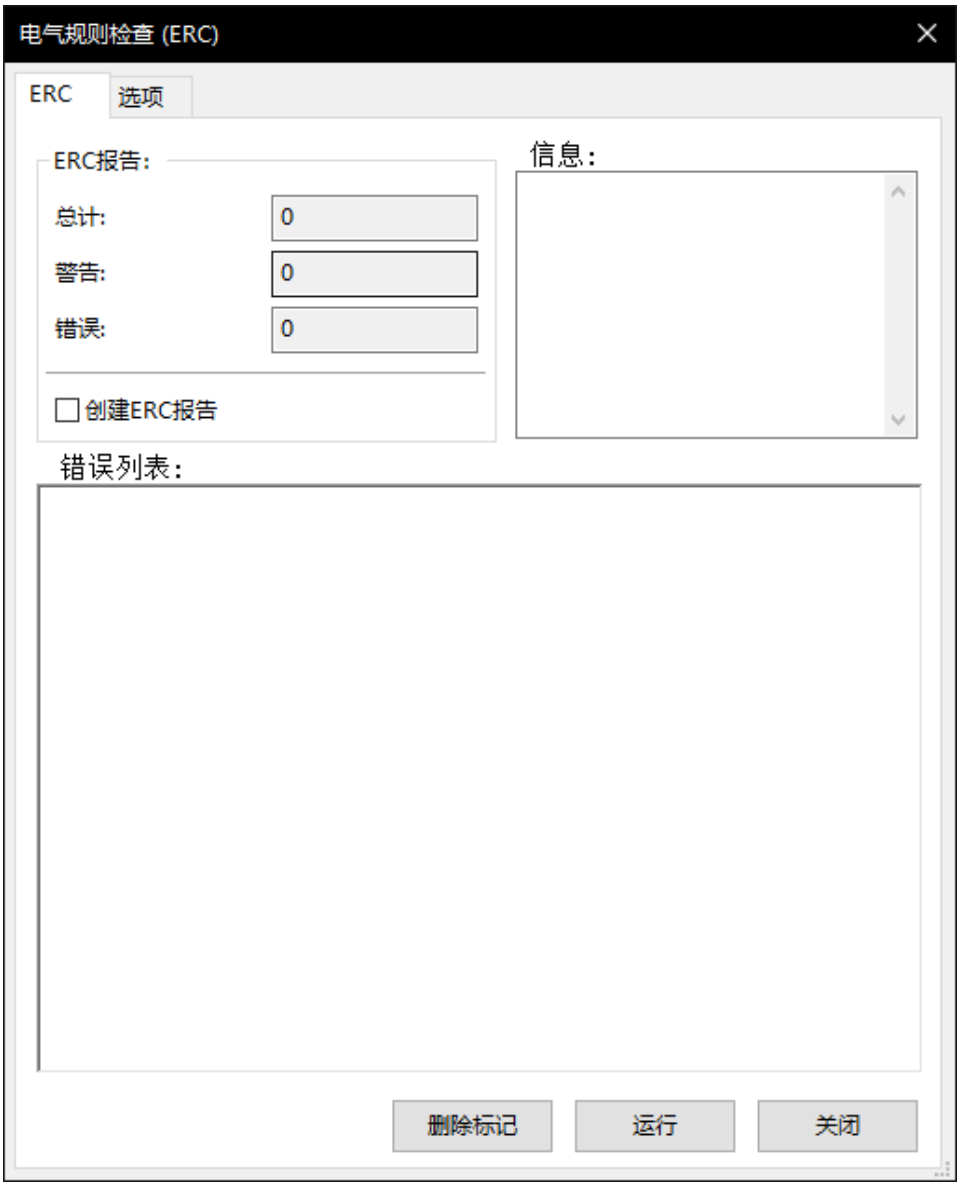
Chapter 9

使用电气规则检查进行设计验证

9.1 简介

电气规则检查（ERC）工具会自动检查原理图。ERC 检查工作表中的任何错误，例如未连接的引脚，未连接的分层符号，短路输出等。当然，自动检查不是绝对可靠的，并且可以检测所有设计错误的软件还不是 100％完成。这样的检查非常有用，因为它允许您检测许多疏忽和小错误。

实际上，必须检查所有检测到的错误，然后在正常进行之前进行纠正。ERC 的质量与在符号库创建期间声明电引脚属性时所采取的谨慎直接相关。ERC 输出报告为 错误或 警告。



9.2 如何使用 ERC

单击图标图像可以启动 ERC：images/icons/erc.png[ERC icon] 。

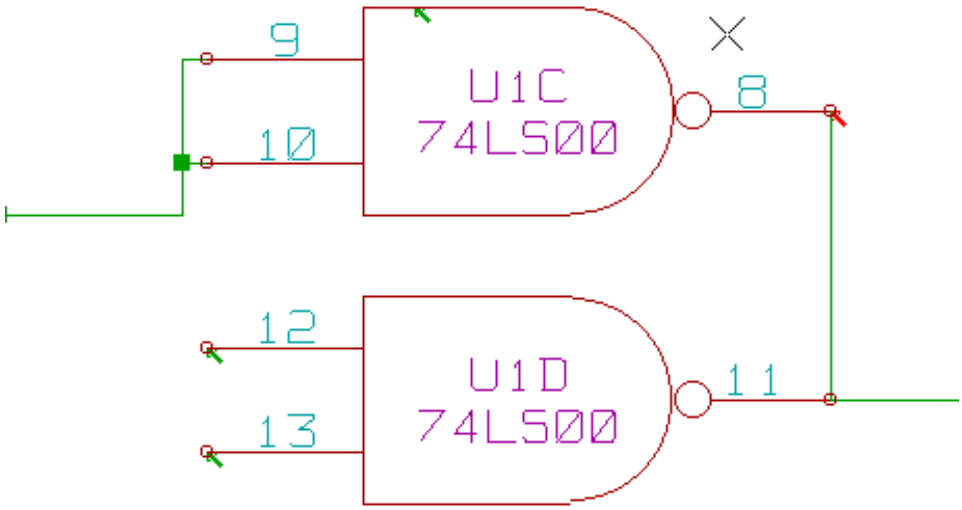
在原理图元素上设置警告, 以引发 ERC 错误 (引脚或标签)。

Note

- **在此对话框窗口中，单击错误消息时，您可以跳转到原理图中的相应标记。**
- **在原理图中，右键单击标记以访问相应的诊断消息。**

您还可以从对话框中删除错误标记。

9.3 ERC 的示例

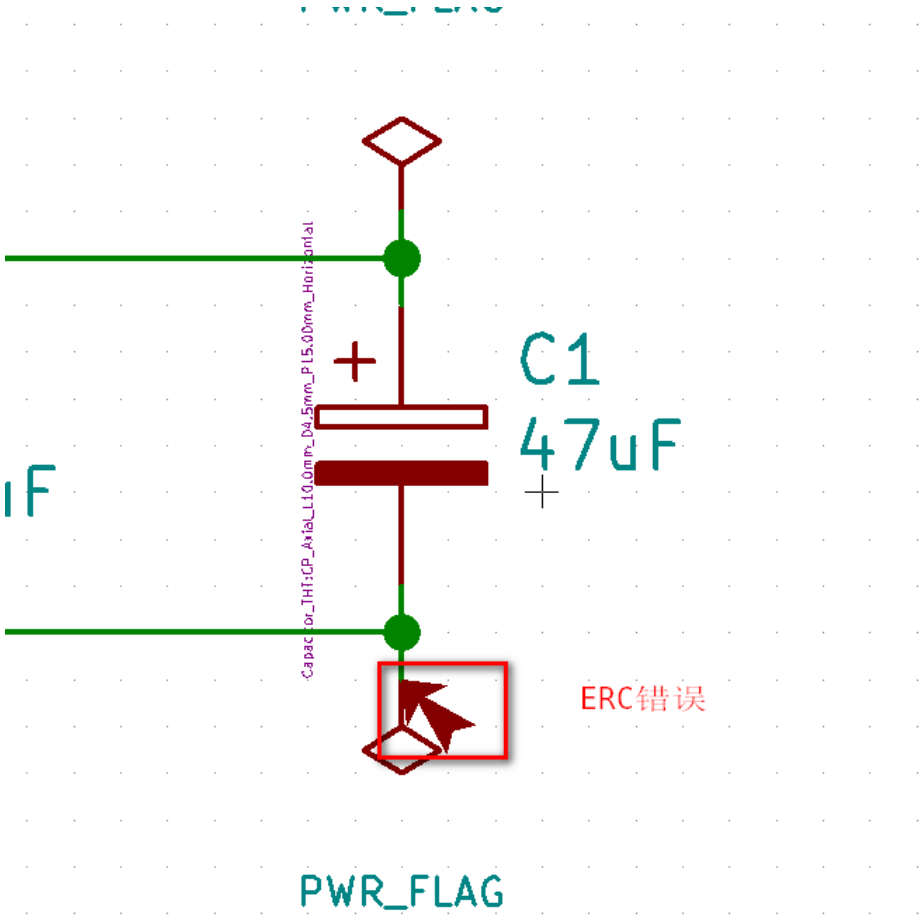


在这里, 您可以看到四个错误:

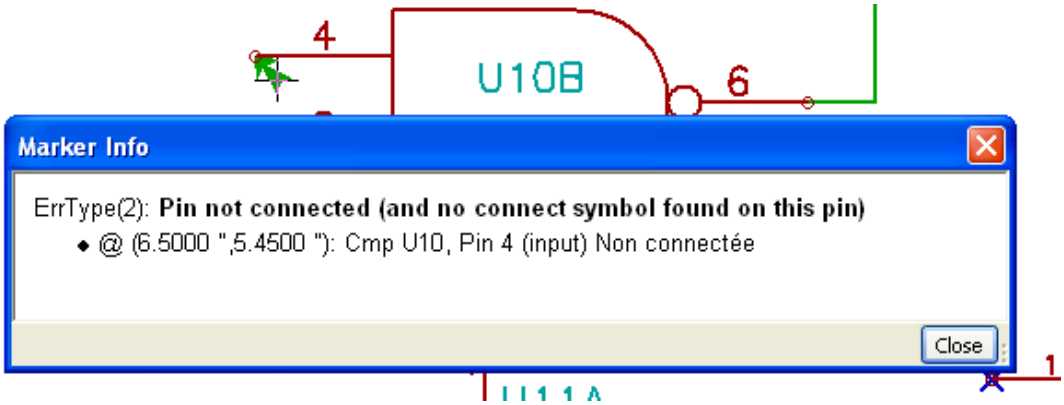
- 两个输出错误地连接在一起（红色箭头）。
- 两个输入未连接（绿色箭头）。
- 隐藏电源端口出现错误，缺少电源标志（顶部为绿色箭头）。

9.4 显示诊断

通过右键单击标记，弹出菜单允许您访问 ERC 标记诊断窗口。



当单击标记错误信息时，您可以获得错误的描述。

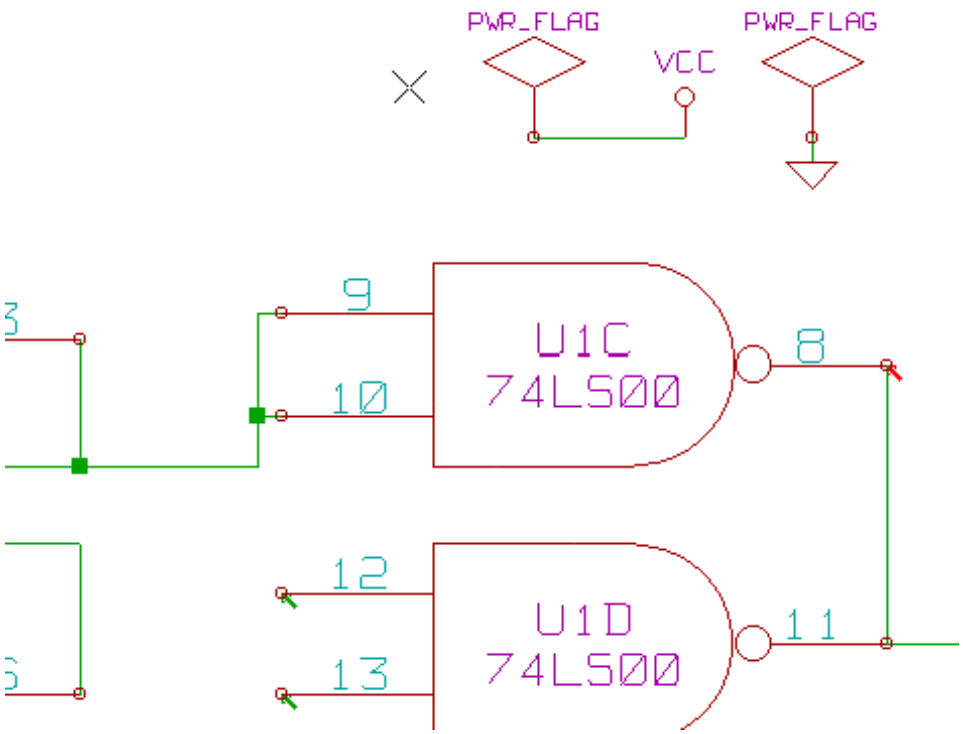


9.5 电源引脚和电源标志

通常在电源引脚上出现错误或警告，即使一切看起来都很正常。见上面的例子。之所以会发生这种情况，是因为在大多数设计中，电源是由不是电源的连接器提供的（如稳压器输出，它被声明为电源输出）。

因此，ERC 不会检测到任何电源输出引脚来控制该电线，并声明它们不是由电源驱动的。

要避免此警告，您必须在此类电源端口上放置 “PWR_FLAG”。看一下下面的例子：

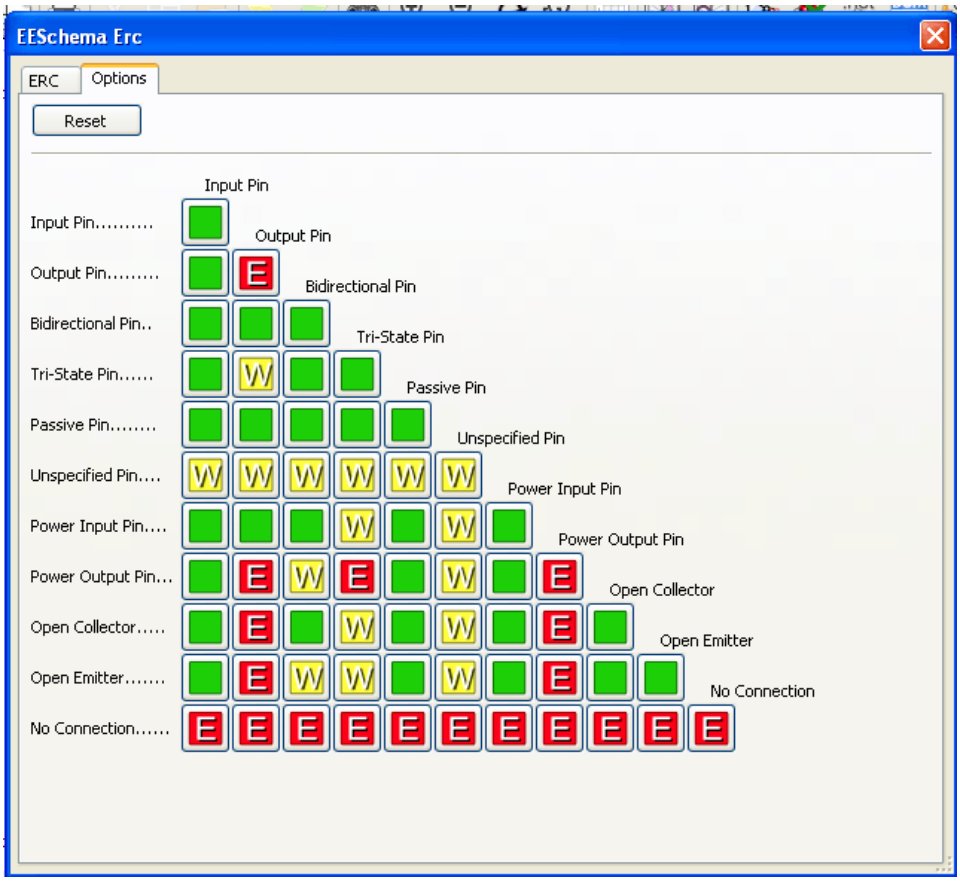


然后错误标记将消失。

大多数情况下，PWR_FLAG 必须连接到 GND，因为稳压器的输出声明为断电，但接地引脚永远不会断电（正常属性是电源输入），因此，在没有电源标志的情况下，接地不会与电源相连。

9.6 配置

选项面板允许您配置连接规则以定义错误和警告检查的电气条件。



单击所需的单元格方块可以更改规则，使其循环选择：正常，警告，错误。

9.7 ERC 报告文件

通过选中写入 ERC 报告选项，可以生成并保存 ERC 报告文件。ERC 报告文件的文件扩展名为.erc。以下是 ERC 报告文件的示例。

```
ERC control (4/1/1997-14:16:4)

***** Sheet 1 (INTERFACE UNIVERSAL)
ERC: Warning Pin input Unconnected @ 8.450, 2.350
ERC: Warning passive Pin Unconnected @ 8.450, 1.950
ERC: Warning: BiDir Pin connected to power Pin (Net 6) @ 10.100, 3.300
ERC: Warning: Power Pin connected to BiDir Pin (Net 6) @ 4.950, 1.400

>> Errors ERC: 4
```

Chapter 10

创建网络列表

10.1 概述

网表是描述符号之间的电连接的文件。这些连接称为网络。在网表文件中，您可以找到：

- 符号列表
- 符号之间的连接 (网) 列表。

存在许多不同的网表格式。有时，符号列表和网络列表是两个单独的文件。该网表是使用原理图捕获软件的基础，因为网表是与其他电子 CAD 软件的链接，例如：

- PCB 布局软件。
- 原理图和电信号模拟器。
- CPLD（和其他可编程 IC ）编译器。

Eeschema 支持几种网络列表格式。

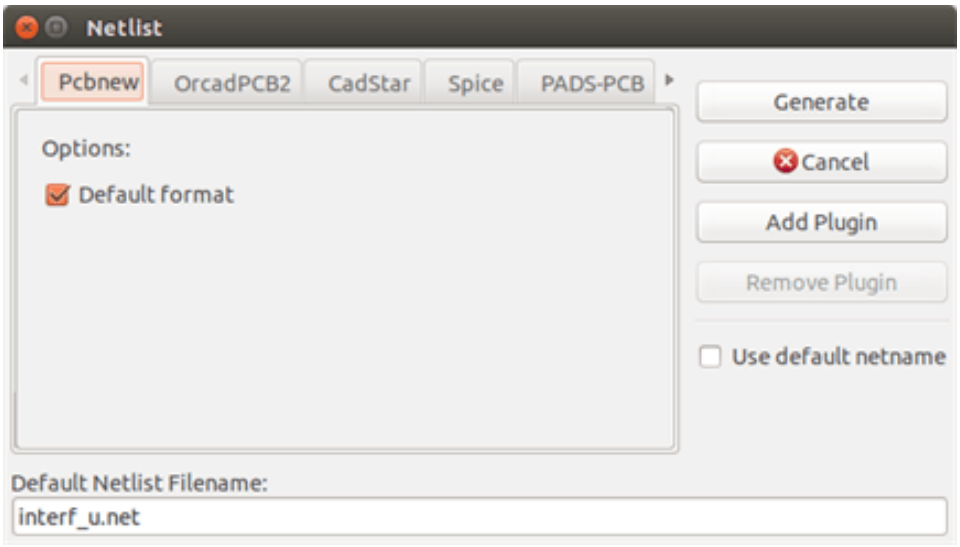
- PCBNEW 格式（印刷电路）。
- ORCAD PCB2 格式（印刷电路）。
- CADSTAR 格式（印刷电路）。
- Spice 格式，适用于各种模拟器（其他模拟器也使用 Spice 格式）。

10.2 网表格式

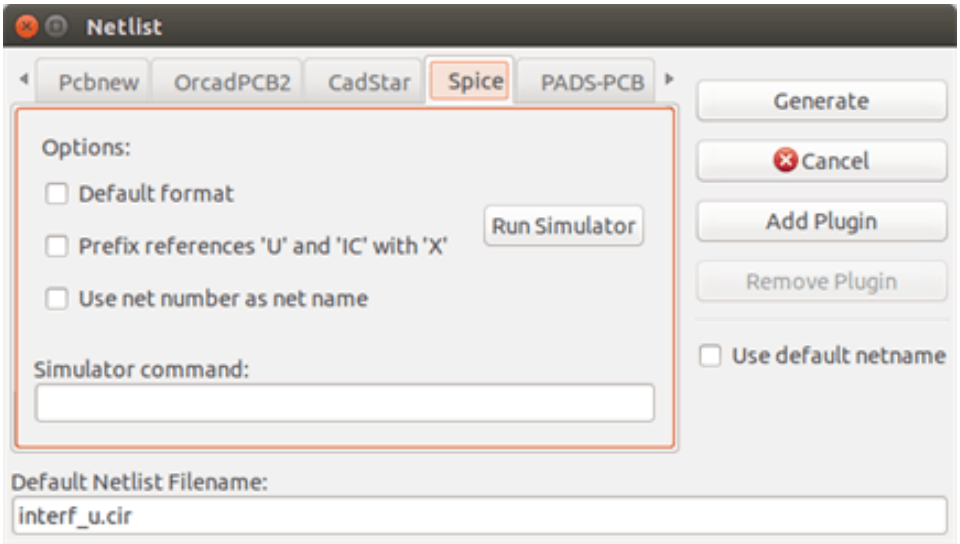
选择工具 

以打开网表创建对话框。

选择的 Pcbnew



选择的 Spice



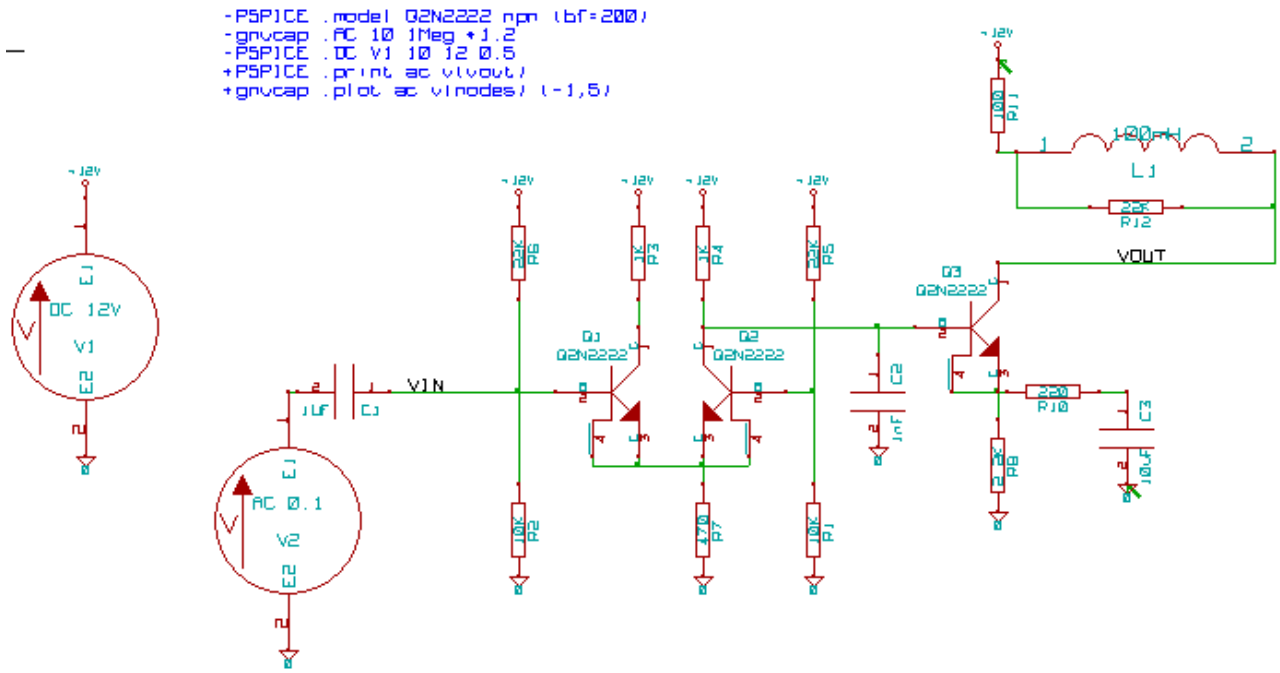
使用不同的选项卡，您可以选择所需的格式。在 Spice 格式中，您可以使用网络名称生成网表，这使得 SPICE 文件更易于阅读，或者使用旧版 Spice 使用的网络编号。通过单击“网表”按钮，将要求您提供网表文件名。

Note

对于大型原理图，网表生成最多可能需要几分钟时间。

10.3 网表示例

您可以在下面看到使用 PSPICE 库的原理图设计:



PCBNEW 网表文件示例:

```
# Eeschema Netlist Version 1.0 generee le 21/1/1997-16:51:15
(
(32E35B76 $noname C2 1NF {Lib=C}
(1 0)
(2 VOUT_1)
)
(32CFC454 $noname V2 AC_0.1 {Lib=VSOURCE}
(1 N-000003)
(2 0)
)
(32CFC413 $noname C1 1UF {Lib=C}
(1 INPUT_1)
(2 N-000003)
)
(32CFC337 $noname V1 DC_12V {Lib=VSOURCE}
(1 +12V)
(2 0)
)
(32CFC293 $noname R2 10K {Lib=R}
(1 INPUT_1)
(2 0)
)
(32CFC288 $noname R6 22K {Lib=R}
(1 +12V)
(2 INPUT_1)
)
(32CFC27F $noname R5 22K {Lib=R}
```

```
(1 +12V)
(2 N-000008)
)
(32CFC277 $noname R1 10K {Lib=R}
(1 N-000008)
(2 0)
)
(32CFC25A $noname R7 470 {Lib=R}
(1 EMET_1)
(2 0)
)
(32CFC254 $noname R4 1K {Lib=R}
(1 +12V)
(2 VOUT_1)
)
(32CFC24C $noname R3 1K {Lib=R}
(1 +12V)
(2 N-000006)
)
(32CFC230 $noname Q2 Q2N2222 {Lib=NPN}
(1 VOUT_1)
(2 N-000008)
(3 EMET_1)
)
(32CFC227 $noname Q1 Q2N2222 {Lib=NPN}
(1 N-000006)
(2 INPUT_1)
(3 EMET_1)
)
)
# End
```

在 PSPICE 格式中，网表如下：

```
* Eeschema 网表 1.1 版(Spice 格式)创建日期: 18/6/2008-08:38:03

.model Q2N2222 npn (bf=200)
.AC 10 1Meg \*1.2
.DC V1 10 12 0.5



R12   /VOUT N-000003 22K
R11   +12V N-000003 100
L1   N-000003 /VOUT 100mH
R10   N-000005 N-000004 220
C3   N-000005 0 10uF
C2   N-000009 0 1nF
R8   N-000004 0 2.2K
```

```
Q3   /VOUT N-000009 N-000004 N-000004 Q2N2222
V2   N-000008 0 AC 0.1
C1   /VIN N-000008 1UF
V1   +12V 0 DC 12V
R2   /VIN 0 10K
R6   +12V /VIN 22K
R5   +12V N-000012 22K
R1   N-000012 0 10K
R7   N-000007 0 470
R4   +12V N-000009 1K
R3   +12V N-000010 1K
Q2   N-000009 N-000012 N-000007 N-000007 Q2N2222
Q1   N-000010 /VIN N-000007 N-000007 Q2N2222

.print ac v(vout)
.plot ac v(nodes) (-1,5)

.end
```

10.4 关于网表的说明

10.4.1 网表名称注意事项

许多使用网表的软件工具不接受元件名称、引脚、网络或其他信息中的空格。避免在标签中使用空格, 或元件或其引脚的名称和值字段, 以确保最大程度的兼容性。

同样，字母和数字以外的特殊字符也可能导致问题。请注意，此限制与 Eeschema 无关，而是与网表格式无关，后者可以变为不可翻译为使用网表文件的软件。

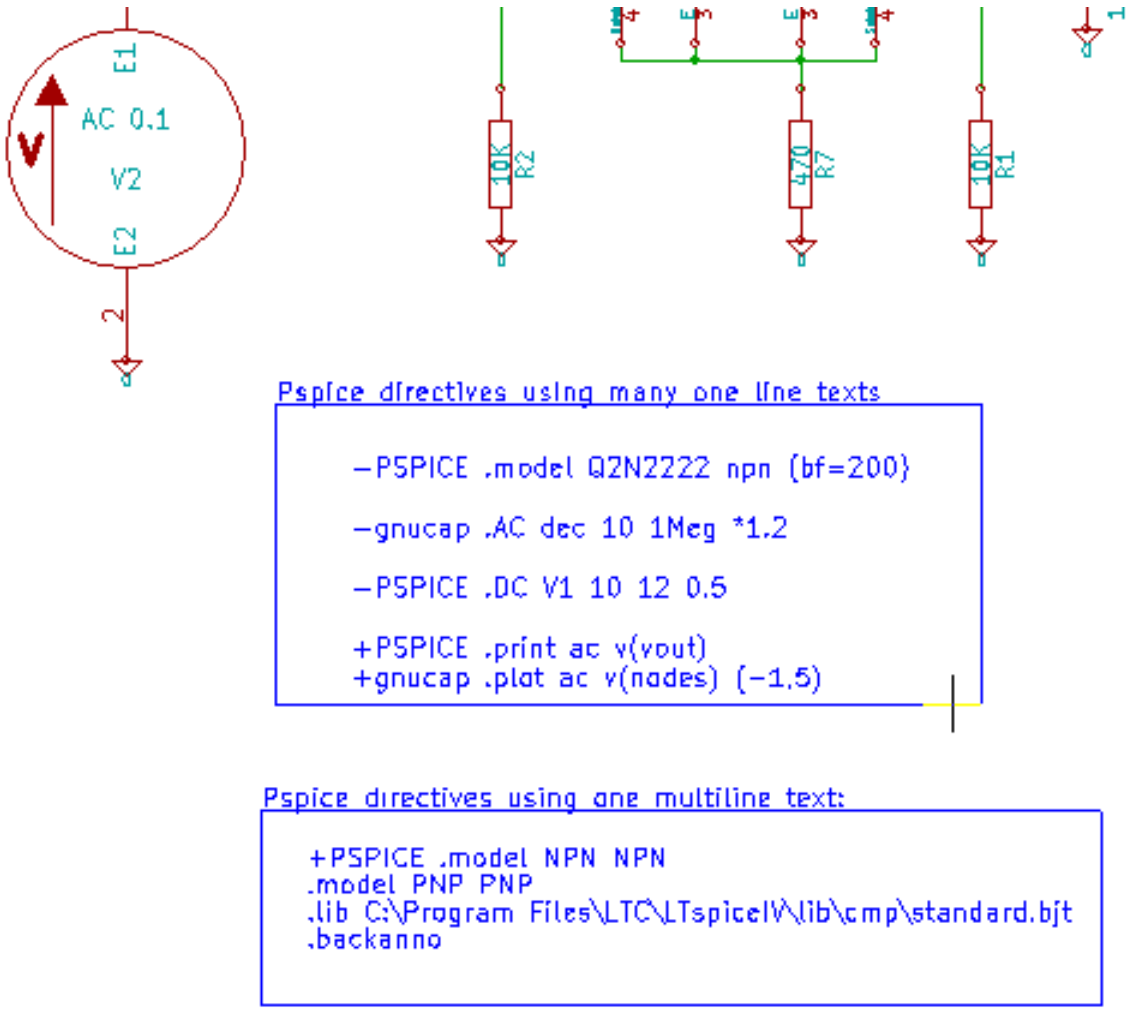
10.4.2 PSPICE 网表

对于 Pspice 模拟器，您必须在网表本身（.PROBE，.AC 等）中包含一些命令行。

从关键字 **-pspice** 或 **-gnucap** 开始的示意图中包含的任何文本行都将在网表的顶部插入（不带关键字）。

以关键字 **+pspice** 或 **+gnucap** 开头的示意图中包含的任何文本行都将在网表的末尾插入（不带关键字）。

下面是一个使用许多单行文本和一个多行文本的示例:



例如，如果您键入以下文本（不要使用标签!）：

```
-PSPICE .PROBE
```

一行.PROBE 将被插入网表中。

在前面的例子中，在网表的开头插入了三行，用这种技术插入了两行。

如果您使用的是多行文本，则只需要 **+pspice** 或 **+gnucap** 关键字一次：

```
+PSPICE .model NPN NPN
.model PNP PNP
.lib C:\Program Files\LTC\LTspiceIV\lib\cmp\standard.bjt
.backanno
```

创建四行:

```
.model NPN NPN
.model PNP PNP
.lib C:\Program Files\LTC\LTspiceIV\lib\cmp\standard.bjt
.backanno
```

另请注意，Pspice 的 GND 网络必须命名为 0（零）。

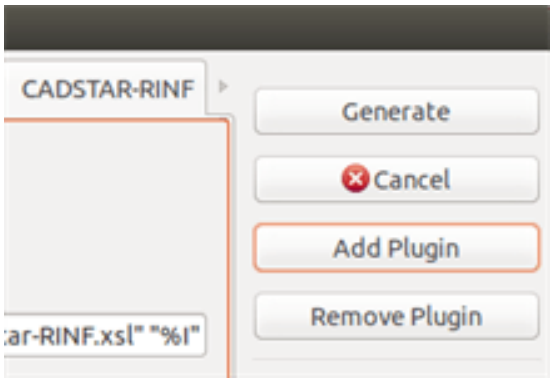
10.5 其他格式

对于其他网表格式，您可以以插件的形式添加网表转换器。这些转换器由 Eeschema 自动启动。第 14 章给出了转换器的一些解释和示例。

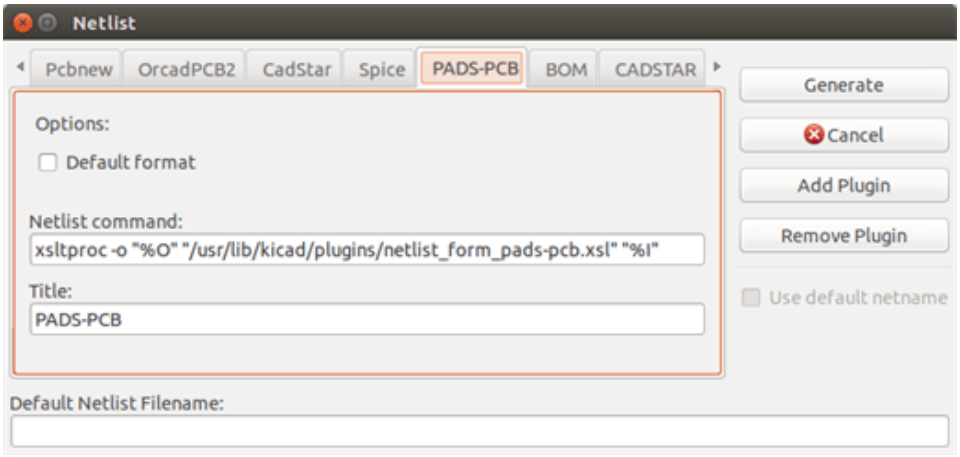
转换器是一个文本文件（xsl 格式），但可以使用其他语言，如 Python。使用 xsl 格式时，工具（xsltproc.exe 或 xsltproc）读取 Eeschema 创建的中间文件和转换器文件以创建输出文件。在这种情况下，转换器文件（工作表样式）非常小并且非常容易编写。

10.5.1 在对话框窗口中

您可以通过 添加插件按钮添加新的网表插件。



这是插件 PadsPcb 设置窗口：



设置将需要：

- 标题（例如，网表格式的名称）。
- 要启动的插件。

生成网表时：

1. Eeschema 创建一个中间文件 *.tmp，例如 test.tmp。
2. Eeschema 运行插件，读取 test.tmp 并创建 test.net。

10.5.2 命令行格式

下面是一个示例，使用 xsltproc.exe 作为转换.xsl 文件的工具，使用文件 netlist_form_pads-pcb.xsl 作为转换表样式：

f:/kicad/bin/xsltproc.exe -o %O.net f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl %I

附：

f:/kicad/bin/xsltproc.exe	A tool to read and convert xsl file
-o %O.net	Output file: %O will define the output file.
f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl	File name converter (a sheet style, xsl format).
%I	Will be replaced by the intermediate file created by Eeschema (*.tmp).

对于名为 test.sch 的原理图，实际的命令行是：

f:/kicad/bin/xsltproc.exe -o test.net f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl test.tmp.

10.5.3 转换器和工作表样式 (插件)

这是一个非常简单的软件，因为它的目的只是将输入文本文件（中间文本文件）转换为另一个文本文件。此外，从中间文本文件中，您可以创建 BOM 清单。

使用 xsltproc 作为转换器工具时，仅生成工作表样式。

10.5.4 中间网表文件格式

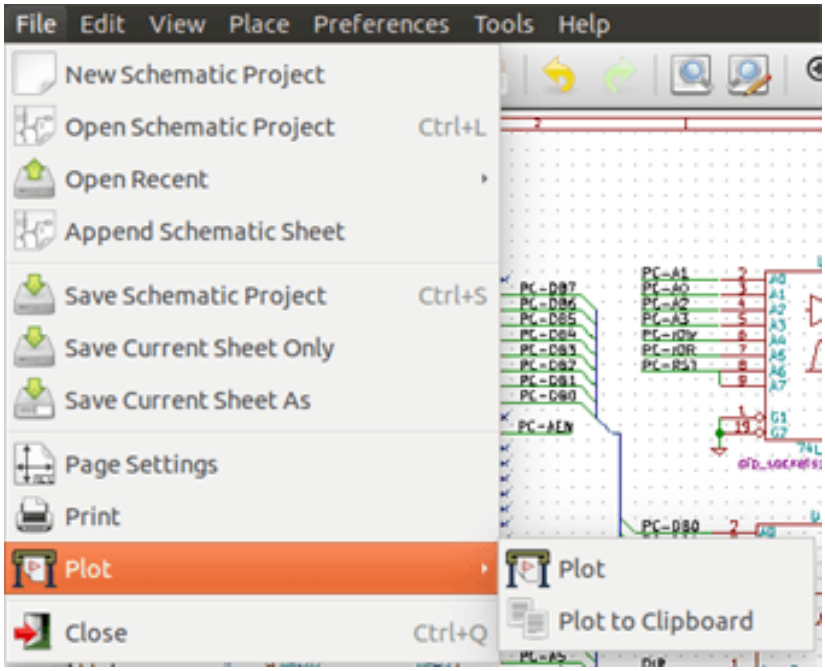
有关 xslproc 的更多说明，中间文件格式的说明以及转换器的工作表样式的一些示例，请参见第 14 章。

Chapter 11

绘图和打印

11.1 简介

您可以通过文件菜单访问打印和绘图命令。



支持的输出格式是 Postscript，PDF，SVG，DXF 和 HPGL。您也可以直接打印到您的打印机。

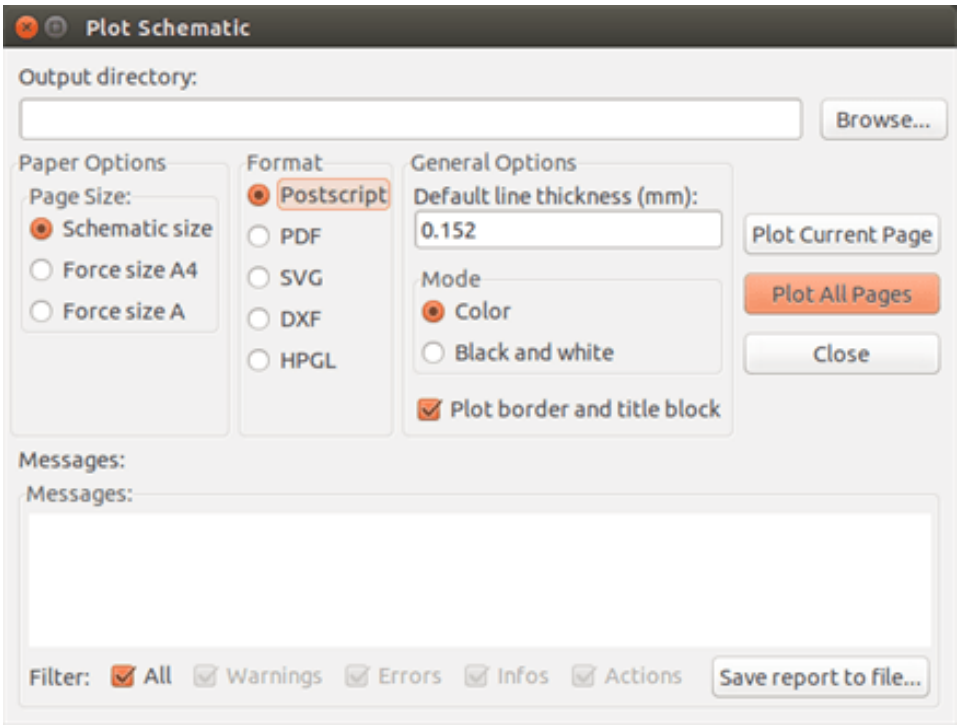
11.2 常见的打印命令

绘制当前页面
仅打印当前工作表的一个文件。

绘制所有页面
允许您绘制整个层次结构（为每个工作表生成一个打印文件）。

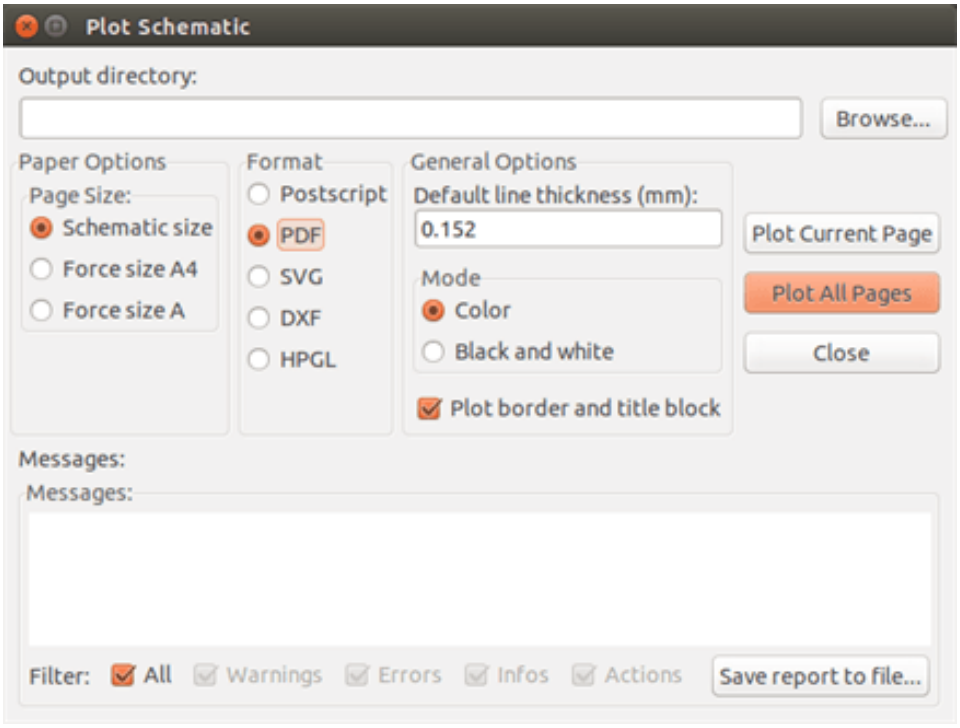
11.3 在 Postscript 中绘制

此命令允许您创建 PostScript 文件。



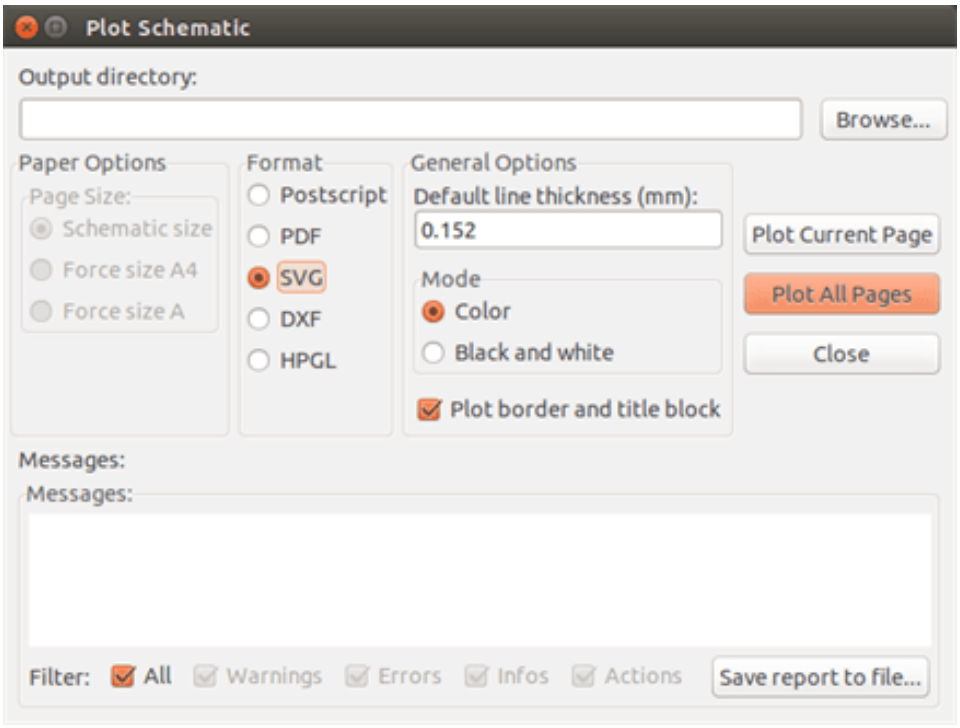
文件名是扩展名为.ps 的工作表名称。您可以禁用”绘制边框和标题栏“选项。如果要创建用于封装的 postscript 文件（格式.eps），这通常用于在文字处理软件中插入图表，这非常有用。消息窗口显示创建的文件名。

11.4 以 PDF 格式绘制



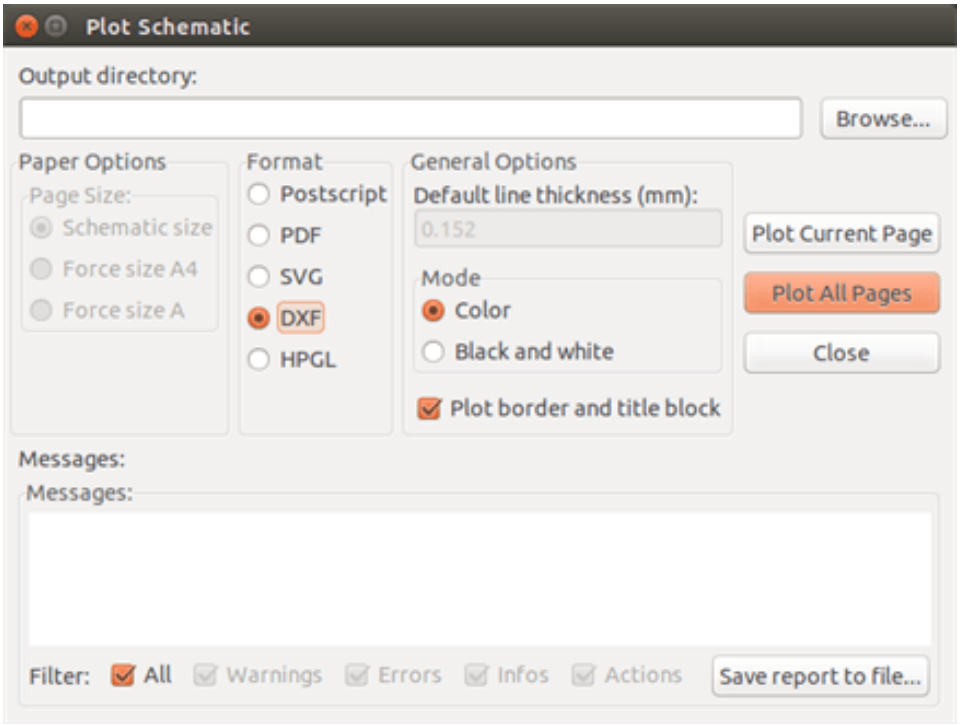
允许您使用 PDF 格式创建打印文件。文件名是扩展名为.pdf 的工作表名称。

11.5 在 SVG 中绘图



允许您使用矢量格式 SVG 创建打印文件。文件名是扩展名为.svg 的工作表名称。

11.6 在 DXF 中绘图



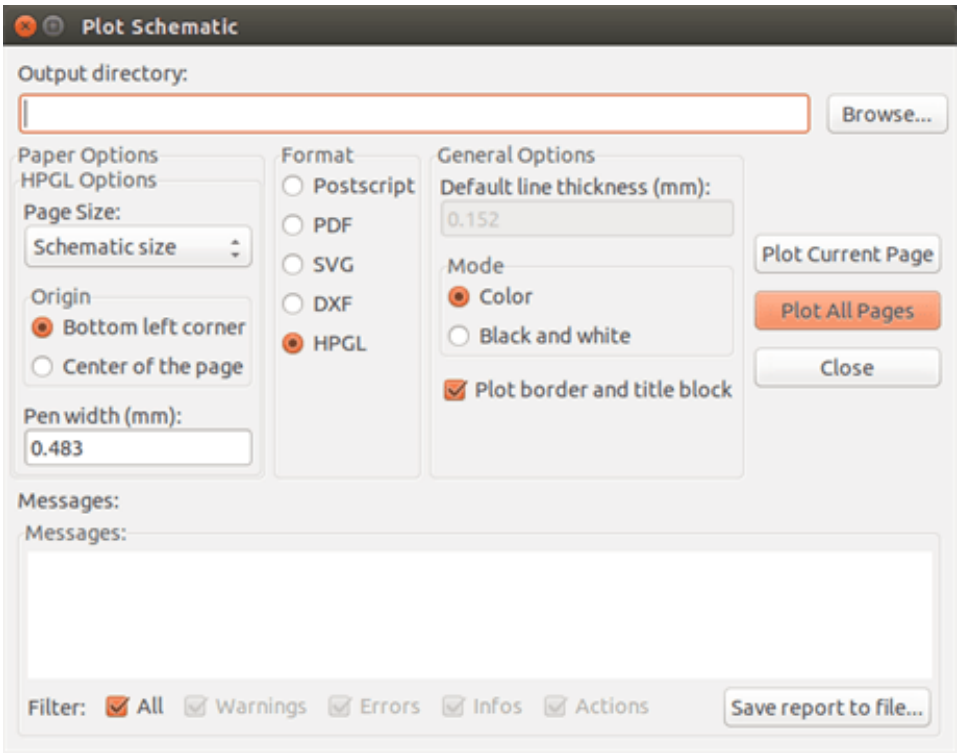
允许您使用 DXF 格式创建打印文件。文件名是扩展名为.dxf 的工作表名称。

11.7 在 HPGL 中绘图

此命令允许您创建 HPGL 文件。在这种格式中，您可以定义：

- 页面大小。
- 原点。
- 笔宽（mm）。

绘图仪设置对话框窗口如下所示：



输出文件名将是工作表名称加上扩展名.plt。

11.7.1 纸张尺寸选择

通常检查纸张尺寸。在这种情况下，将使用标题栏菜单中定义的纸张尺寸，并且所选的比例将为 1. 如果选择了不同的纸张尺寸（A4 为 A0，或 A 为 E），则自动调整比例以填充页面。

11.7.2 偏移调整

对于所有标准尺寸，您可以调整偏移以尽可能准确地使图形居中。由于绘图仪在工作表的中心或左下角有原点，因此必须能够引入偏移以便正确绘图。

一般来说：

- 对于原点位于纸张中心的绘图仪，偏移量必须为负值并设置为纸张尺寸的一半。
- 对于原点位于纸张左下角的绘图仪，偏移量必须设置为 0。

要设置偏移量：

- 选择纸张尺寸。
- 设置偏移量 X 和偏移量 Y.
- 单击接受偏移量。

11.8 在纸上打印

此命令可通过图标  获得，允许您可视化并生成标准打印机的设计文件。

“打印表格参考和标题栏”选项可启用或禁用图纸参考和标题栏。

“黑白打印”选项设置单色打印。如果使用黑白激光打印机，通常需要此选项，因为颜色打印成半色调，通常不太可读。

Chapter 12

符号库编辑器

12.1 关于符号库的一般信息

符号是一个示意图元素，包含图形表示，电气连接和定义符号的字段。原理图中使用的符号存储在符号库中。Eeschema 提供了一个符号库编辑工具，允许您在库之间创建库，添加，删除或传输符号，将符号导出到文件以及从文件导入符号。库编辑工具提供了一种管理符号库文件的简单方法。

12.2 符号库概述

符号库由一个或多个符号组成。通常，符号按功能，类型和/或制造商进行逻辑分组。

符号由以下部分组成：

- 提供符号定义的图形项（线，圆，圆弧，文本等）。
- 具有图形属性（线，时钟，反转，低电平有效等）和电气规则检查（ERC）工具使用的电气属性（输入，输出，双向等）的引脚。
- 诸如参考，值，PCB 设计的相应封装名称等字段等。
- 别名用于将常用符号（如 7400）与其所有衍生物（如 74LS00,74HC00 和 7437）相关联。所有这些别名共享相同的库符号。

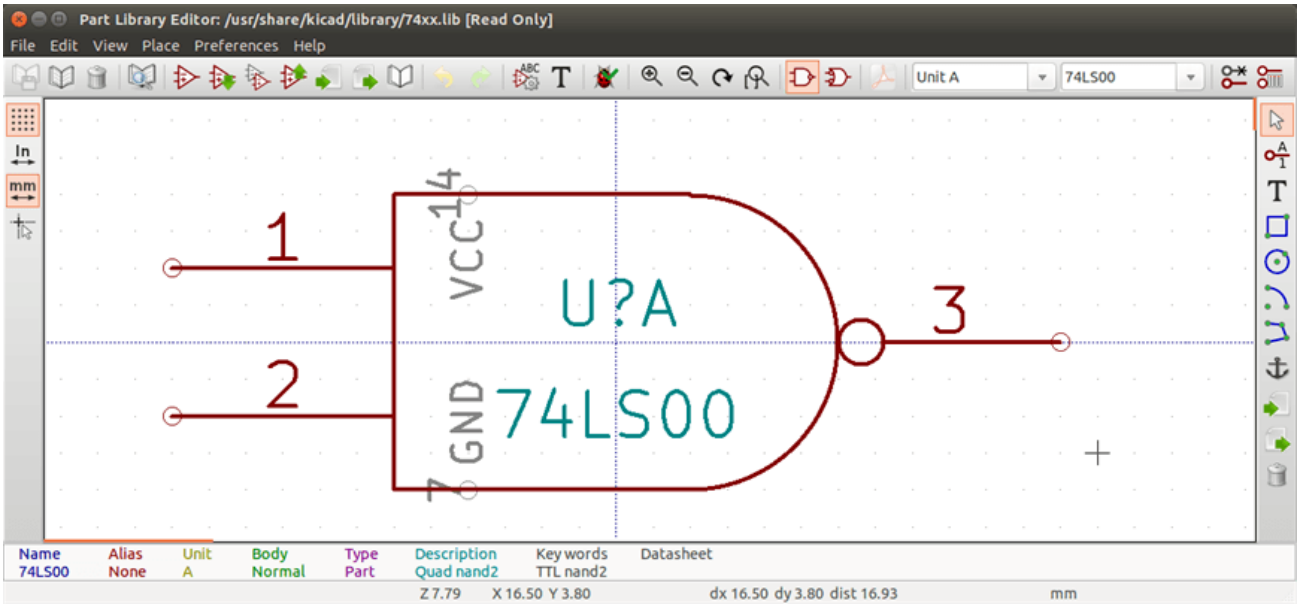
正确的符号设计需要：

- 定义符号是否由一个或多个单元组成。
- 定义符号是否具有替代的体型，也称为 De Morgan 表示。
- 使用线条，矩形，圆形，多边形和文本设计其符号表示。
- 通过仔细定义每个引脚的图形元素，名称，编号和电气属性（输入，输出，三态，电源端口等）来添加引脚。
- 如果其他符号具有相同的设计，则添加别名，如果已从其他符号创建符号，则将其固定或删除。

- 添加可选字段，例如 PCB 设计软件使用的封装名称和/或定义其可见性。
- 通过添加描述字符串和数据表链接等来记录符号。
- 将其保存在所需的库中。

12.3 符号库编辑器概述

符号库编辑器主窗口如下所示。它由三个工具栏组成，可快速访问常用功能和符号查看/编辑区域。并非所有命令都可在工具栏上使用，但可以使用菜单访问。



12.3.1 主工具栏

通常位于主窗口顶部的主工具栏包括库管理工具，撤消/重做命令，缩放命令和符号属性对话框。

	保存当前选定的库。如果没有，该按钮将被禁用库当前已选中或当前未选择更改库已经制作完成。
	选择要编辑的库。
	从当前选定的库或任何库中删除符号如果当前没有选择库，则由项目定义。
	打开符号库浏览器以选择库和符号编辑。
	创建一个新符号。
	从当前选定的库中加载符号以进行编辑。

	从当前加载的符号创建新符号。
	将当前符号更改保存在内存中。库文件不是改变。
	从文件中导入一个符号。
	将当前符号导出到文件。
	创建包含当前符号的新库文件。注意：新的库不会自动添加到项目中。
	撤消上次编辑。
	重做最后一次撤消。
	编辑当前符号属性。
	编辑当前符号的字段。
	测试当前符号是否存在设计错误。
	放大。
	缩小。
	刷新显示。
	缩放以适合显示中的符号。
	选择正常的体型。如果当前，该按钮被禁用符号没有替代的体型风格。
	选择备用体型。如果当前，该按钮被禁用符号没有替代的体型风格。
	显示相关文档。如果没有，该按钮将被禁用文档是为当前符号定义的。
	选择要显示的单位。如果，则将禁用下拉控件当前符号不是从多个单元派生的。
	选择别名。如果是，则下拉控件将 被禁用当前符号没有任何别名。
	引脚编辑：针对引脚形状和位置的独立编辑具有多个单位和替代符号的符号。
	显示引脚表。

12.3.2 元素工具栏

垂直工具栏通常位于主窗口的右侧，允许您放置设计符号所需的所有元素。下表定义了每个工具栏按钮。

	选择工具。使用选择工具右键单击可打开上下文菜单对于光标下的对象。使用选择工具单击鼠标左键在消息中显示光标下对象的属性主窗口底部的面板。双击左键选择工具将打开该对象下的属性对话框光标。

	引脚工具。单击鼠标左键以添加新引脚。
	图形文本工具。单击鼠标左键以添加新的图形文本项。
	矩形工具。单击鼠标左键以开始绘制 a 的第一个角图形矩形。再次单击鼠标左键以放置对角矩形。
	圆形工具。单击鼠标左键以开始绘制新的图形圆圈中心。再次单击鼠标左键以定义圆的半径。
	弧工具。单击鼠标左键以开始从中绘制新的图形弧项中央。再次单击鼠标左键以定义第一个圆弧终点。左键点击再次定义第二个弧终点。
	多边形工具。单击鼠标左键以开始绘制新的图形多边形项在当前的符号。左键单击每个加法多边形线。双击左键以完成多边形。
	锚点工具。单击鼠标左键以设置符号的锚点位置。
	从文件导入符号。
	将当前符号导出到文件。
	删除工具。单击鼠标左键以从当前符号中删除对象。

12.3.3 选项工具栏

垂直工具栏通常位于主窗口的左侧，允许您设置一些编辑器绘图选项。下表定义了每个工具栏按钮。

	打开和关闭网格可见性。
	将单位设置为英寸。
	将单位设置为毫米。
	打开和关闭全屏光标。

12.4 库选择与维护

可以通过  选择当前库，它会显示所有可用库并允许您选择一个库。加载或保存符号时，它将被放入此库中。符号的库名称是其 value 字段的内容。

Note

- **您必须将库加载到 Eeschema 中才能访问其内容。**
- **通过单击主工具栏上的**  **，可以在修改后保存当前库的内容。**
- **通过单击图像可以从任何库中删除符号：**  **。**

12.4.1 选择并保存符号

当您编辑符号时，您实际上并不是在其库中的符号上工作，而是在计算机内存中的符号上复制它。任何编辑操作都可以轻松撤消。可以从本地库或现有符号加载符号。

12.4.1.1 符号选择

单击主工具栏上的  将显示可从当前所选库中选择和加载的可用符号列表。

Note

如果通过别名选择了符号，则加载的符号的名称将显示在窗口标题栏上，而不是显示在选定的别名上。符号别名列表始终随每个符号一起加载，并且可以进行编辑。您可以通过从图像中选择当前符号的别名来创建新符号： 

。别名列表中的第一项是符号的根名称。

Note

或者，单击  **可以加载先前由图像保存的符号：**  **。**

12.4.1.2 保存符号

修改后，符号可以保存在当前库中，新库中，也可以导出到备份文件中。

要将修改后的符号保存在当前库中，请单击  。请注意，更新命令仅将符号更改保存在本地内存中。这样，您可以在恢复库之前做出决定。

要将符号更改永久保存到库文件，请单击  ，它将覆盖现有库文件并更改符号。

如果要创建包含当前符号的新库，请单击  。系统将要求您输入新的库名称。

Note

新库不会自动添加到当前项目中。
您必须使用《manage-sym-lib-table，Symbol Library Table 对话框》将您希望在原理图中使用的任何新库添加到 Eeschema 中的项目库列表中。

单击  以创建仅包含当前符号的文件。该文件将是一个标准库文件，它只包含一个符号。此文件可用于将符号导入另一个库。实际上，创建新的库命令和导出命令基本相同。

12.4.1.3 将符号转移到另一个库

您可以使用以下命令很容易地将符号从源库复制到目标库中:

- 单击图像选择源库:  。
- 通过单击图像加载要传输的符号:  。符号将显示在编辑区域中。
- 单击图像选择目标库:  。
- 单击图像将当前符号保存到本地内存中的新库:  。
- 单击图像将符号保存在当前本地库文件中:  。

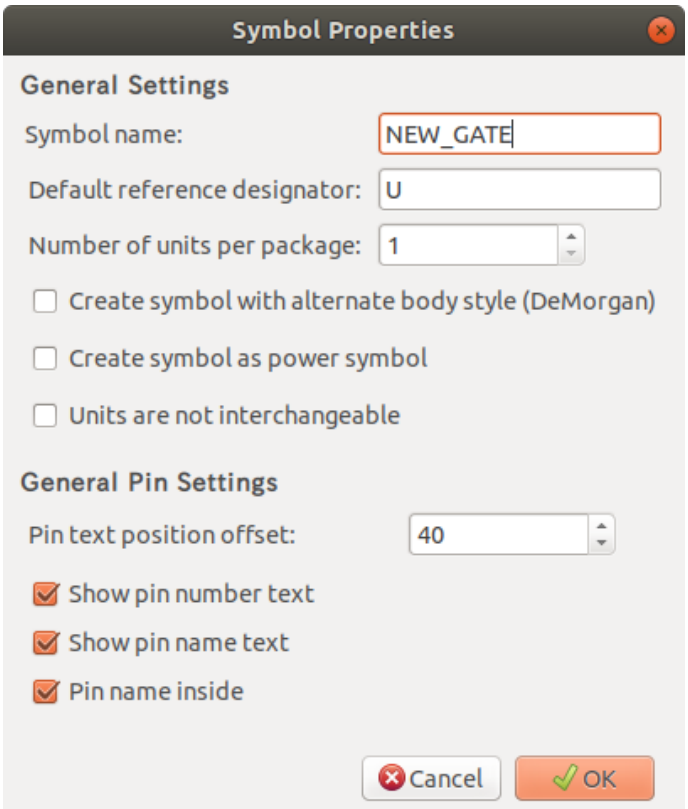
12.4.1.4 丢弃符号变化

处理符号时，编辑的符号只是其库中实际符号的工作副本。这意味着只要你没有保存它，你可以重新加载它以丢弃所做的所有更改。如果您已在本地内存中更新它并且尚未将其保存到库文件，则可以随时退出并重新启动。Eeschema 将撤消所有更改。

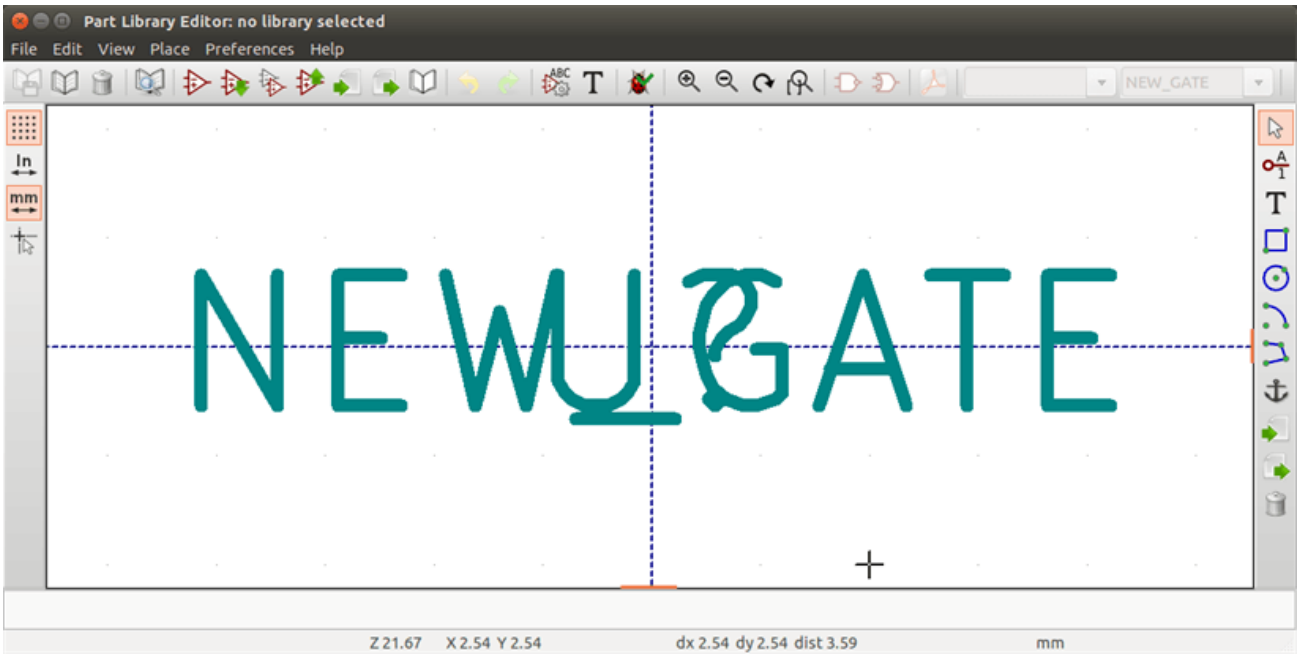
12.5 创建库符号

12.5.1 创建一个新符号

单击图像可以创建新符号:  。系统将要求您输入符号名称（此名称用作原理图编辑器中值字段的默认值），参考编号（U，IC，R …），每个包装的单位数（例如一个 7400 由每个包装 4 个单元组成）并且如果需要替代的体型（有时称为 DeMorgan）。如果参考指示符字段为空，则默认为 U 。稍后可以更改这些属性，但最好在创建符号时正确设置它们。



将使用上面的属性创建一个新符号，它将出现在编辑器中，如下所示。



12.5.2 从另一个符号创建符号

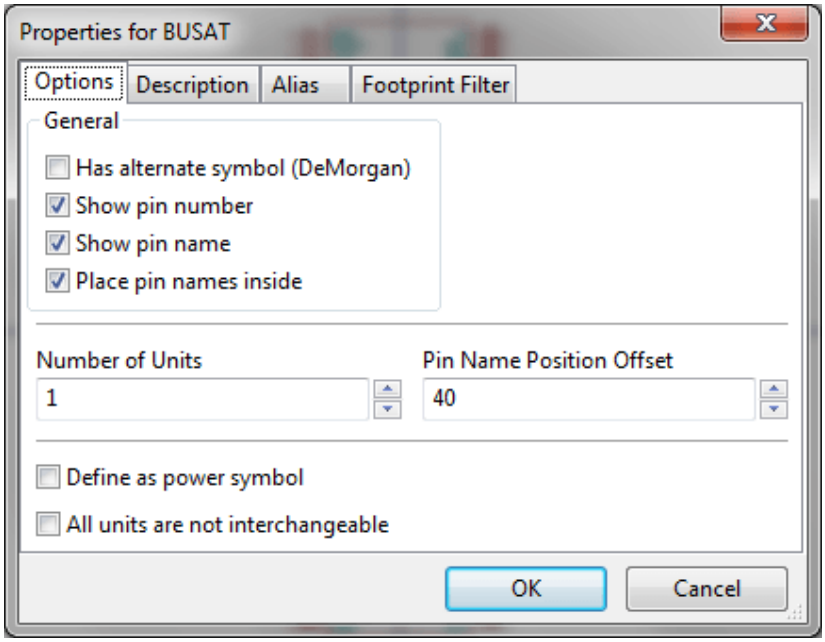
通常，您要制作的符号类似于符号库中已有的符号。在这种情况下，很容易加载和修改现有符号。

- 加载将用作起点的符号。

- 单击 或通过右键单击值字段并编辑文本来修改其名称。如果您选择复制当前符号，系统将提示您输入新的符号名称。
- 如果模型符号具有别名，系统将提示您从新符号中删除与当前库冲突的别名。如果答案为否，则将中止新的符号创建。符号库不能包含任何重复的名称或别名。
- 根据需要编辑新符号。
- 单击图像更新当前库中的新符号： 或单击图像保存到新库： 或者如果要将此新符号保存在其他现有库中，请单击图像选择其他库： 并保存新符号。
- 单击图像将当前库文件保存到磁盘： 。

12.5.3 符号属性

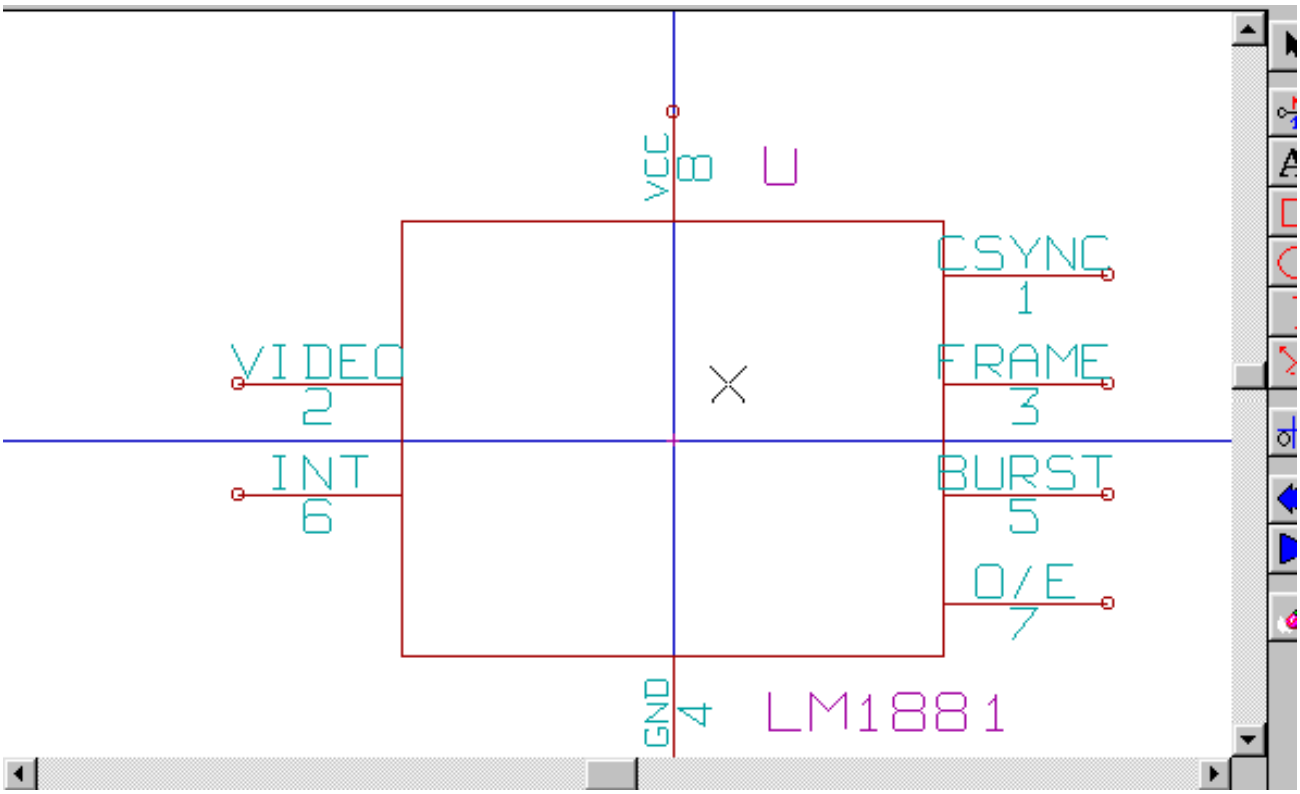
应在符号创建期间仔细设置符号属性，或者从复制的符号继承符号属性。要更改符号属性，请单击 以显示下面的对话框。



正确设置每个封装的单元数和备用符号表示（如果已启用）非常重要，因为在编辑或创建引脚时，每个单元的相应引脚都会受到影响。如果在创建和编辑引脚后更改每个封装的单元数，则还需要额外的工作来添加新的单元引脚和符号。然而，可以随时修改这些属性。

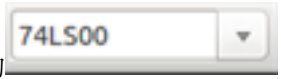
图形选项 *显示引脚编号* 和 *显示引脚名称* 定义了引脚编号和引脚名称文本的可见性。如果选中相应的选项，则此文本将可见。*将引脚名称放在内部* 选项定义了相对于引脚体的引脚名称位置。如果选中该选项，则该文本将显示在符号轮廓内。在这种情况下，*引脚名称位置偏移* 属性定义文本从引脚的主体端移开。30 到 40（1/1000 英寸）的值是合理的。

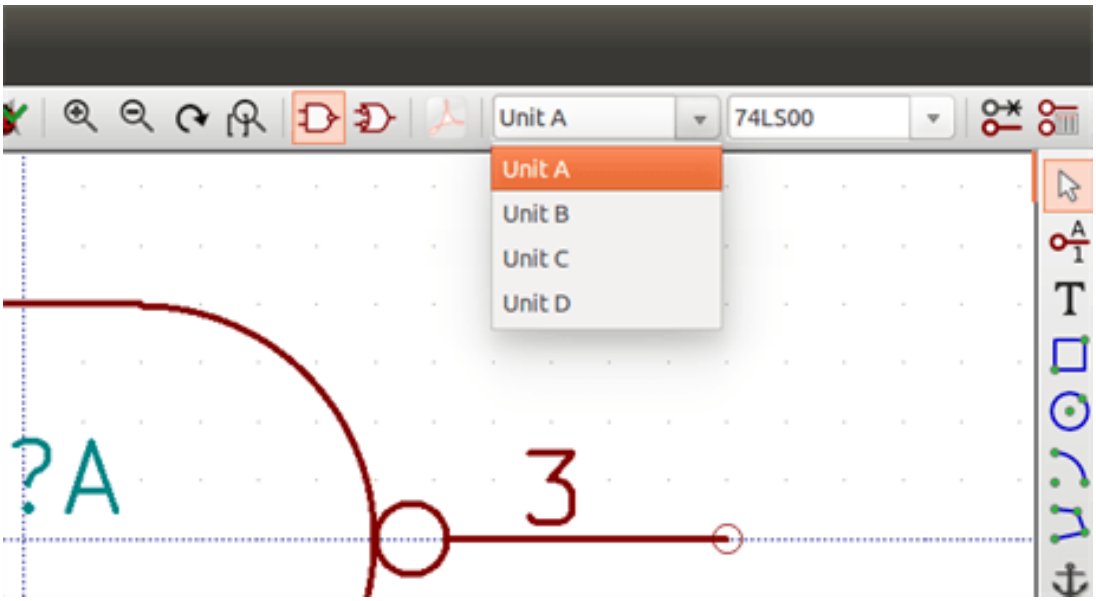
下面的示例显示了未选中“放置引脚名称”选项的符号。注意名称和引脚号的位置。



12.5.4 带有替代符号表示的符号

如果符号具有多个符号再存储，则必须选择一个表示来编辑它们。要编辑常规表示，请单击  。

要编辑备用表示，请单击  。使用下面显示的  选择要编辑的单位。



12.6 图形元素

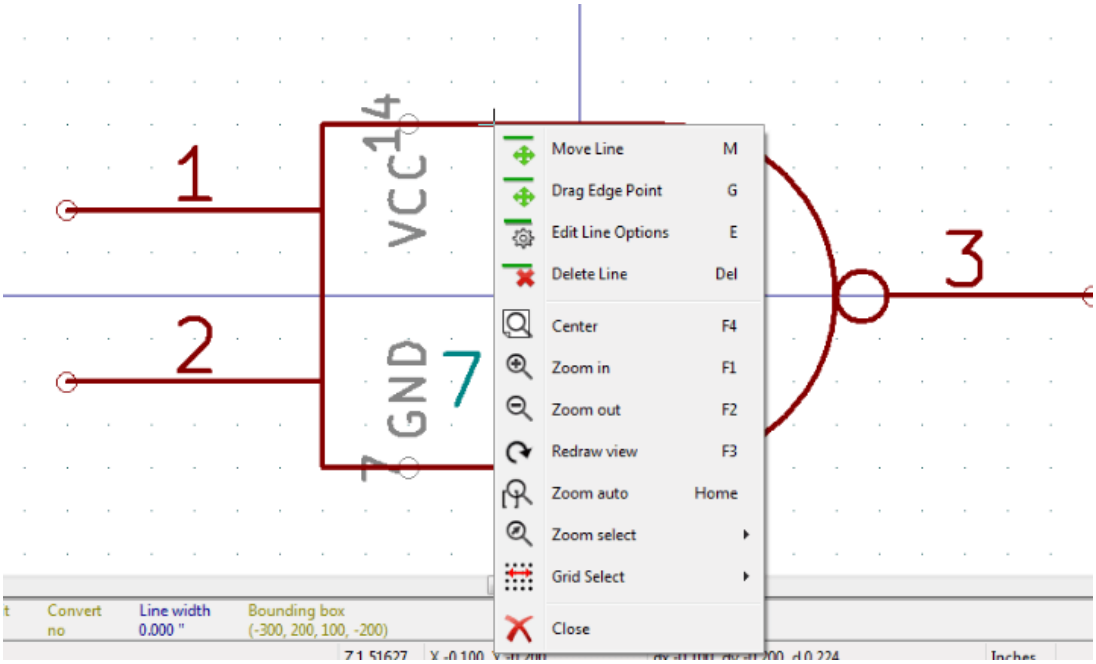
图形元素创建符号的表示，不包含电气连接信息。使用以下工具可以设计它们：

- 由起点和终点定义的线和多边形。
- 由两个对角线定义的矩形。
- 由中心和半径定义的圆。
- 由弧的起点和终点及其中心定义的弧。弧度从 0° 到 180°。

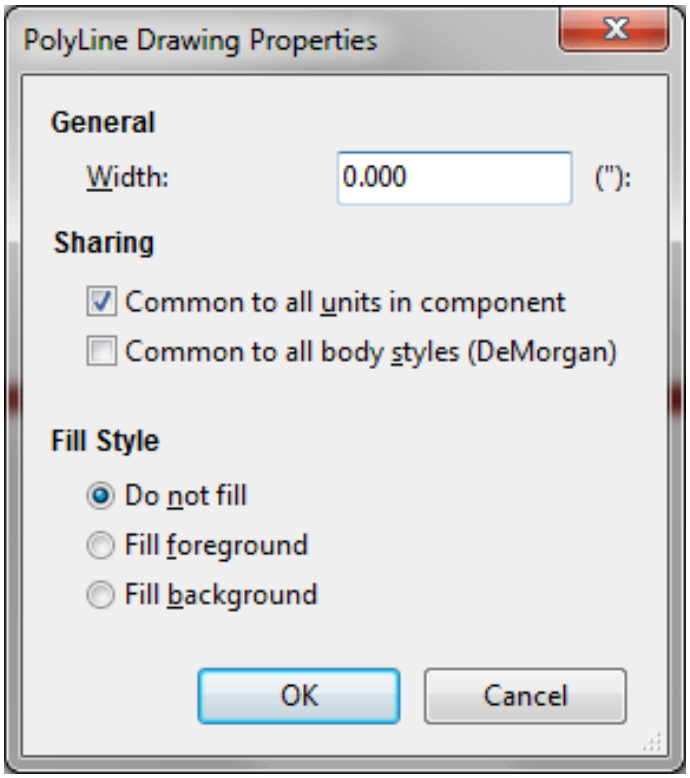
主窗口右侧的垂直工具栏允许您放置设计符号表示所需的所有图形元素。

12.6.1 图形元素成员资格

每个图形元素（线，弧，圆等）可以被定义为对于所有单元和/或主体样式是共同的或者对于给定单元和/或主体样式是特定的。右键单击元素可以快速访问元素选项，以显示所选元素的上下文菜单。下面是线元素的上下文菜单。



您还可以双击元素以修改其属性。下面是多边形元素的属性对话框。



图形元素的属性是:

- 线宽，用于定义当前图形单位中元素线条的宽度。
- 符号中所有单位的共同设置定义是否为每个包含多个单位的符号绘制图形元素，或者是否仅为当前单位绘制图形元素。
- “所有主体样式的共同（DeMorgan）”设置定义是否为具有替代主体样式的符号中的每个符号表示绘制图形元素，或者是否仅针对当前主体样式绘制图形元素。
- 填充样式设置确定图形元素定义的符号是否要绘制为未填充，背景填充或前景填充。

12.6.2 图形文本元素

这个 T 允许创建图形文本。即使镜像符号，图形文本也始终可读。请注意，图形文本项不是字段。

12.7 每个符号多个单位和替代体型样式

符号可以具有两个符号表示（标准符号和备用符号，通常称为 *DeMorgan* ）和/或每个包具有多于一个单元（例如逻辑门）。某些符号每个封装可以有多个单元，每个单元具有不同的符号和引脚配置。

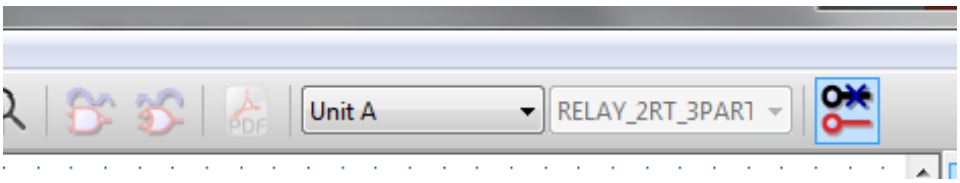
例如，考虑具有两个开关的继电器，其可以被设计为具有三个不同单元的符号：线圈，开关 1 和开关 2. 设计每个封装具有多个单元和/或交替的主体样式的符号是非常灵活的。引脚或体型符号项对于所有单元可以是共同的或者对于给定单元是特定的，或者它们对于两个符号表示可以是共同的，因此特定于给定符号表示。

默认情况下，引脚特定于每个单元的每个符号表示，因为引脚编号特定于单元，并且形状取决于符号表示。当引脚对于每个单元或每个符号表示是公共的时，您需要为所有单元和所有符号表示创建一个引脚（这通常是电源引脚的情况）。这也是身体样式图形形状和文本的情况，每个单元可能是共同的（但通常特定于每个符号表示）。

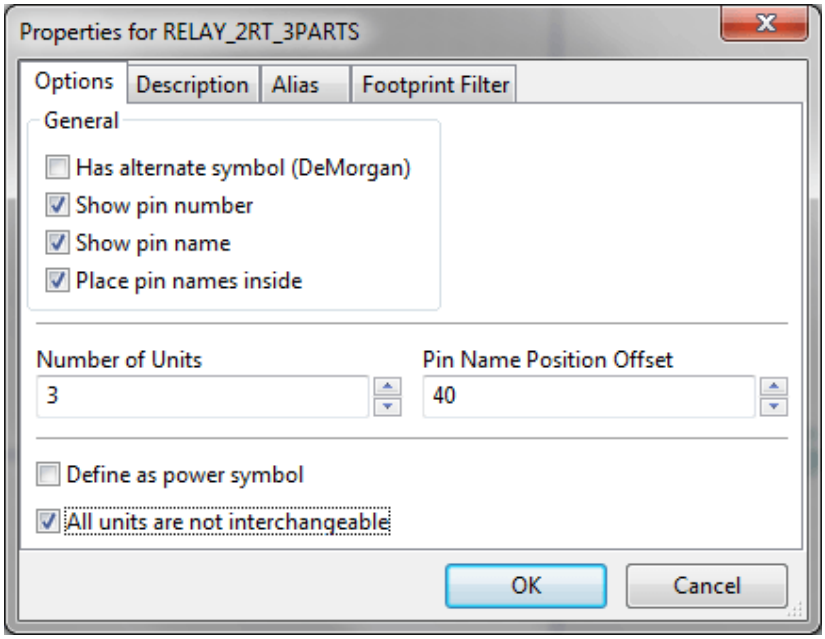
12.7.1 具有不同符号的多个单元的符号示例：

这是一个继电器的例子，每个包装有三个单元，开关 1，开关 2 和线圈：

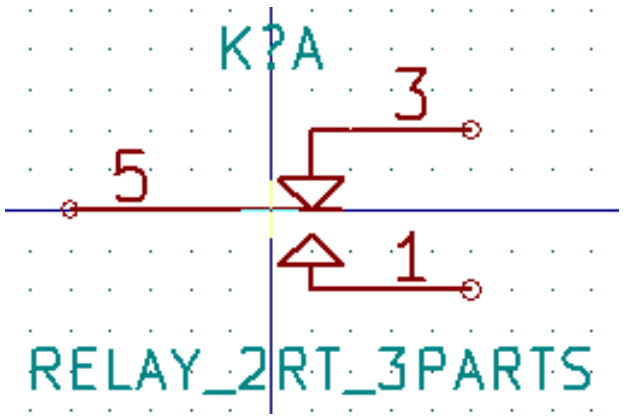
选项：引脚未链接。可以为每个单元添加或编辑引脚，而无需与其他单元的引脚耦合。



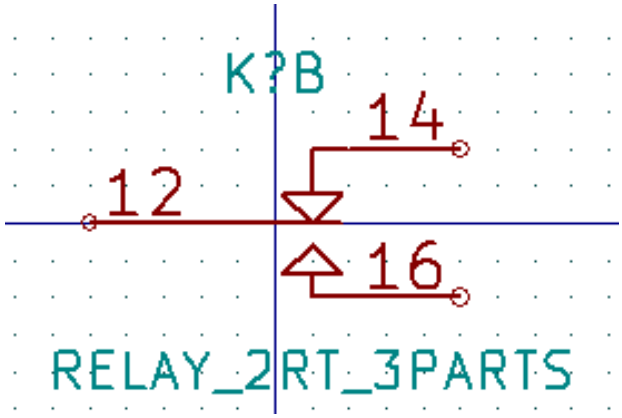
必须选择所有不可互换的单元。



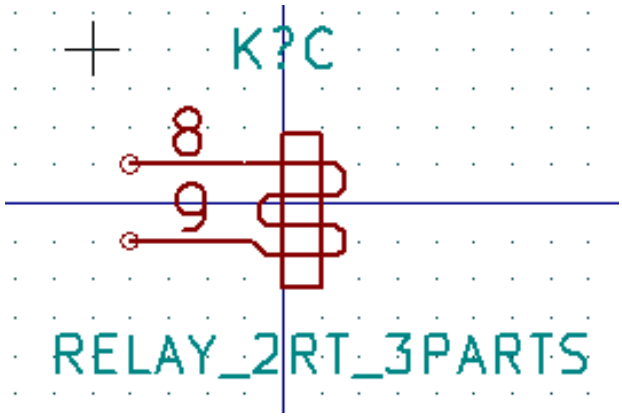
单元 1



单元 2



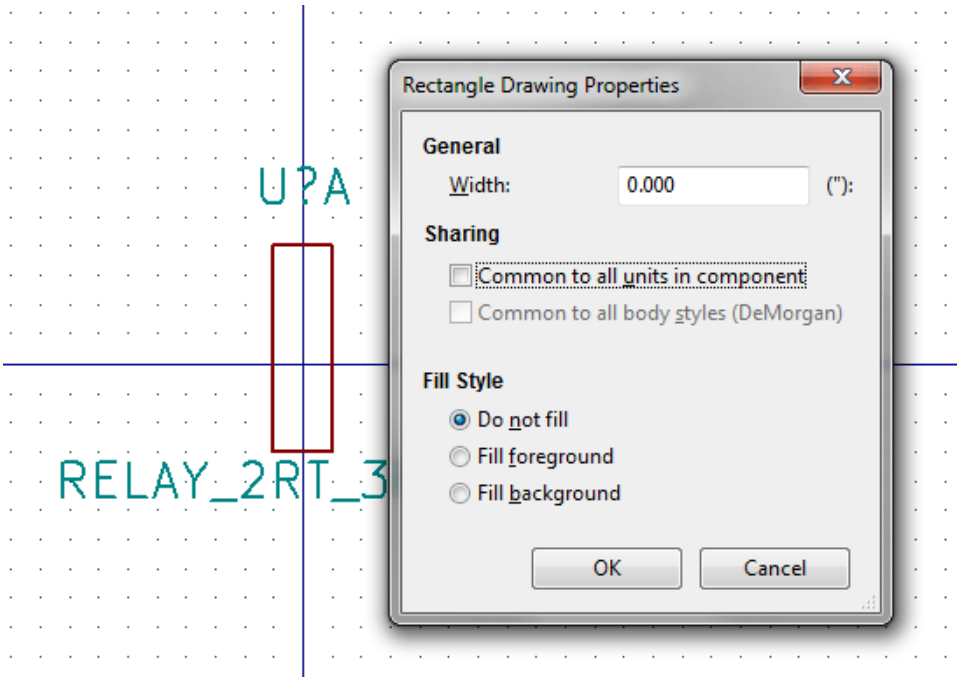
单元 3



它没有相同的符号和引脚布局，因此不能与单元 1 和 2 互换。

12.7.1.1 图形符号元素

下面显示的是图形主体元素的属性。从上面的继电器示例中，三个单元具有不同的符号表示。因此，每个单元都是单独创建的，并且图形主体元素必须禁用“符号中所有单元的通用”。



12.8 引脚创建和编辑

您可以单击 来创建和插入引脚。通过双击引脚或右键单击引脚以打开引脚上下文菜单，可以编辑所有引脚属性。必须仔细创建引脚，因为任何错误都会对 PCB 设计产生影响。可以编辑，删除和/或移动已放置的任何引脚。

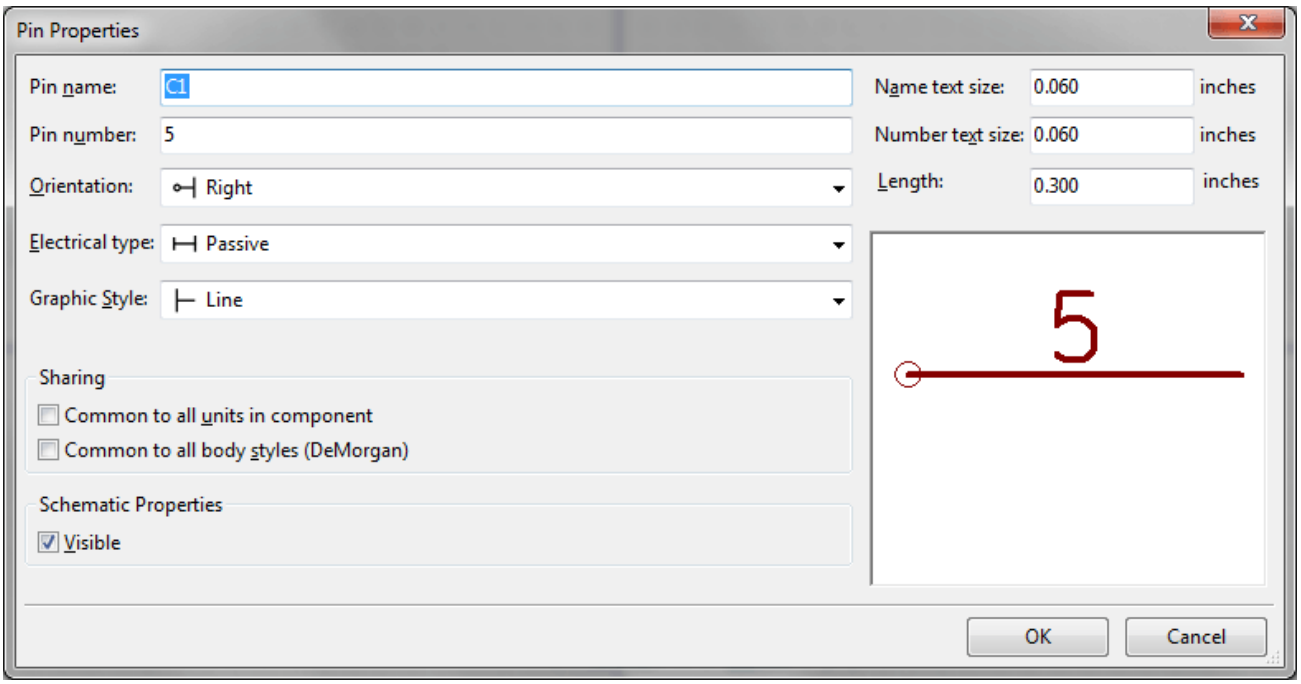
12.8.1 引笔概述

引脚由其图形表示，名称和“编号”定义。引脚的“数字”由一组 4 个字母和/或编号定义。要使电气规则检查（ERC）工具有用，还必须正确定义引脚的“电气”类型（输入，输出，三态……）。如果未正确定义此类型，则原理图 ERC 检查结果可能无效。

重要笔记：

- 不要在引脚名称和数字中使用空格。
- 要使用反转信号（上线）定义引脚名称，请使用“~（代字号）字符。下一个“~”字符将关闭上线。例如“~FO~O”将显示 [上线] #FO #O。
- 如果引脚名称减少为单个符号，则该引脚被视为未命名。
- 以“#”开头的引脚名称保留用于电源端口符号。
- 引脚“编号”由 1 到 4 个字母和/或数字组成。1,2，…9999 是有效数字。A1，B3，Anod，Gnd，Wire 等也有效。
- 符号中不能存在重复的引脚“编号”。

12.8.2 引脚属性

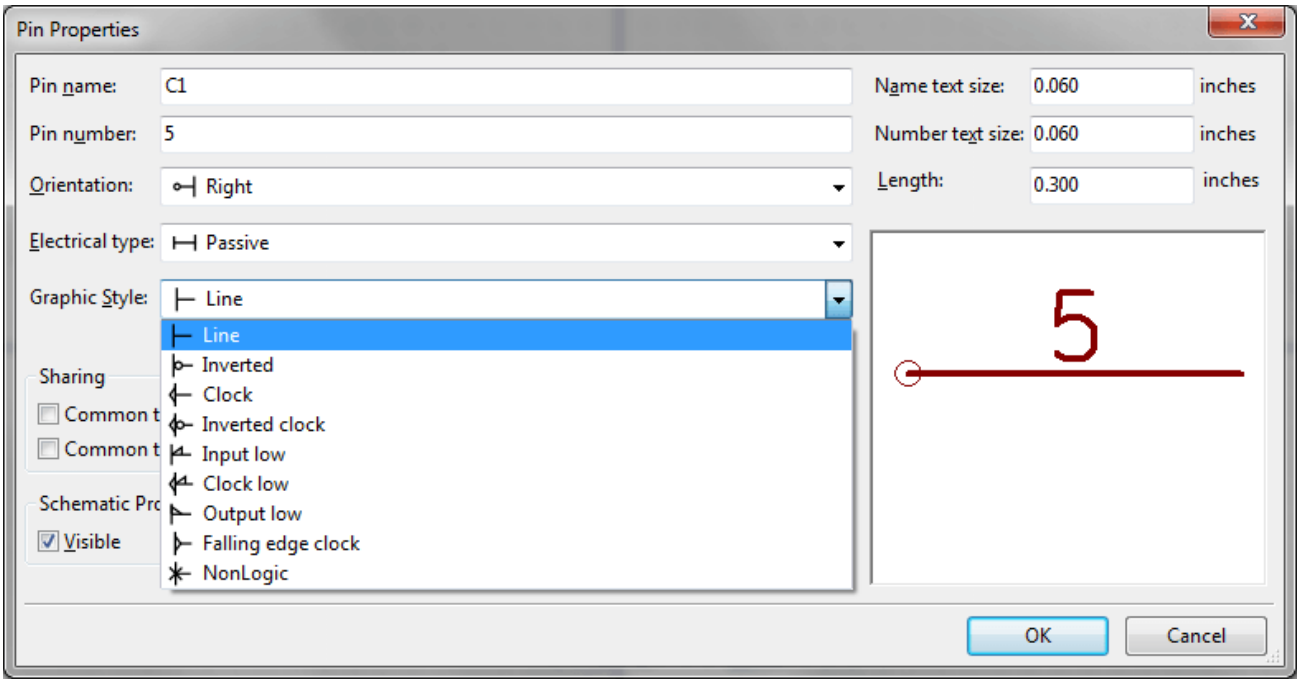


引脚属性对话框允许您编辑引脚的所有特性。创建引脚或双击现有引脚时，会自动弹出此对话框。此对话框允许您修改：

- 名称和名称的文字大小。
- 数字和数字的文字大小。
- 长度。
- 电气和图形类型。
- 单位和替代代表成员资格。
- 可见性。

12.8.3 引脚图形样式

下图显示了不同的图形引脚样式。图形样式的选择对引脚的电气类型没有任何影响。



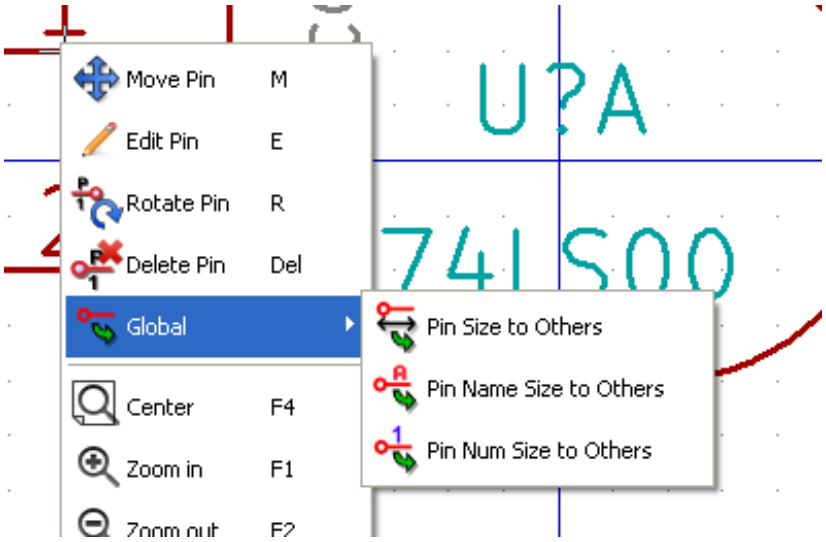
12.8.4 引脚电气类型

选择正确的电气类型对于原理图 ERC 工具很重要。定义的电气类型是：

- 双向指示输入和输出之间可交换的双向引脚（例如微处理器数据总线）。
- 三态是通常的 3 态输出。
- 无源用于无源符号引脚，电阻，连接器等。
- 当 ERC 检查无关紧要时，可以使用未指定的。
- 电源输入用于符号的电源引脚。电源引脚自动连接到具有相同名称的其他电源输入引脚。
- 功率输出用于稳压器输出。
- 开路发射极和开路集电极类型可用于如此定义的逻辑输出。
- 当符号具有没有内部连接的引脚时，使用未连接。

12.8.5 引脚全局属性

您可以使用引脚上下文菜单的全局命令条目修改所有引脚的名称和/或编号的长度或文本大小。单击要修改的参数，然后键入新值，然后将该值应用于所有当前符号的引脚。



12.8.6 为多个单元和备用符号表示定义引脚

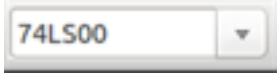
在创建和编辑引脚时，具有多个单元和/或图形表示的符号尤其成问题。大多数引脚特定于每个单元（因为它们的引脚编号特定于每个单元）和每个符号表示（因为它们的形式和位置特定于每个符号表示）。对于每个封装具有多个单元的符号和替代符号表示，引脚的创建和编辑可能是有问题的。符号库编辑器允许同时创建引脚。默认情况下，对多个单位符号的所有单位以及具有替代符号表示的符号的两个表示都进行对引脚所做的更改。

唯一的例外是引脚的图形类型和名称。建立此依赖关系以便在大多数情况下更容易创建和编辑引脚。可以通过切换主工具栏上的  来禁用此依赖关系。这将允许您完全独立地为每个单元和表示创建引脚。

符号可以具有两个符号表示（表示为“Demorgan”），并且可以由多个单元组成，如具有逻辑门的符号的情况。对于某些符号，您可能需要几个不同的图形元素和引脚。与“前一部分具有多个不同符号的符号的示例”，中所示的继电器样本类似，继电器可以由三个不同的单元表示：线圈，开关触点 1 ，并切换开关触点 2。

具有多个单元的符号的管理和具有替代符号表示的符号是灵活的。引脚可以是通用的或特定于不同的单元。引脚也可以是符号表示或每个符号表示特有的。

默认情况下，引脚特定于每个单元的每个表示，因为它们的数量因每个单元而不同，并且它们的设计对于每个符号表示是不同的。当一个引脚对所有单元都是通用的时，它只需要绘制一次，例如在电源引脚的情况下。

一个例子是输出引脚 7400 四路双输入与非门。由于有四个单元和两个符号表示，因此在符号定义中定义了八个单独的输出引脚。创建新的 7400 符号时，正常符号表示的单元 A 将显示在库编辑器中。要在备用符号表示中编辑引脚样式，必须首先通过单击工具栏上的  按钮启用它。要编辑每个单元的引脚编号，请使用以下图像选择相应的单元： 

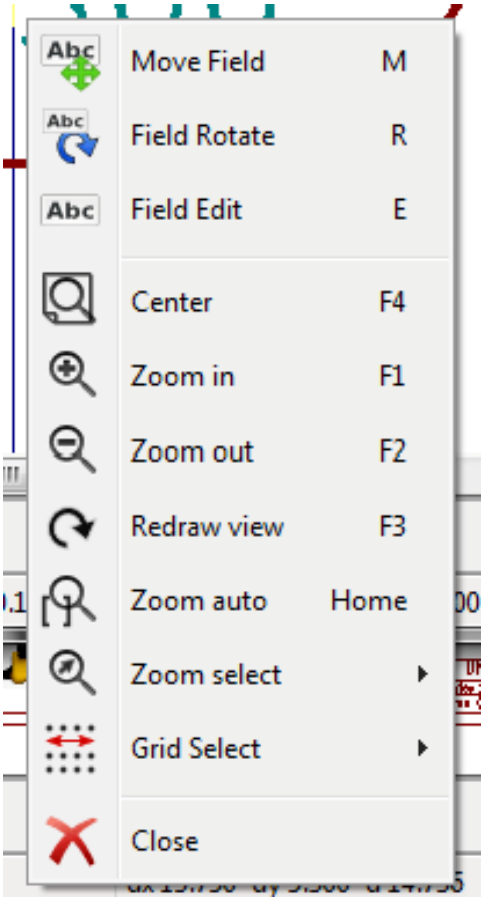
下拉控件。

12.9 符号字段

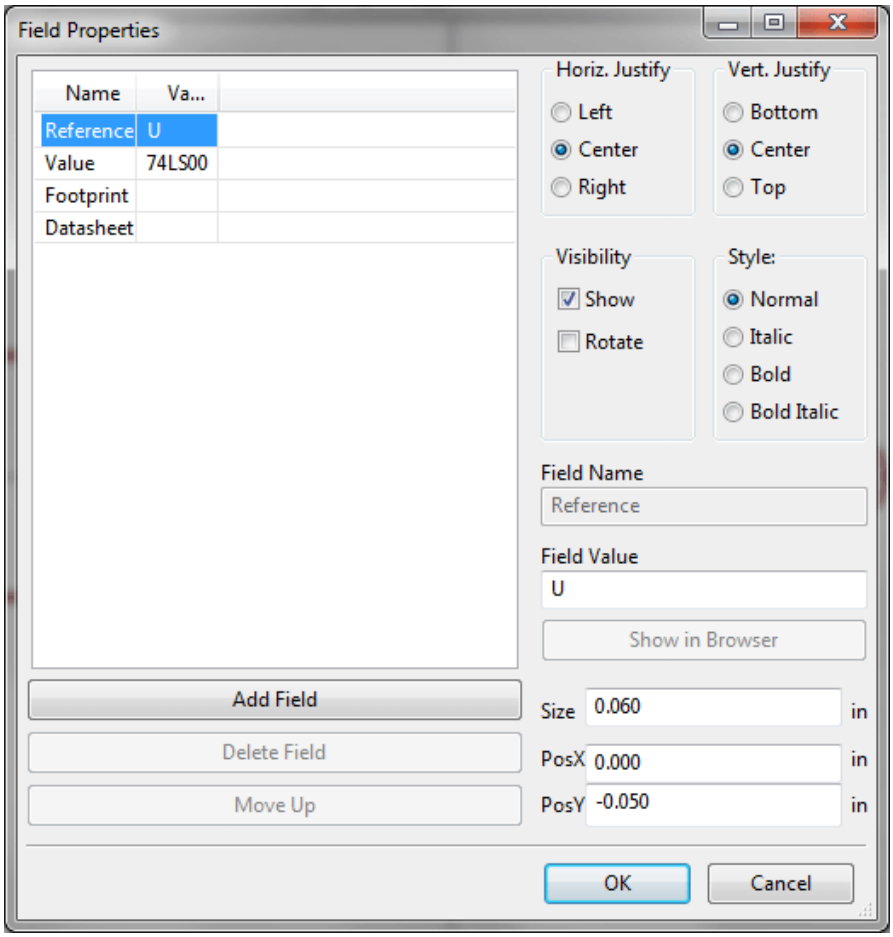
所有库符号都定义了四个默认字段。无论何时创建或复制符号，都会创建参考指示符，值，占用空间分配和文档文件链接字段。仅需要参考指示符和值字段。对于现有字段，可以通过右键单击引脚来使用上下文菜单命令。库中定义的符号通常使用这四个默认字段定义。诸如供应商，部件号，单位成本等附加字段可以添加到库符号中，但通常这是在原理图编辑器中完成的，因此附加字段可以应用于原理图中的所有符号。

12.9.1 编辑符号字段

要编辑现有符号字段，请右键单击字段文本以显示下面显示的字段上下文菜单。



要编辑未定义的字段，添加新字段或删除主工具栏上的可选字段 image:images/icons/text.png[icons/text.png] ，以打开下面显示的字段属性对话框。



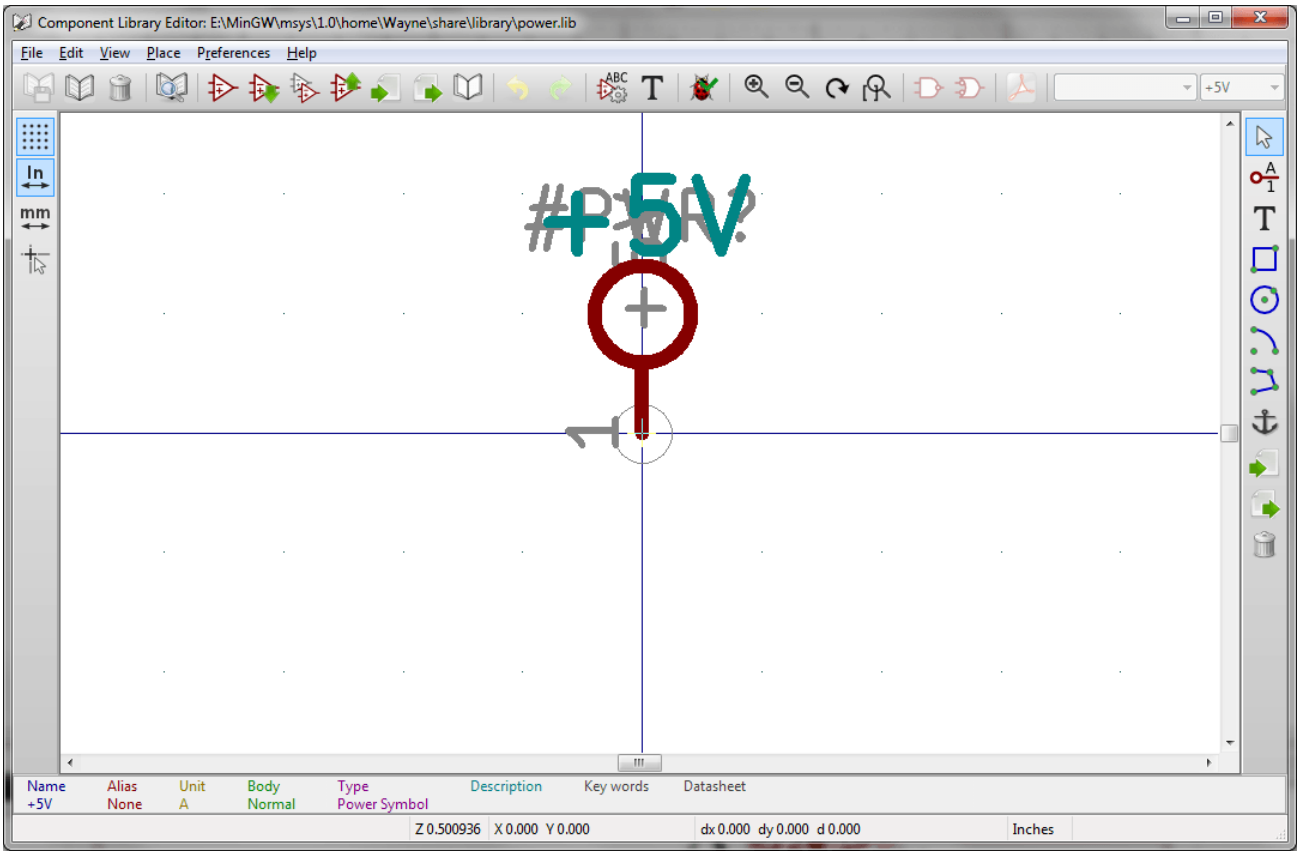
字段是与符号关联的文本部分。不要将它们与属于此符号图形表示的文本混淆。

重要笔记:

- 修改值字段有效地使用当前符号作为新符号的起点创建新符号。将新符号保存到当前选定的库时，该值将包含在值字段中。
- 上面的字段编辑对话框必须用于编辑空的字段或启用了不可见属性。
- 封装定义为使用 LIBNAME:FPNAME 格式的绝对封装，其中 LIBNAME 是封装库表中定义的封装库的名称（请参阅 Pcbnew *参考手册*中的 *封装库表*部分），FPNAME 是库 LIBNAM 中的封装名称。

12.10 电源符号

功率符号的创建方式与普通符号相同。将它们放在专用库（如 power.lib）中可能很有用。电源符号由图形符号和 *电源隐藏*类型的引脚组成。原理图捕获软件可以像处理任何其他符号一样处理电源端口符号。一些预防措施至关重要。以下是电源 + 5V 符号的示例。



要创建电源符号，请使用以下步骤:

- 添加名为 +5V 的“电源输入”类型的引脚（重要的是因为此名称将建立与网络 +5V 的连接），引脚编号为 1（不重要的数字），长度为 0，以及“线路”“图形风格”。
- 如图所示，将一个小圆圈和一个从引脚到圆圈的部分放置。
- 符号的锚点在引脚上。
- 符号值为“+5V”。
- 符号引用是“#+5V”。除了必须为“”的第一个字符表示符号是幂符号外，参考文本并不重要。按照惯例，引用字段以“”开头的每个符号都不会出现在符号列表或网表中，并且引用被声明为不可见。

创建新的电源端口符号的更简单方法是使用另一个符号作为模型:

- 加载现有的电源符号。
- 使用新电源符号的名称编辑引脚名称。
- 如果要显示电源端口值，请将值字段编辑为与引脚相同的名称。
- 保存新符号。

Chapter 13

LibEdit - 符号

13.1 概述

符号由以下元素组成

- 图形表示（几何形状，文本）。
- 引脚。
- 后置处理器使用的字段或关联文本：网表，符号列表。

要初始化两个字段：引用和值。与符号关联的设计的名称以及关联的足迹的名称，其他字段是空闲字段，它们通常可以保持为空，并且可以在原理图捕获期间填充。

但是，管理与任何符号相关的文档有助于库的研究，使用和维护。相关文档包括

- 一行注释。
- 一系列关键字，如 TTL CMOS NAND2，由空格分隔。
- 附加文件名（例如应用程序注释或 pdf 文件）。
  附加文件的默认目录：
  kicad/share/library/doc
  如果没有找到：
  kicad/library/doc
  在 linux 下：
  /usr/local/kicad/share/library/doc
  /usr/share/kicad/library/doc
  /usr/local/share/kicad/library/doc

关键字允许您根据各种选择标准有选择地搜索符号。注释和关键字显示在各种菜单中，特别是从库中选择符号时。

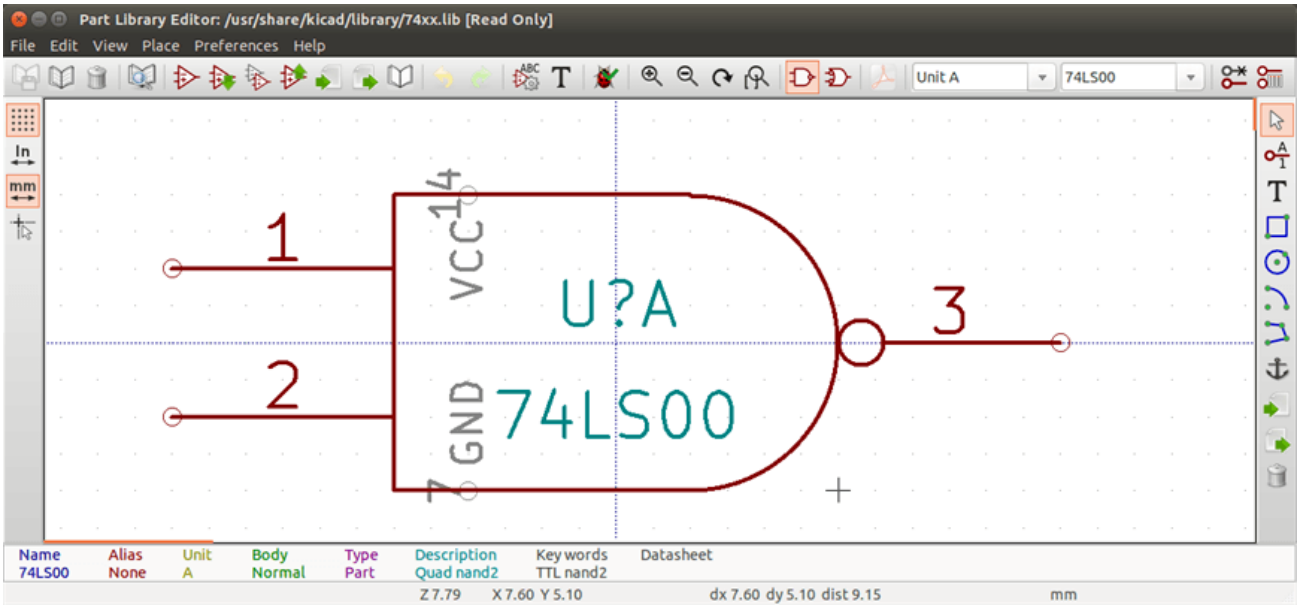
符号也有一个锚点。相对于该锚点进行旋转或镜像，并且在放置期间，该点用作参考位置。因此准确定位该锚是有用的。

符号可以具有别名，即等效名称。这允许您显着减少需要创建的符号数量（例如，74LS00 可以具有别名，例如 74000,74HC00,74HCT00 ……）。

最后，符号分布在库中（按主题或制造商分类）以便于管理。

13.2 定位符号锚点

锚点位于坐标（0,0）处，并由屏幕上显示的蓝色轴显示。



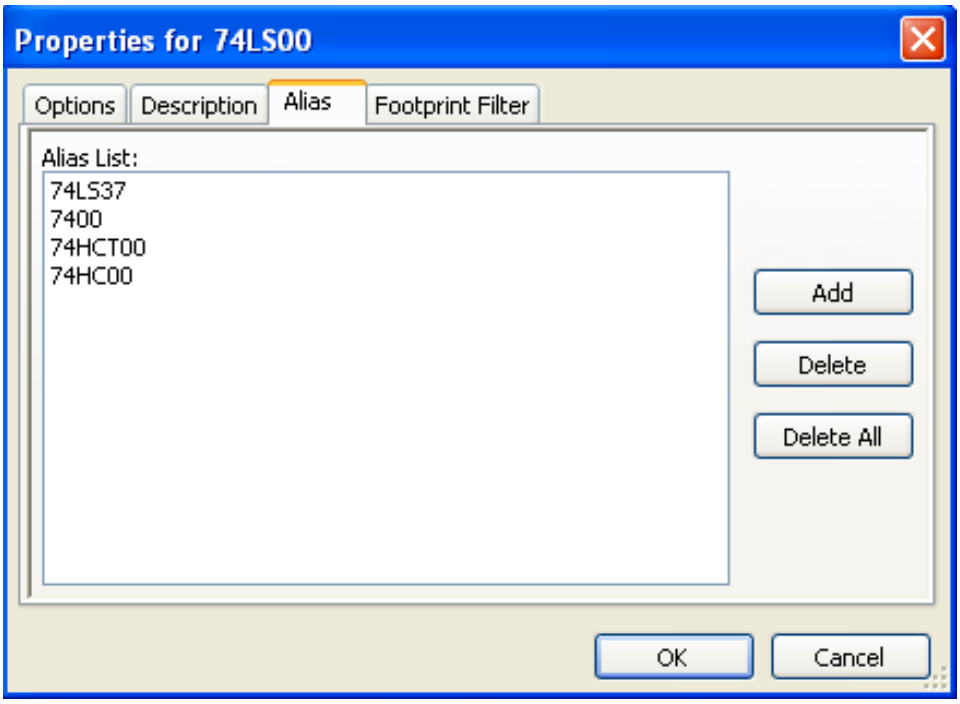
可以通过选择图标： 并单击新的所需锚点位置来重新定位锚点。绘图将自动重新定位在新锚点上。

13.3 符号别名

别名是与库中相同符号对应的另一个名称。具有类似引脚和表示的符号然后可以仅由一个符号表示，具有若干别名（例如，具有别名 74LS00,74HC00,74LS37 的 7400）。

使用别名可以快速构建完整的库。此外，这些库更加紧凑，可以轻松加载 KiCad。

要修改别名列表，必须通过图标 选择主编辑窗口，然后选择别名文件夹。



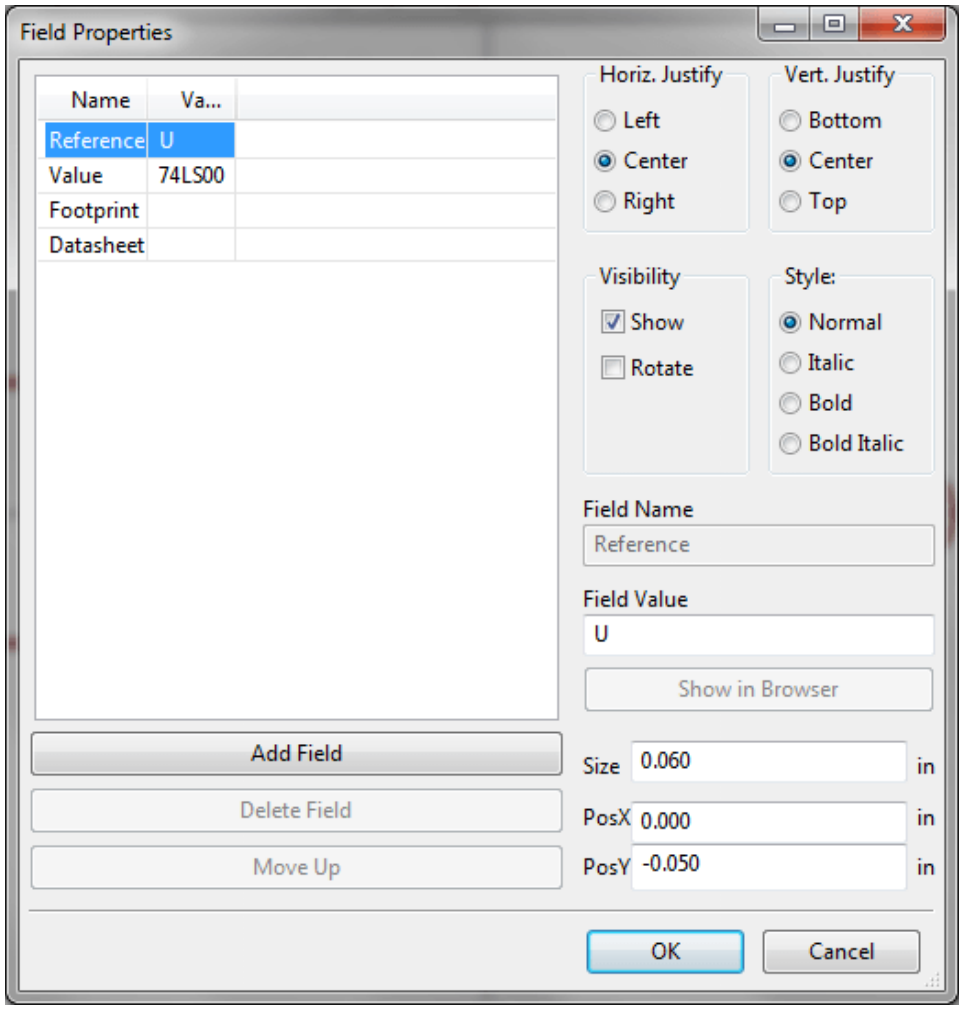
因此，您可以添加或删除所需的别名。由于编辑了当前的别名，因此显然无法将其删除。

要删除所有别名，首先要选择根符号。选择主工具栏窗口中别名列表中的第一个符号。

13.4 符号字段

字段编辑器通过图标调用： T 。

有四个特殊字段（符号附加的文本）和可配置的用户字段

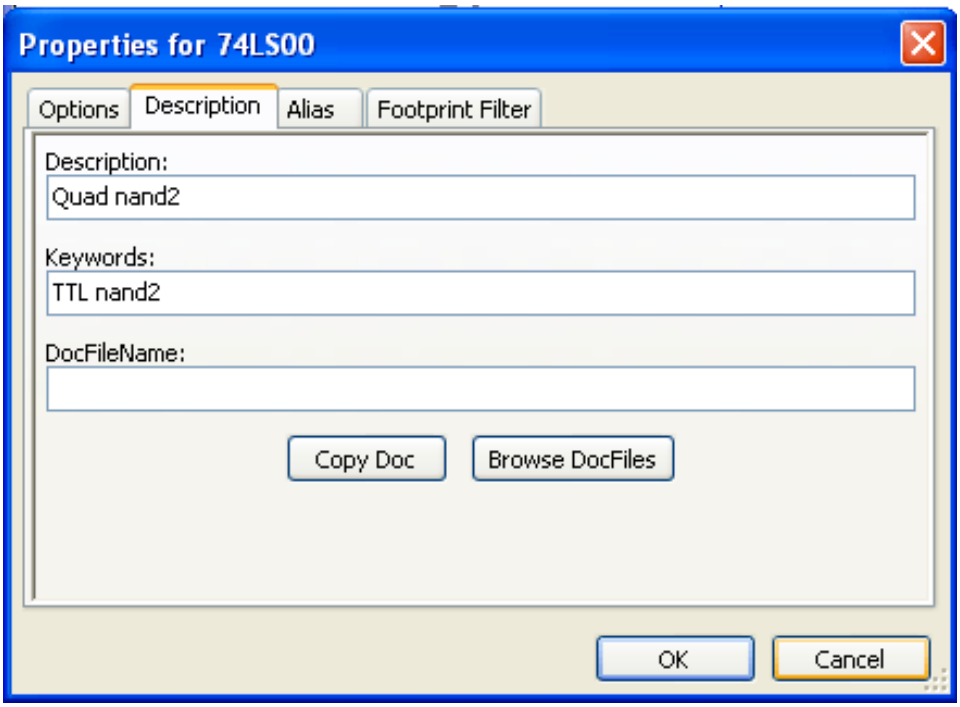


特殊字段

- 参考。
- 值。它是库中的符号名称和原理图中的默认值字段。
- 封装。它是用于电路板的封装名称。使用 CvPcb 设置封装列表时不是很有用，但如果不使用 CvPcb 则必须使用。
- 表。它是一个保留字段, 在编写时不使用。

13.5 符号文档

要编辑文档信息，需要通过图标 调用符号的主编辑窗口，并选择文档文件夹。



请务必选择正确的别名或根符号，因为此文档是别名之间唯一不同的特征。“复制文档”按钮允许您将文档信息从根符号复制到当前编辑的别名。

13.5.1 符号关键字

关键字允许您根据特定的选择标准（功能，技术系列等）以选择的方式搜索符号

Eeschema 研究工具不区分大小写。库中使用的最新关键词是

- 用于逻辑系列的 CMOS TTL
- AND2 NOR3 XOR2 INV …用于门（AND2 = 2 输入 AND 门，NOR3 = 3 输入 NOR 门）。
- JKFF DFF …用于 JK 或 D 触发器。
- ADC, DAC, MUX…
- 具有开路集电极输出的门的 OpenCol。因此，如果在原理图捕获软件中，您搜索符号：通过关键字 NAND2 OpenCol Eeschema 将显示具有这两个关键字的符号列表。

13.5.2 符号文档（Doc）

注释行（和关键字）显示在各种菜单中，尤其是在库的显示符号列表和 ViewLib 菜单中选择符号时。

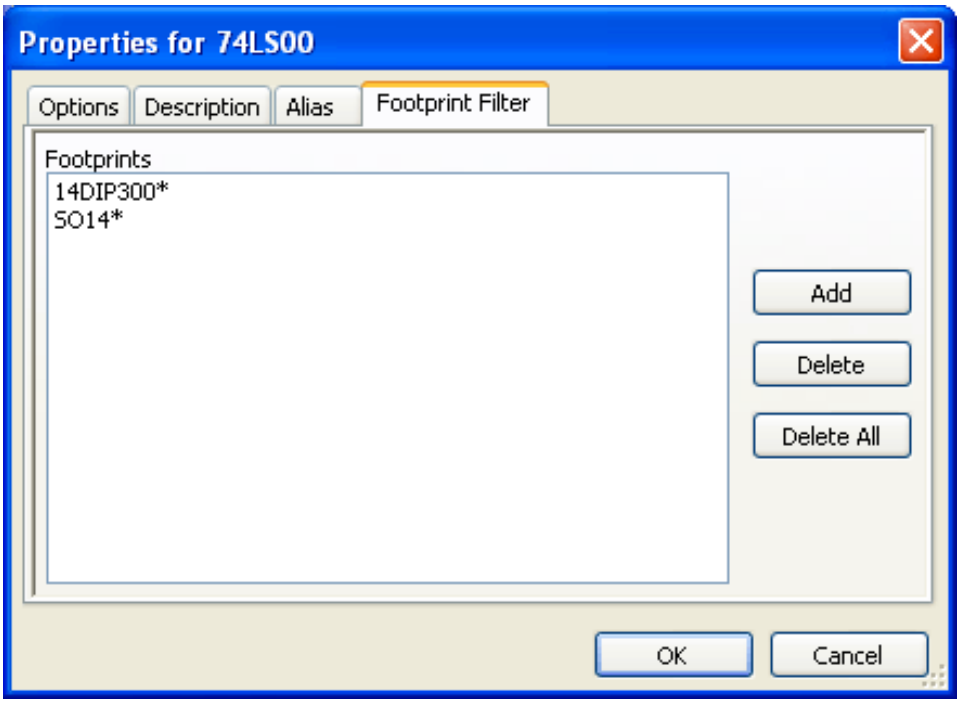
如果这个 Doc. 文件存在，也可以在原理图捕获软件中，通过右键单击符号显示的弹出菜单中访问它。

13.5.3 相关文档文件（DocFileName）

表示可用的附件（文档，应用原理图）（pdf 文件，原理图等）。

13.5.4 CvPcb 的封装过滤

您可以输入符号允许的覆盖区列表。此列表充当 CvPcb 用于仅显示允许的覆盖区的过滤器。无效列表不会过滤任何内容。



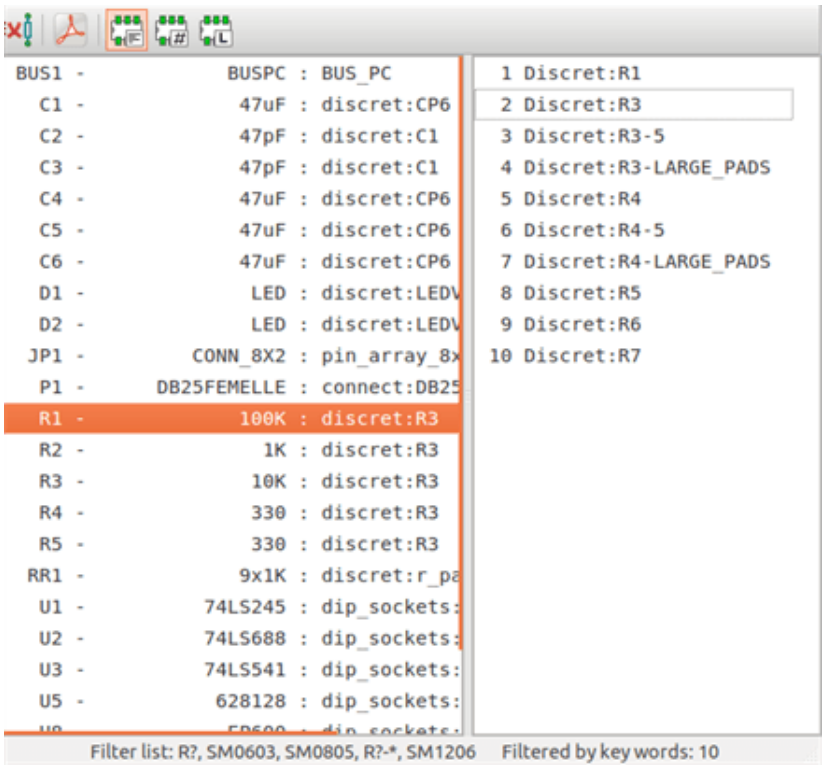
允许使用通配符。

S014* 允许 CvPcb 显示名称以 SO14 开头的所有封装。

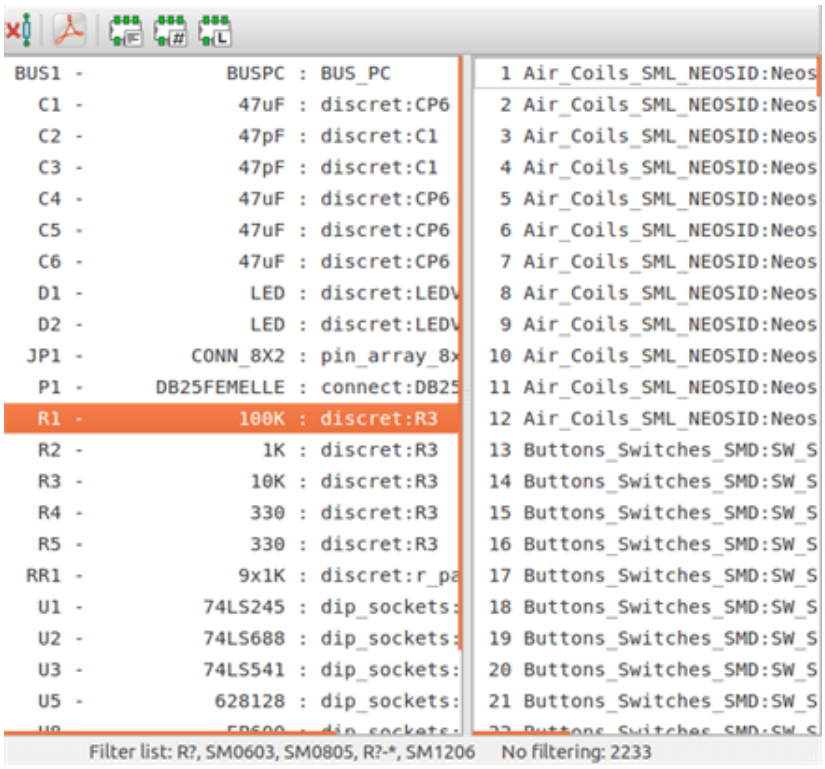
对于电阻器，R? 显示所有带有以 R 开头的 2 个字母名称的封装。

以下是样本：有和没有过滤

有过滤



没有过滤



13.6 符号库

您可以轻松编译包含常用符号的图形符号库文件。这可以用于创建符号（三角形，与，或，异或门等的形状）以用于保存和随后的重复使用。

这些文件默认存储在库目录中，并具有“.sym”扩展名。这些符号不像普通符号那样收集在库中，因为它们通常不是那么多。

13.6.1 导出或创建符号

可以使用按钮图像导出符号：  。您通常只能创建一个图形，删除所有引脚（如果存在）也是一个好主意。

13.6.2 导入符号

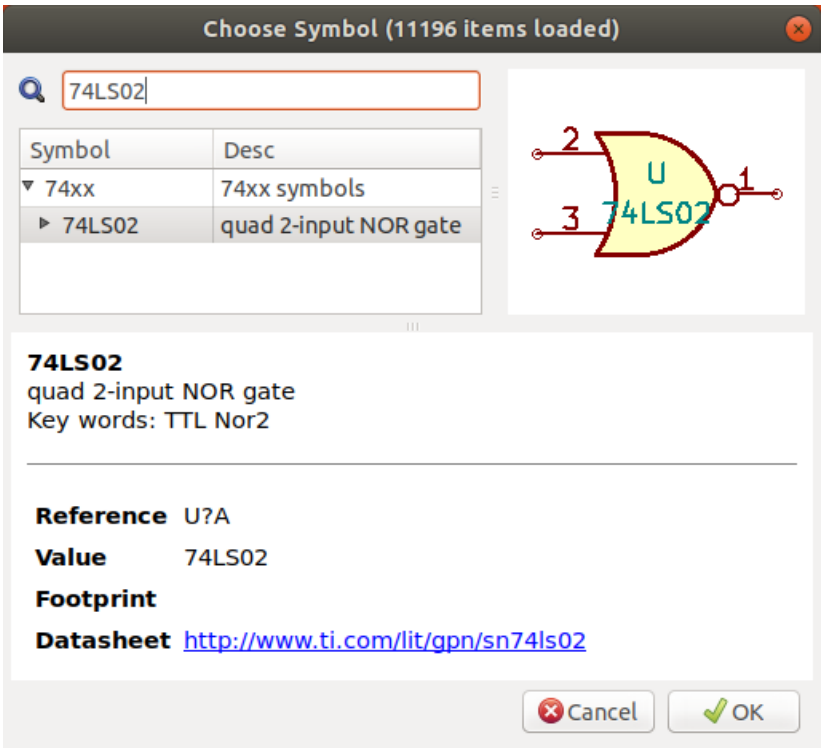
导入允许您将图形添加到正在编辑的符号中。使用按钮图像导入符号：  。导入的图形在现有图形中创建时添加。

Chapter 14

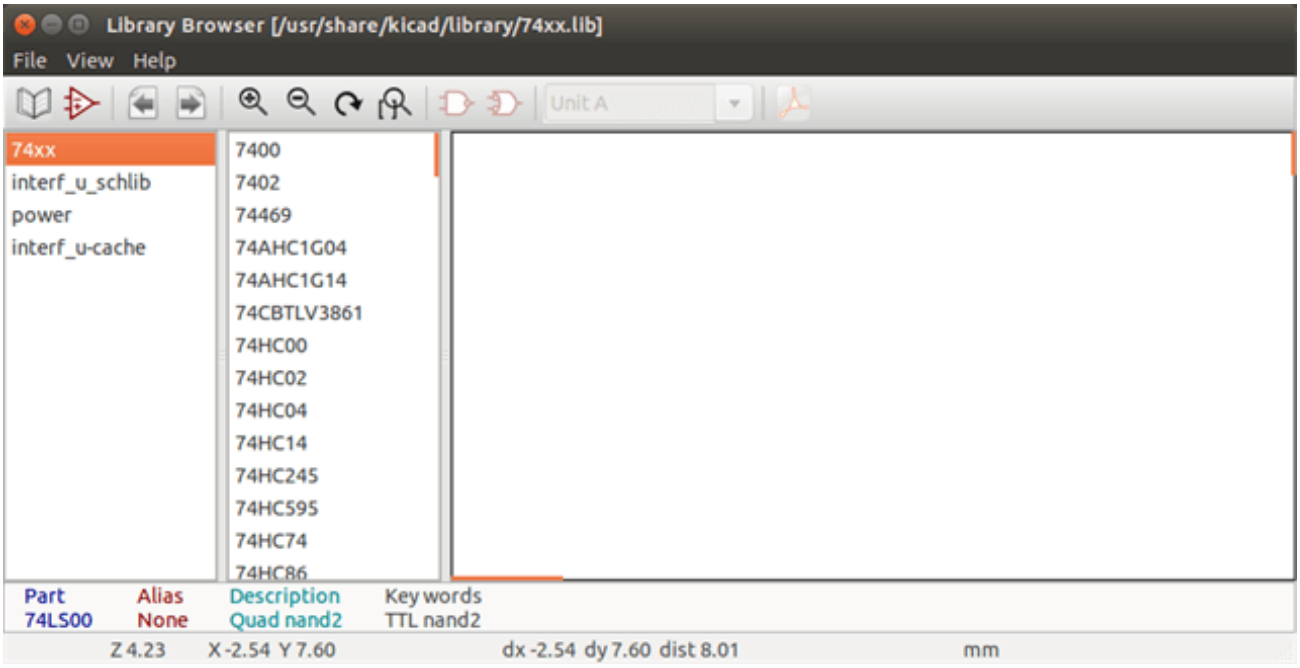
符号库浏览器

14.1 简介

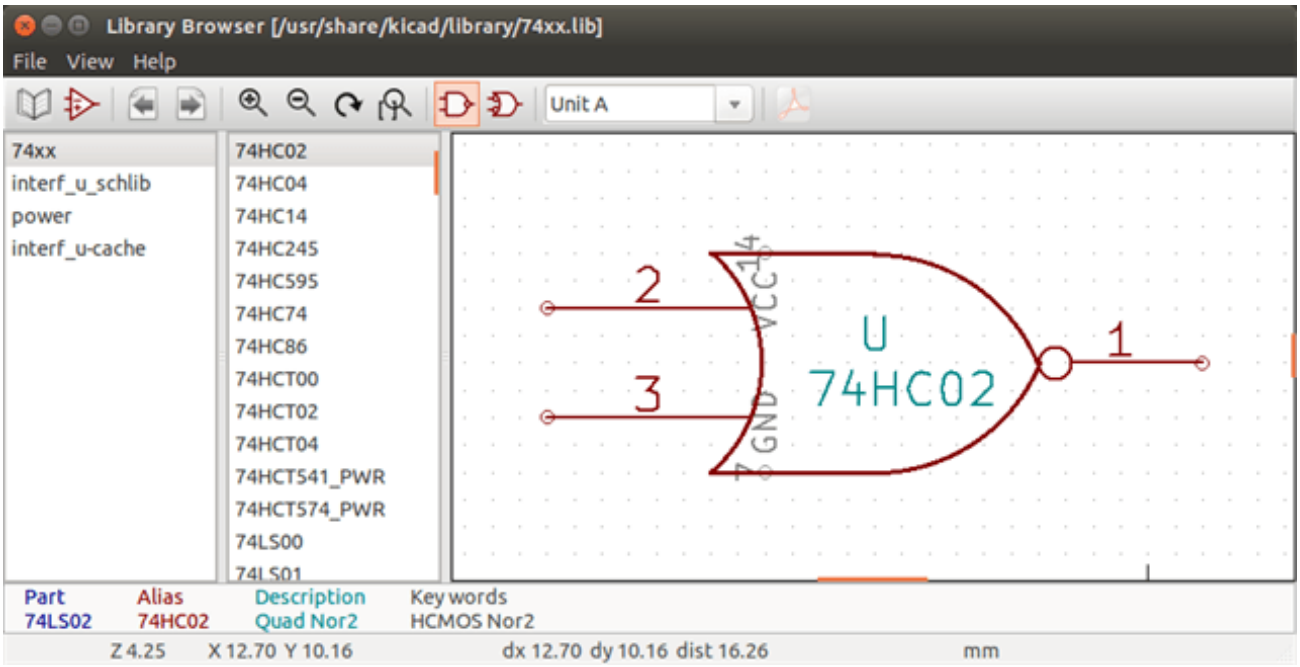
符号库浏览器允许您快速检查符号库的内容。可以通过单击主工具栏上的  图标，在 视图菜单中选择 库浏览器条目或双击 选择符号上的符号图像来访问符号库查看器窗口。



14.2 视图-主屏幕



要检查库的内容，请从左侧窗格的列表中选择一个库。所选库中的所有符号都将显示在第二个窗格中。选择符号名称以查看符号。



14.3 符号库浏览器顶部工具栏

符号库浏览器中的顶部工具栏如下所示。

可用的命令是：

	选择所需的库，也可以在中选择显示列表。
	选择可在显示中选择的符号名单。
	显示上一个符号。
	显示下一个符号。
	缩放工具。
	如果存在，则选择表示（正常或转换）。
	选择包含多个单位的符号的单位。
	如果存在，则显示关联的文档。仅在被叫时存在来自 Eeschema 的地方符号对话框。
	关闭浏览器并将所选符号放在 Eeschema 中。只有在从 Eeschema 调用浏览器时才会显示此图标（双击在元件选择器中的符号上）。

Chapter 15

创建自定义网表和 BOM 文件

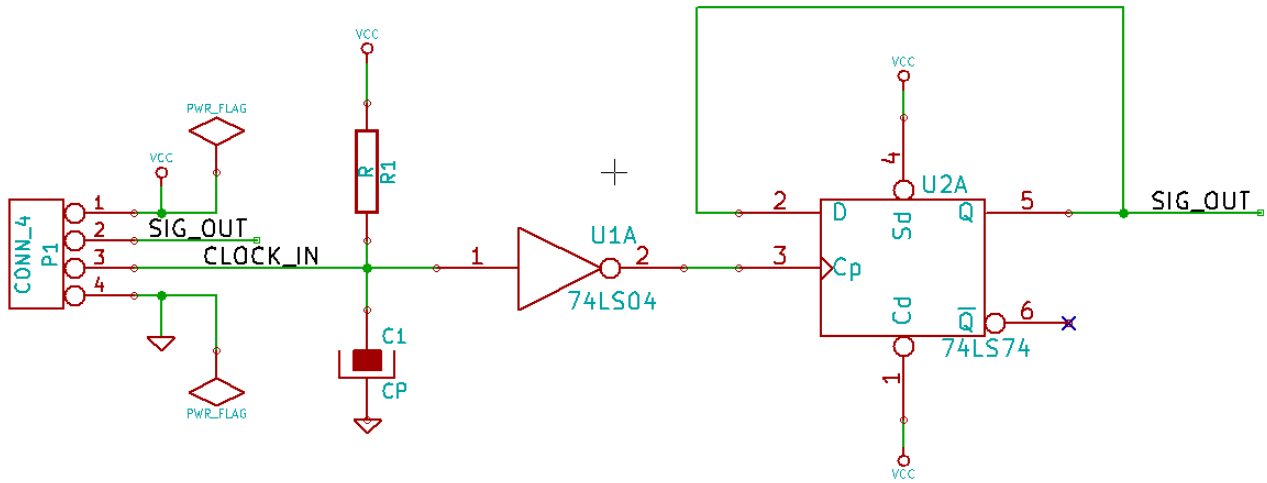
15.1 中间网表文件格式

BOM 文件和网表文件可以从 Eeschema 创建的中间网表文件转换。

此文件使用 XML 语法，称为中间网表。中间网表包含有关您的电路板的大量数据，因此，它可以与后处理一起用于创建 BOM 或其他报告。

根据输出（BOM 或网表），将在后处理中使用完整的中间网表文件的不同子集。

15.1.1 原理图样本



15.1.2 中间网表文件示例

上述电路的相应中间网表 (使用 XML 语法) 如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 20:35:21</date>
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2141</tstamp>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20BA</tstamp>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20A6</tstamp>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2094</tstamp>
    </comp>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E208A</tstamp>
    </comp>
  </components>
  <libparts>
    <libpart lib="device" part="C">
      <description>Condensateur non polarise</description>
      <footprints>
        <fp>SM*</fp>
```

```
      <fp>C?</fp>
      <fp>C1-1</fp>
    </footprints>
    <fields>
      <field name="Reference">C</field>
      <field name="Value">C</field>
    </fields>
    <pins>
      <pin num="1" name="~" type="passive"/>
      <pin num="2" name="~" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="device" part="R">
    <description>Resistance</description>
    <footprints>
      <fp>R?</fp>
      <fp>SM0603</fp>
      <fp>SM0805</fp>
      <fp>R?-*</fp>
      <fp>SM1206</fp>
    </footprints>
    <fields>
      <field name="Reference">R</field>
      <field name="Value">R</field>
    </fields>
    <pins>
      <pin num="1" name="~" type="passive"/>
      <pin num="2" name="~" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="conn" part="CONN_4">
    <description>Symbole general de connecteur</description>
    <fields>
      <field name="Reference">P</field>
      <field name="Value">CONN_4</field>
    </fields>
    <pins>
      <pin num="1" name="P1" type="passive"/>
      <pin num="2" name="P2" type="passive"/>
      <pin num="3" name="P3" type="passive"/>
      <pin num="4" name="P4" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="74xx" part="74LS04">
    <description>Hex Inverseur</description>
    <fields>
      <field name="Reference">U</field>
      <field name="Value">74LS04</field>
```

```
    </fields>
    <pins>
      <pin num="1" name="~" type="input"/>
      <pin num="2" name="~" type="output"/>
      <pin num="3" name="~" type="input"/>
      <pin num="4" name="~" type="output"/>
      <pin num="5" name="~" type="input"/>
      <pin num="6" name="~" type="output"/>
      <pin num="7" name="GND" type="power_in"/>
      <pin num="8" name="~" type="output"/>
      <pin num="9" name="~" type="input"/>
      <pin num="10" name="~" type="output"/>
      <pin num="11" name="~" type="input"/>
      <pin num="12" name="~" type="output"/>
      <pin num="13" name="~" type="input"/>
      <pin num="14" name="VCC" type="power_in"/>
    </pins>
  </libpart>
  <libpart lib="74xx" part="74LS74">
    <description>Dual D FlipFlop, Set &amp;amp; Reset</description>
    <docs>74xx/74hc_hct74.pdf</docs>
    <fields>
      <field name="Reference">U</field>
      <field name="Value">74LS74</field>
    </fields>
    <pins>
      <pin num="1" name="Cd" type="input"/>
      <pin num="2" name="D" type="input"/>
      <pin num="3" name="Cp" type="input"/>
      <pin num="4" name="Sd" type="input"/>
      <pin num="5" name="Q" type="output"/>
      <pin num="6" name="~Q" type="output"/>
      <pin num="7" name="GND" type="power_in"/>
      <pin num="8" name="~Q" type="output"/>
      <pin num="9" name="Q" type="output"/>
      <pin num="10" name="Sd" type="input"/>
      <pin num="11" name="Cp" type="input"/>
      <pin num="12" name="D" type="input"/>
      <pin num="13" name="Cd" type="input"/>
      <pin num="14" name="VCC" type="power_in"/>
    </pins>
  </libpart>
</libparts>
<libraries>
  <library logical="device">
    <uri>F:\kicad\share\library\device.lib</uri>
  </library>
  <library logical="conn">
```

```
      <uri>F:\kicad\share\library\conn.lib</uri>
    </library>
    <library logical="74xx">
      <uri>F:\kicad\share\library\74xx.lib</uri>
    </library>
  </libraries>
  <nets>
    <net code="1" name="GND">
      <node ref="U1" pin="7"/>
      <node ref="C1" pin="2"/>
      <node ref="U2" pin="7"/>
      <node ref="P1" pin="4"/>
    </net>
    <net code="2" name="VCC">
      <node ref="R1" pin="1"/>
      <node ref="U1" pin="14"/>
      <node ref="U2" pin="4"/>
      <node ref="U2" pin="1"/>
      <node ref="U2" pin="14"/>
      <node ref="P1" pin="1"/>
    </net>
    <net code="3" name="">
      <node ref="U2" pin="6"/>
    </net>
    <net code="4" name="">
      <node ref="U1" pin="2"/>
      <node ref="U2" pin="3"/>
    </net>
    <net code="5" name="/SIG_OUT">
      <node ref="P1" pin="2"/>
      <node ref="U2" pin="5"/>
      <node ref="U2" pin="2"/>
    </net>
    <net code="6" name="/CLOCK_IN">
      <node ref="R1" pin="2"/>
      <node ref="C1" pin="1"/>
      <node ref="U1" pin="1"/>
      <node ref="P1" pin="3"/>
    </net>
  </nets>
</export>
```

15.2 转换为新的网表格式

通过将后处理过滤器应用于中间网表文件，您可以生成外部网表文件以及 BOM 文件。由于此转换是文本到文本转换，因此可以使用 Python，XSLT 或任何其他能够将 XML 作为输入的工具来编写此后处理过滤器。

XSLT 本身是一种非常适合 XML 转换的 XML 语言。有一个名为 *xsltproc* 的免费程序，您可以下载并安装。xsltproc 程序可用于读取中间 XML 网表输入文件，应用样式表来转换输入，并将结果保存在输出文件中。使用 xsltproc 需要使用 XSLT 约定的样式表文件。完成转换过程由 Eeschema 处理，在配置一次后以特定方式运行 xsltproc。

15.3 XSLT 方法

描述 XSL 转换（XSLT）的文档可在此处获得：

http://www.w3.org/TR/xslt

15.3.1 创建 Pads-Pcb 网表文件

“pads-pcb” 的格式由两部分组成。

- 封装列表。
- 网表: 按网络对焊盘引用进行分组。

紧接下面是样式表，它将中间网表文件转换为 pad-pcb 网表格式：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--XSL style sheet to Eeschema Generic Netlist Format to PADS netlist format
    Copyright (C) 2010, SoftPLC Corporation.
    GPL v2.

    How to use:
        https://lists.launchpad.net/kicad-developers/msg05157.html
-->

<!DOCTYPE xsl:stylesheet [
  <!ENTITY nl  "&#xd;&#xa;"> <!--new line CR, LF -->
]>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

<xsl:template match="/export">
    <xsl:text>*PADS-PCB*&nl;*PART*&nl;</xsl:text>
    <xsl:apply-templates select="components/comp"/>
    <xsl:text>&nl;*NET*&nl;</xsl:text>
    <xsl:apply-templates select="nets/net"/>
    <xsl:text>*END*&nl;</xsl:text>
</xsl:template>

<!-- for each component -->
<xsl:template match="comp">
    <xsl:text> </xsl:text>
```

```
    <xsl:value-of select="@ref"/>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "footprint != '' ">
            <xsl:apply-templates select="footprint"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>unknown</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!-- for each net -->
<xsl:template match="net">
    <!-- nets are output only if there is more than one pin in net -->
    <xsl:if test="count(node)>1">
        <xsl:text>*SIGNAL* </xsl:text>
        <xsl:choose>
            <xsl:when test = "@name != '' ">
                <xsl:value-of select="@name"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:text>N-</xsl:text>
                <xsl:value-of select="@code"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:text>&nl;</xsl:text>
        <xsl:apply-templates select="node"/>
    </xsl:if>
</xsl:template>

<!-- for each node -->
<xsl:template match="node">
    <xsl:text> </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text>.</xsl:text>
    <xsl:value-of select="@pin"/>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

这是运行 xsltproc 后的 pads-pcb 输出文件:

```
*PADS-PCB*
*PART*
P1 unknown
```

```
U2 unknown
U1 unknown
C1 unknown
R1 unknown
*NET*
*SIGNAL* GND
U1.7
C1.2
U2.7
P1.4
*SIGNAL* VCC
R1.1
U1.14
U2.4
U2.1
U2.14
P1.1
*SIGNAL* N-4
U1.2
U2.3
*SIGNAL* /SIG_OUT
P1.2
U2.5
U2.2
*SIGNAL* /CLOCK_IN
R1.2
C1.1
U1.1
P1.3

*END*
```

进行此转换的命令行是:

```
kicad\\bin\\xsltproc.exe -o test.net kicad\\bin\\plugins\\netlist_form_pads-pcb.xsl test. ↩
    tmp
```

15.3.2 创建一个 Cadstar 网表文件

Cadstar 格式由两个部分组成。

- 封装列表。
- 网表: 按网络对焊盘引用进行分组。

以下是进行此特定转换的样式表文件:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--XSL style sheet to Eeschema Generic Netlist Format to CADSTAR netlist format
    Copyright (C) 2010, Jean-Pierre Charras.
    Copyright (C) 2010, SoftPLC Corporation.
    GPL v2.

<!DOCTYPE xsl:stylesheet [
  <!ENTITY nl  "&#xd;&#xa;"> <!--new line CR, LF -->
]>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

<!-- Netlist header -->
<xsl:template match="/export">
    <xsl:text>.HEA&nl;</xsl:text>
    <xsl:apply-templates select="design/date"/>  <!-- Generate line .TIM <time> -->
    <xsl:apply-templates select="design/tool"/>  <!-- Generate line .APP <eeschema version> ↩
         -->
    <xsl:apply-templates select="components/comp"/>  <!-- Generate list of components -->
    <xsl:text>&nl;&nl;</xsl:text>
    <xsl:apply-templates select="nets/net"/>          <!-- Generate list of nets and  ↩
        connections -->
    <xsl:text>&nl;.END&nl;</xsl:text>
</xsl:template>

 <!-- Generate line .TIM 20/08/2010 10:45:33 -->
<xsl:template match="tool">
    <xsl:text>.APP "</xsl:text>
    <xsl:apply-templates/>
    <xsl:text>"&nl;</xsl:text>
</xsl:template>

 <!-- Generate line .APP "eeschema (2010-08-17 BZR 2450)-unstable" -->
<xsl:template match="date">
    <xsl:text>.TIM </xsl:text>
    <xsl:apply-templates/>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!-- for each component -->
<xsl:template match="comp">
    <xsl:text>.ADD_COM </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "value != '' ">
```

```
            <xsl:text>"</xsl:text> <xsl:apply-templates select="value"/> <xsl:text>"</xsl: ↩
                text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>""</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!-- for each net -->
<xsl:template match="net">
    <!-- nets are output only if there is more than one pin in net -->
    <xsl:if test="count(node)>1">
    <xsl:variable name="netname">
        <xsl:text>"</xsl:text>
        <xsl:choose>
            <xsl:when test = "@name != '' ">
                <xsl:value-of select="@name"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:text>N-</xsl:text>
                <xsl:value-of select="@code"/>
        </xsl:otherwise>
        </xsl:choose>
        <xsl:text>"&nl;</xsl:text>
        </xsl:variable>
        <xsl:apply-templates select="node" mode="first"/>
        <xsl:value-of select="$netname"/>
        <xsl:apply-templates select="node" mode="others"/>
    </xsl:if>
</xsl:template>

<!-- for each node -->
<xsl:template match="node" mode="first">
    <xsl:if test="position()=1">
       <xsl:text>.ADD_TER </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text>.</xsl:text>
    <xsl:value-of select="@pin"/>
    <xsl:text> </xsl:text>
    </xsl:if>
</xsl:template>

<xsl:template match="node" mode="others">
    <xsl:choose>
        <xsl:when test='position()=1'>
        </xsl:when>
```

```
        <xsl:when test='position()=2'>
           <xsl:text>.TER     </xsl:text>
        </xsl:when>
        <xsl:otherwise>
           <xsl:text>         </xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:if test="position()>1">
        <xsl:value-of select="@ref"/>
        <xsl:text>.</xsl:text>
        <xsl:value-of select="@pin"/>
        <xsl:text>&nl;</xsl:text>
    </xsl:if>
</xsl:template>

</xsl:stylesheet>
```

这是 Cadstar 输出文件。

```
.HEA
.TIM 21/08/2010 08:12:08
.APP "eeschema (2010-08-09 BZR 2439)-unstable"
.ADD_COM P1 "CONN_4"
.ADD_COM U2 "74LS74"
.ADD_COM U1 "74LS04"
.ADD_COM C1 "CP"
.ADD_COM R1 "R"


.ADD_TER U1.7 "GND"
.TER     C1.2
         U2.7
         P1.4
.ADD_TER R1.1 "VCC"
.TER     U1.14
         U2.4
         U2.1
         U2.14
         P1.1
.ADD_TER U1.2 "N-4"
.TER     U2.3
.ADD_TER P1.2 "/SIG_OUT"
.TER     U2.5
         U2.2
.ADD_TER R1.2 "/CLOCK_IN"
.TER     C1.1
         U1.1
         P1.3
```

```
.END
```

15.3.3 创建 OrcadPCB2 网表文件

此格式只有一个部分是封装列表。每个封装包括其参考网络的焊盘列表。

以下是此特定转换的样式表:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--XSL style sheet to Eeschema Generic Netlist Format to CADSTAR netlist format
    Copyright (C) 2010, SoftPLC Corporation.
    GPL v2.

    How to use:
        https://lists.launchpad.net/kicad-developers/msg05157.html
-->

<!DOCTYPE xsl:stylesheet [
  <!ENTITY nl  "&#xd;&#xa;"> <!--new line CR, LF -->
]>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>

<!--
    Netlist header
    Creates the entire netlist
    (can be seen as equivalent to main function in C
-->
<xsl:template match="/export">
    <xsl:text>( { Eeschema Netlist Version 1.1  </xsl:text>
    <!-- Generate line .TIM <time> -->
<xsl:apply-templates select="design/date"/>
<!-- Generate line eeschema version ... -->
<xsl:apply-templates select="design/tool"/>
<xsl:text>}&nl;</xsl:text>

<!-- Generate the list of components -->
<xsl:apply-templates select="components/comp"/>  <!-- Generate list of components -->

<!-- end of file -->
<xsl:text>)&nl;*&nl;</xsl:text>
</xsl:template>

<!--
    Generate id in header like "eeschema (2010-08-17 BZR 2450)-unstable"
-->
```

```
<xsl:template match="tool">
    <xsl:apply-templates/>
</xsl:template>

<!--
    Generate date in header like "20/08/2010 10:45:33"
-->
<xsl:template match="date">
    <xsl:apply-templates/>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!--
    This template read each component
    (path = /export/components/comp)
    creates lines:
     ( 3EBF7DBD $noname U1 74LS125
       ... pin list ...
       )
    and calls "create_pin_list" template to build the pin list
-->
<xsl:template match="comp">
    <xsl:text> ( </xsl:text>
    <xsl:choose>
        <xsl:when test = "tstamp != '' ">
            <xsl:apply-templates select="tstamp"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>00000000</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "footprint != '' ">
            <xsl:apply-templates select="footprint"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>$noname</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text> </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "value != '' ">
            <xsl:apply-templates select="value"/>
        </xsl:when>
        <xsl:otherwise>
```

```
            <xsl:text>"~"</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
    <xsl:call-template name="Search_pin_list" >
        <xsl:with-param name="cmplib_id" select="libsource/@part"/>
        <xsl:with-param name="cmp_ref" select="@ref"/>
    </xsl:call-template>
    <xsl:text> )&nl;</xsl:text>
</xsl:template>

<!--
    This template search for a given lib component description in list
    lib component descriptions are in /export/libparts,
    and each description start at ./libpart
    We search here for the list of pins of the given component
    This template has 2 parameters:
        "cmplib_id" (reference in libparts)
        "cmp_ref"   (schematic reference of the given component)
-->
<xsl:template name="Search_pin_list" >
    <xsl:param name="cmplib_id" select="0" />
    <xsl:param name="cmp_ref" select="0" />
        <xsl:for-each select="/export/libparts/libpart">
            <xsl:if test = "@part = $cmplib_id ">
                <xsl:apply-templates name="build_pin_list" select="pins/pin">
                    <xsl:with-param name="cmp_ref" select="$cmp_ref"/>
                </xsl:apply-templates>
            </xsl:if>
        </xsl:for-each>
</xsl:template>


<!--
    This template writes the pin list of a component
    from the pin list of the library description
    The pin list from library description is something like
          <pins>
            <pin num="1" type="passive"/>
            <pin num="2" type="passive"/>
          </pins>
    Output pin list is ( <pin num> <net name> )
    something like
            ( 1 VCC )
            ( 2 GND )
-->
<xsl:template name="build_pin_list" match="pin">
    <xsl:param name="cmp_ref" select="0" />
```

```
    <!-- write pin numner and separator -->
    <xsl:text>  ( </xsl:text>
    <xsl:value-of select="@num"/>
    <xsl:text> </xsl:text>

    <!-- search net name in nets section and write it: -->
    <xsl:variable name="pinNum" select="@num" />
    <xsl:for-each select="/export/nets/net">
        <!-- net name is output only if there is more than one pin in net
             else use "?" as net name, so count items in this net
        -->
        <xsl:variable name="pinCnt" select="count(node)" />
        <xsl:apply-templates name="Search_pin_netname" select="node">
            <xsl:with-param name="cmp_ref" select="$cmp_ref"/>
            <xsl:with-param name="pin_cnt_in_net" select="$pinCnt"/>
            <xsl:with-param name="pin_num"> <xsl:value-of select="$pinNum"/>
            </xsl:with-param>
        </xsl:apply-templates>
    </xsl:for-each>

    <!-- close line -->
    <xsl:text> )&nl;</xsl:text>
</xsl:template>

<!--
    This template writes the pin netname of a given pin of a given component
    from the nets list
    The nets list description is something like
      <nets>
        <net code="1" name="GND">
          <node ref="J1" pin="20"/>
              <node ref="C2" pin="2"/>
        </net>
        <net code="2" name="">
          <node ref="U2" pin="11"/>
        </net>
    </nets>
    This template has 2 parameters:
        "cmp_ref"   (schematic reference of the given component)
        "pin_num"   (pin number)
-->

<xsl:template name="Search_pin_netname" match="node">
    <xsl:param name="cmp_ref" select="0" />
    <xsl:param name="pin_num" select="0" />
    <xsl:param name="pin_cnt_in_net" select="0" />
```

```
    <xsl:if test = "@ref = $cmp_ref ">
        <xsl:if test = "@pin = $pin_num">
        <!-- net name is output only if there is more than one pin in net
             else use "?" as net name
        -->
            <xsl:if test = "$pin_cnt_in_net>1">
                <xsl:choose>
                    <!-- if a net has a name, use it,
                        else build a name from its net code
                    -->
                    <xsl:when test = "../@name != '' ">
                        <xsl:value-of select="../@name"/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:text>$N-0</xsl:text><xsl:value-of select="../@code"/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:if>
            <xsl:if test = "$pin_cnt_in_net &lt;2">
                <xsl:text>?</xsl:text>
            </xsl:if>
        </xsl:if>
    </xsl:if>

</xsl:template>

</xsl:stylesheet>
```

这是 OrcadPCB2 输出文件。

```
( { Eeschema Netlist Version 1.1  29/08/2010 21:07:51
eeschema (2010-08-28 BZR 2458)-unstable}
 ( 4C6E2141 $noname P1 CONN_4
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 /CLOCK_IN )
  (  4 GND )
 )
 ( 4C6E20BA $noname U2 74LS74
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 N-04 )
  (  4 VCC )
  (  5 /SIG_OUT )
  (  6 ? )
  (  7 GND )
  (  14 VCC )
 )
```

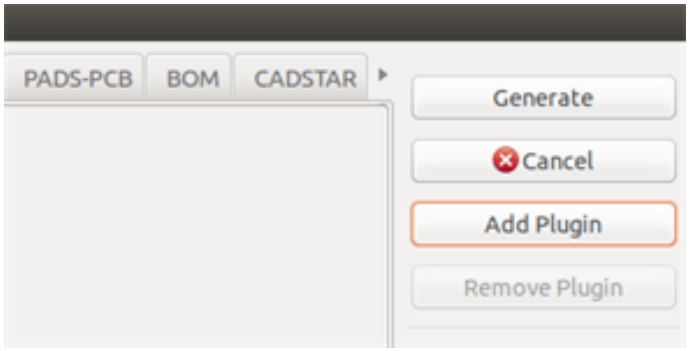
```
 ( 4C6E20A6 $noname U1 74LS04
  (  1 /CLOCK_IN )
  (  2 N-04 )
  (  7 GND )
  (  14 VCC )
 )
 ( 4C6E2094 $noname C1 CP
  (  1 /CLOCK_IN )
  (  2 GND )
 )
 ( 4C6E208A $noname R1 R
  (  1 VCC )
  (  2 /CLOCK_IN )
 )
)
*
```

15.3.4 Eeschema 插件界面

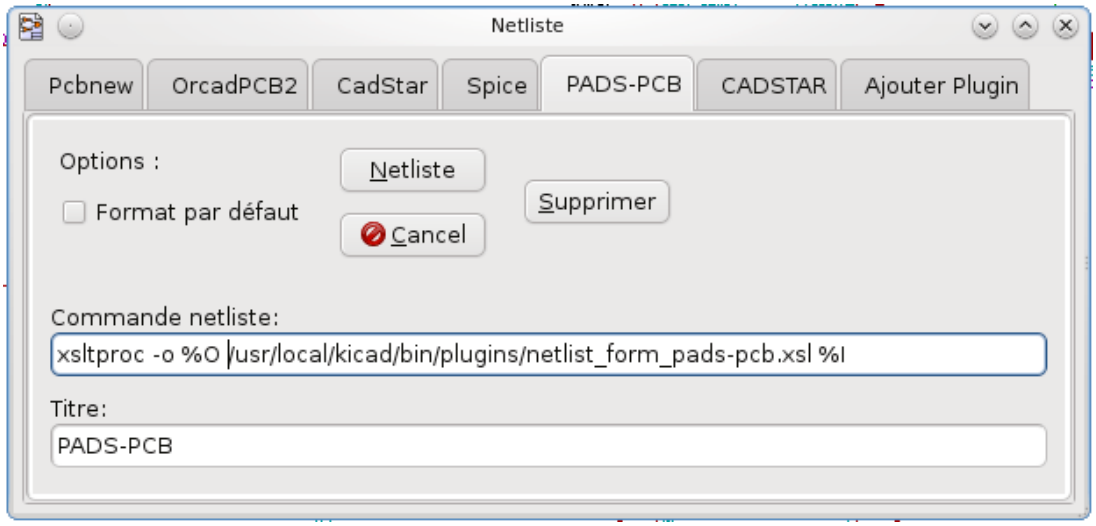
中间网表转换器可以在 Eeschema 中自动启动。

15.3.4.1 初始化对话窗口

可以通过单击添加插件按钮添加新的网表插件用户界面选项卡。



以下是 PadsPcb 选项卡的配置数据：



15.3.4.2 插件配置参数

Eeschema 插件配置对话框需要以下信息：

- 标题：例如，网表格式的名称。
- 用于启动转换器的命令行。

单击网表按钮后，将发生以下情况：

1. Eeschema 创建了一个中间网表文件 *.xml，例如 test.xml。
2. Eeschema 通过读取 test.xml 来运行插件，并创建 test.net。

15.3.4.3 使用命令行生成网络列表文件

假设我们使用程序 *xsltproc.exe* 将工作表样式应用于中间文件，则使用以下命令执行 *xsltproc.exe*：

xsltproc.exe -o <output filename> < style-sheet filename> <input XML file to convert>

在 Windows 下的 KiCad 中，命令行如下：

f:/kicad/bin/xsltproc.exe -o "%O" f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl "%I"

在 Linux 下，命令变为如下：

xsltproc -o "%O" /usr/local/kicad/bin/plugins/netlist_form_pads-pcb.xsl "%I"

netlist_form_pads-pcb.xsl 是您要应用的样式表。不要忘记文件名周围的双引号，这允许它们在 Eeschema 替换后有空格。

命令行格式接受文件名的参数:

支持的格式设置参数是。

- %B　基本文件名和所选输出文件的路径，减去路径和扩展名。
- %I　完整的文件名和临时输入文件的路径（中间网络文件）。

- %O　完整的文件名和用户选择的输出文件的路径。

%I 将被实际的中间文件名替换

%O 将替换为实际输出文件名。

15.3.4.4　命令行格式：xsltproc 的示例

xsltproc 的命令行格式如下：

<path of xsltproc> xsltproc <xsltproc parameters>

在 Windows 下：

f:/kicad/bin/xsltproc.exe -o "%O" f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl "%I"

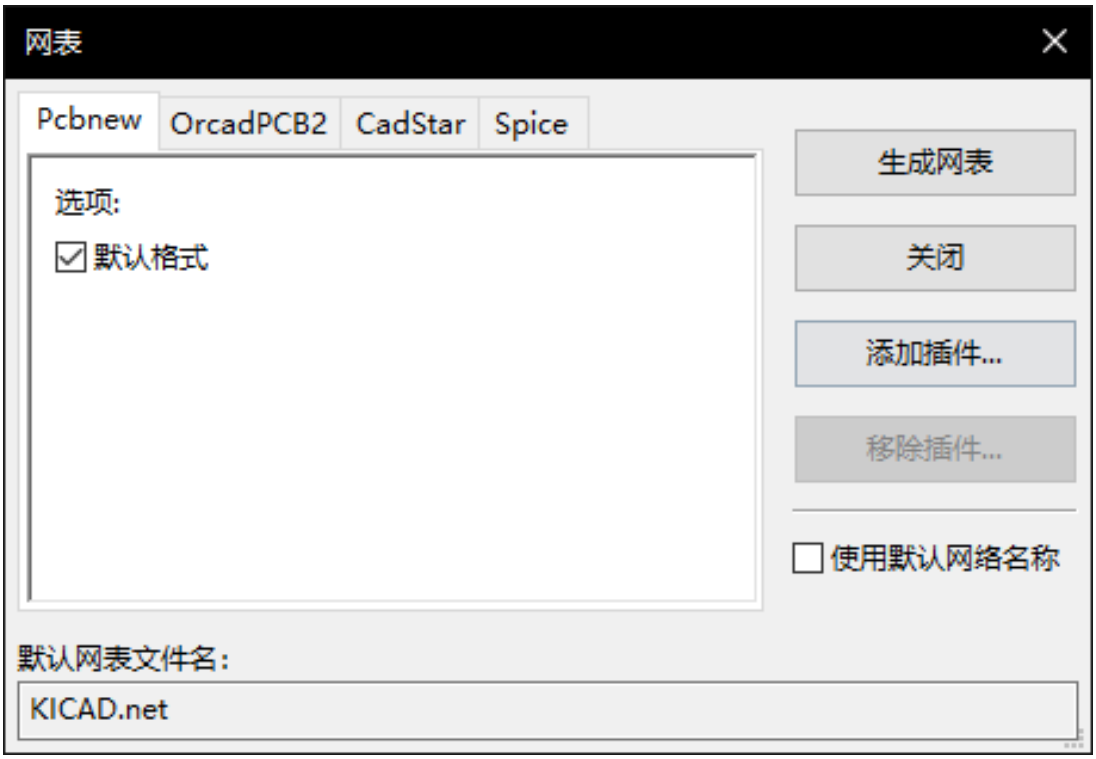
在 Linux 下：

xsltproc -o "%O" /usr/local/kicad/bin/plugins/netlist_form_pads-pcb.xsl "%I"

上面的示例假设 xsltproc 安装在 Windows 下的 PC 上，所有文件都位于 kicad/bin 中。

15.3.5　物料清单（BOM）生成

由于中间网表文件包含有关已使用元件的所有信息，因此可以从中提取 BOM。以下是用于创建自定义物料清单（BOM）文件的插件设置窗口（在 Linux 上）：



样式表 bom2csv.xsl 的路径取决于系统。目前用于 BOM 生成的最佳 XSLT 样式表称为 *bom2csv.xsl*。您可以根据自己的需要自由修改它，如果您开发了一些非常有用的东西，请让它成为 KiCad 项目的一部分。

15.4 命令行格式：python 脚本的示例

python 的命令行格式如下：

python < 脚本文件名 > < 输入文件名 > < 输出文件名 >

在 Windows 下：

python *.exe f:/kicad/python/my_python_script.py "%I" "%O"

在 Linux 下：

python /usr/local/kicad/python/my_python_script.py "%I" "%O"

假设你的 PC 上安装了 python。

15.5 中间网表结构

此示例提供了网表文件格式的概念。

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 21:07:51</date>
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2141</tstamp>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20BA</tstamp>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20A6</tstamp>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
```

```
      <tstamp>4C6E2094</tstamp>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E208A</tstamp>
    </comp>
  </components>
  <libparts/>
  <libraries/>
  <nets>
    <net code="1" name="GND">
      <node ref="U1" pin="7"/>
      <node ref="C1" pin="2"/>
      <node ref="U2" pin="7"/>
      <node ref="P1" pin="4"/>
    </net>
    <net code="2" name="VCC">
      <node ref="R1" pin="1"/>
      <node ref="U1" pin="14"/>
      <node ref="U2" pin="4"/>
      <node ref="U2" pin="1"/>
      <node ref="U2" pin="14"/>
      <node ref="P1" pin="1"/>
    </net>
    <net code="3" name="">
      <node ref="U2" pin="6"/>
    </net>
    <net code="4" name="">
      <node ref="U1" pin="2"/>
      <node ref="U2" pin="3"/>
    </net>
    <net code="5" name="/SIG_OUT">
      <node ref="P1" pin="2"/>
      <node ref="U2" pin="5"/>
      <node ref="U2" pin="2"/>
    </net>
    <net code="6" name="/CLOCK_IN">
      <node ref="R1" pin="2"/>
      <node ref="C1" pin="1"/>
      <node ref="U1" pin="1"/>
      <node ref="P1" pin="3"/>
    </net>
  </nets>
</export>
```

15.5.1 一般网表文件结构

中间网表占五个部分。

- “标题”部分。
- “元件”部分。
- “库元件”部分。
- “库”部分。
- “网”部分。

文件内容具有分隔符 <export>

```
<export version="D">
...
</export>
```

15.5.2 “标题”部分

标题具有分隔符 <design>

```
<design>
<source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
<date>21/08/2010 08:12:08</date>
<tool>eeschema (2010-08-09 BZR 2439)-unstable</tool>
</design>
```

此部分可被视为注释部分。

15.5.3 “元件”部分

元件部分具有分隔符 <components>

```
<components>
<comp ref="P1">
<value>CONN_4</value>
<libsource lib="conn" part="CONN_4"/>
<sheetpath names="/" tstamps="/"/>
<tstamp>4C6E2141</tstamp>
</comp>
</components>
```

本节包含原理图中的元件列表。每个元件都是这样描述的：

```
<comp ref="P1">
<value>CONN_4</value>
<libsource lib="conn" part="CONN_4"/>
<sheetpath names="/" tstamps="/"/>
<tstamp>4C6E2141</tstamp>
</comp>
```

libsource	找到此元件的库的名称。
part	此库中的组件名称。
sheetpath	层次结构中工作表的路径：标识工作表在完整的原理图层次结构中。
tstamps (time stamps)	原理图文件的时间戳。
tstamp (time stamp)	元件的时间戳。

15.5.3.1 关于元件的时间戳的注意事项

要识别网表中的元件，从而识别板上，时间戳参考对每个元件都是唯一的。然而，KiCad 提供了一种辅助方法来识别元件，该元件是电路板上相应的占位面积。这允许重新注释原理图项目中的元件，并且不会丢失元件与其占用空间之间的链接。

时间戳是原理图项目中每个元件或工作表的唯一标识符。但是, 在复杂的层次结构中, 同一工作表多次使用, 因此此工作表包含具有相同时间戳的元件。

复杂层次结构中的给定工作表具有唯一标识符：其 sheetpath。给定元件（在复杂层次结构内）具有唯一标识符: sheetpath + 其 tstamp

15.5.4 “库部件”部分

库部件部分具有分隔符 <libparts>，并且此部分的内容在原理图库中定义。库部件部分包含

- The allowed footprints names (names use wildcards) delimiter <fp>.
- 库分隔符中定义的字段 <fields>。
- 引脚列表分隔 <pins>。

```
<libparts>
<libpart lib="device" part="CP">
  <description>Condensateur polarise</description>
  <footprints>
    <fp>CP*</fp>
    <fp>SM*</fp>
  </footprints>
  <fields>
    <field name="Reference">C</field>
    <field name="Valeur">CP</field>
  </fields>
  <pins>
```

```
    <pin num="1" name="1" type="passive"/>
    <pin num="2" name="2" type="passive"/>
  </pins>
</libpart>
</libparts>
```

类似 <pin num=”1” type=”passive”/> 的线路也给出了电气引脚类型。可能的电气引脚类型有

Input	输入引脚
Output	输出引脚
Bidirectional	输入或输出
Tri-state	总线输入/输出
Passive	无源元件的结束
Unspecified	未知电气类型
Power input	单元件电源输入引脚
Power output	电源输出引脚作为稳压器输出
Open collector	模拟比较器中常见的开路集电极输出
Open emitter	有时在逻辑中找到开放发射器输出。
Not connected	必须在原理图中保持未连接状态

15.5.5 “库”部分

库部分具有分隔符 <libraries>。本节包含项目中使用的原理图库列表。

```
<libraries>
  <library logical="device">
    <uri>F:\kicad\share\library\device.lib</uri>
  </library>
  <library logical="conn">
    <uri>F:\kicad\share\library\conn.lib</uri>
  </library>
</libraries>
```

15.5.6 “网”部分

“网”部分具有分隔符 <nets>。本节包含原理图的“连接”。

```
<nets>
  <net code="1" name="GND">
    <node ref="U1" pin="7"/>
    <node ref="C1" pin="2"/>
    <node ref="U2" pin="7"/>
    <node ref="P1" pin="4"/>
  </net>
  <net code="2" name="VCC">
    <node ref="R1" pin="1"/>
```

```
    <node ref="U1" pin="14"/>
    <node ref="U2" pin="4"/>
    <node ref="U2" pin="1"/>
    <node ref="U2" pin="14"/>
    <node ref="P1" pin="1"/>
  </net>
</nets>
```

本节列出了原理图中的所有网络。

可能的网络包含以下内容。

```
<net code="1" name="GND">
  <node ref="U1" pin="7"/>
  <node ref="C1" pin="2"/>
  <node ref="U2" pin="7"/>
  <node ref="P1" pin="4"/>
</net>
```

net code	是此网络的内部标识符
name	是此网络的名称
node	给出一个连接到该网络的引脚引用

15.6 有关 xsltproc 的更多信息

请参阅页面：*http://xmlsoft.org/XSLT/xsltproc.html*

15.6.1 简介

xsltproc 是一个命令行工具，用于将 XSLT 样式表应用于 XML 文档。虽然它是作为 GNOME 项目的一部分开发的，但它可以独立于 GNOME 桌面运行。

从命令行调用 xsltproc，其中包含要使用的样式表的名称，后跟要应用样式表的文件的名称。如果提供的文件名是 - ，它将使用标准输入。

如果样式表包含在带有样式表处理指令的 XML 文档中，则不需要在命令行中命名样式表。xsltproc 将自动检测包含的样式表并使用它。默认情况下，输出为 *stdout*。您可以使用 -o 选项指定要输出的文件。

15.6.2 简介

```
xsltproc [[-V] | [-v] | [-o *file* ] | [--timing] | [--repeat] |
[--debug] | [--novalid] | [--noout] | [--maxdepth *val* ] | [--html] |
[--param *name* *value* ] | [--stringparam *name* *value* ] | [--nonet] |
[--path *paths* ] | [--load-trace] | [--catalogs] | [--xinclude] |
[--profile] | [--dumpextensions] | [--nowrite] | [--nomkdir] |
[--writesubtree] | [--nodtdattr]] [ *stylesheet* ] [ *file1* ] [ *file2* ]
```

```
[ *....* ]
```

15.6.3 命令行选项

-V o *--version*

显示使用的 libxml 和 libxslt 的版本。

-v o *--verbose*

输出 xsltproc 在处理样式表和文档时采取的每个步骤。

-o o *--output file*

直接输出到名为 *file* 的文件。对于多个输出，也称为 *chunking*，-o directory/ 将输出文件定向到指定的目录。该目录必须已存在。

--timing

显示用于解析样式表，解析文档和应用样式表并保存结果的时间。以毫秒显示。

--repeat

运行转换 20 次。用于定时测试。

--debug

输出转换后文档的 XML 树，以进行调试。

--novalid

跳过加载文档的 DTD。

--noout

不输出结果。

--maxdepth value

在 libxslt 断定它处于无限循环之前调整模板堆栈的最大深度。默认值为 500。

--html

输入文档是 HTML 文件。

--param name value

将名称 *name* 和值 *value* 的参数传递给样式表。您可以传递多个名称/值对，最多为 32. 如果传递的值是字符串而不是节点标识符，请改用 --stringparam。

--stringparam name value

传递名称 *name* 和值 *value* 的参数，其中 *value* 是字符串而不是节点标识符。(注意：字符串必须是 utf-8。)

--nonet

不要使用互联网来获取 DTD，实体或文档。

--path paths

使用 *paths* 指定的文件系统路径的列表（由空格或列分隔）来加载 DTD，实体或文档。

--load-trace

向 stderr 显示处理期间加载的所有文档。

--catalogs

使用 SGML_CATALOG_FILES 中指定的 SGML 目录来解析外部实体的位置。默认情况下，xsltproc 查找 XML_CATALOG_FILES 中指定的目录。如果未指定，则使用 /etc/xml/catalog。

--xinclude

使用 Xinclude 规范处理输入文档。有关这方面的更多详细信息，请参阅 Xinclude 规范：http://www.w3.org/TR/-xinclude/

--profile --norman

输出分析信息，详细说明样式表的每个部分所花费的时间。这在优化样式表性能时很有用。

--dumpextensions

将所有已注册扩展名的列表转储到 stdout。

--nowrite

拒绝写入任何文件或资源。

--nomkdir

拒绝创建目录。

--writesubtree path

仅允许在 *path* 子树内写入文件。

--nodtdattr

不要从文档的 DTD 应用默认属性。

15.6.4 Xsltproc 返回值

xsltproc 返回一个状态编号，在脚本中调用它时非常有用。

0: 正常

1: 无参数

2: 参数太多

3: 未知选项

4：无法解析样式表

5: 样式表中的错误

6：其中一个文件出错

7: 不支持的 xsl: 输出方法

8：字符串参数包含引号和双引号

9: 内部处理错误

10：通过终止消息停止处理

11：无法将结果写入输出文件

15.6.5 有关 xsltproc 的更多信息

libxml 网页：http://www.xmlsoft.org/

W3C XSLT 页面：http://www.w3.org/TR/xslt

Chapter 16

仿真器

Eeschema 使用 ngspice 作为模拟引擎提供嵌入式电路仿真器。

使用模拟器时，您可能会发现官方的 *pspice* 库很有用。它包含用于模拟的公共符号，如电压/电流源或晶体管，其引脚编号与 ngspice 节点顺序规范相匹配。

还有一些演示项目来说明模拟器的功能。您将在 *demos/simulation* 目录中找到它们。

16.1 分配模型

在启动模拟之前，元件需要分配 Spice 模型。

即使元件由多个单元组成，每个元件也只能分配一个模型。在这种情况下，第一个单元应该具有指定的模型。

”无源模型”参考匹配 Spice 表示法中的器件类型的无源元件（$R*$ 表示电阻器，$C*$ 表示电容器，$L*$ 表示电感器）将隐式分配模型并使用值字段确定他们的属性。

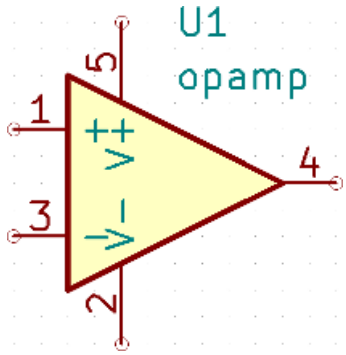
Note

请记住，在 Spice 表示法中，“M”代表 milli，“Meg”代表 mega。如果您更喜欢使用“M”来表示超级前缀，您可以在（模拟设置，模拟设置对话框）中请求这样做。

Spice 模型信息作为文本存储在符号字段中，因此您可以在符号编辑器或原理图编辑器中定义它。打开符号属性对话框，然后单击 *编辑 Spice 模型*按钮以打开 Spice 模型编辑器对话框。

Spice 模型编辑器对话框有三个对应于不同模型类型的选项卡。所有模型类型共有两个选项：

禁用模拟的符号	选中时，元件将从模拟中排除。

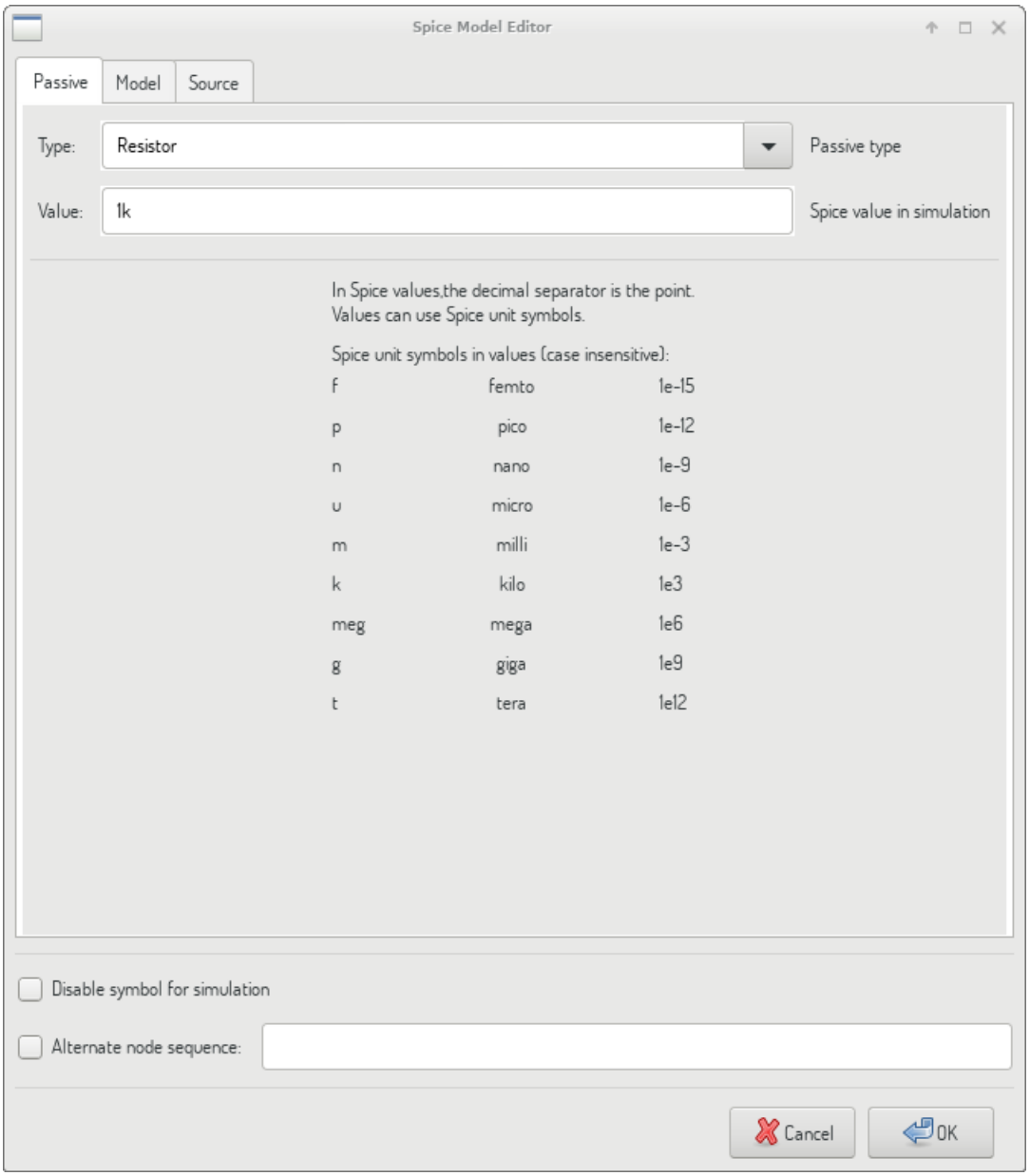
备用节点序列	允许用户将符号引脚覆盖为模型节点映射。要定义不同的映射，请按模型预期的顺序指定引脚编号。 例子：+ “* 连接：” + “* 1: 非反相输入” “* 2: 反相输入” “* 3: 正电源” “* 4: 负电源” “* 5: 输出” “. 子电路 tl071 1 2 3 4 5”  要将符号引脚与上面显示的 Spice 模型节点相匹配，需要使用具有值的备用节点序列选项：“1 3 5 2 4”。它是与 Spice 模型节点顺序对应的引脚编号列表。

16.1.1 无源

无源选项卡允许用户将无源器件模型（电阻，电容或电感）分配给元件。这是一个很少使用的选项，因为通常被动元件的模型分配了 模拟无源模型，隐形，除非元件引用与实际设备类型不匹配。

Note

明确定义的被动设备模型优先于隐式分配的模型。这意味着一旦分配了被动设备模型，在模拟期间不会考虑参考和值字段。当指定的模型值与原理图纸上显示的模型值不匹配时，可能会导致混乱的情况。

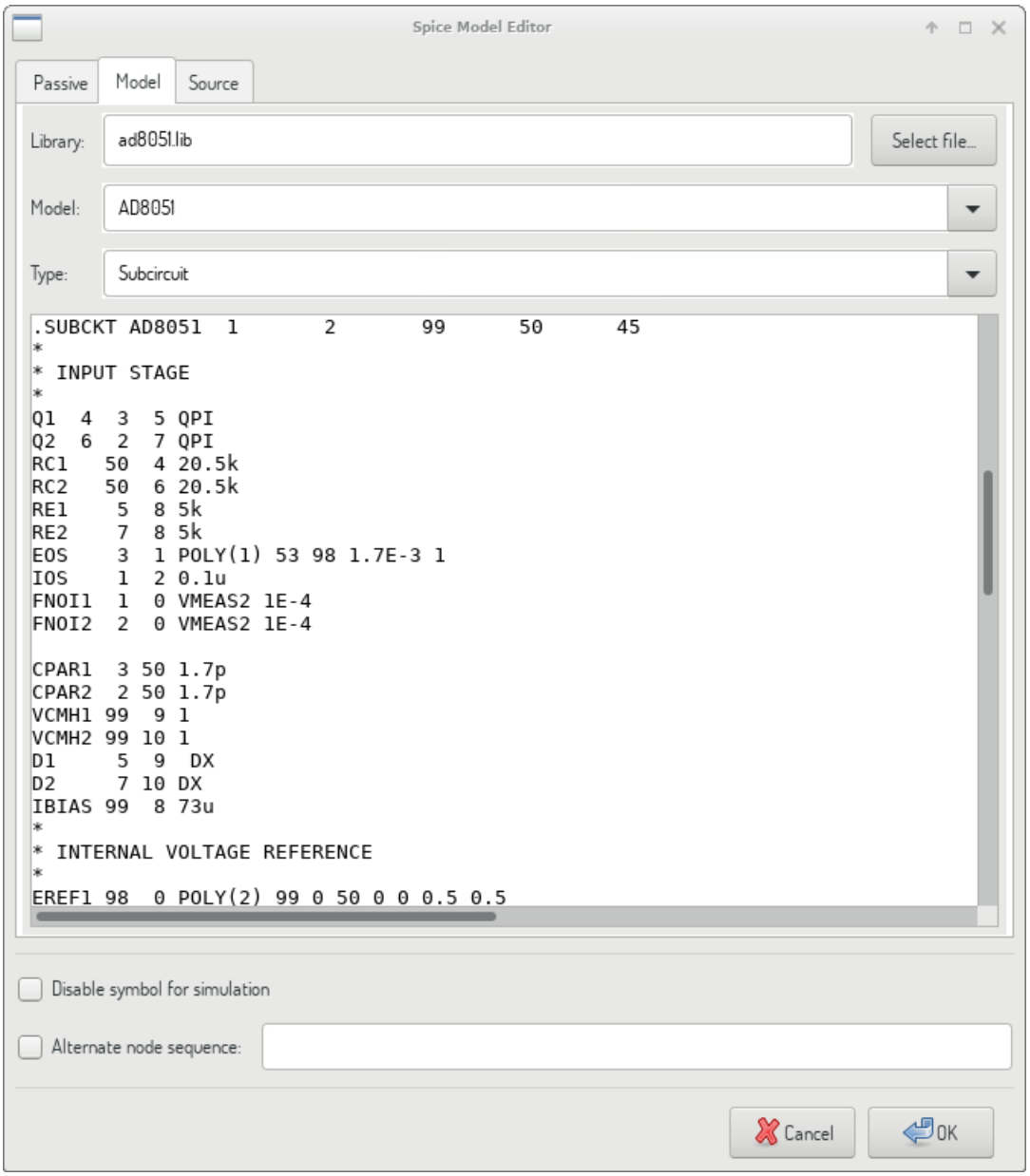


类型	选择器件类型（电阻，电容或电感）。
值	定义器件属性（电阻，电容或电感）。值可以使用常见的 Spice 单元前缀（如文本输入字段下方列出的）和应该使用点作为小数点分隔符。请注意，Spice 不正确解释在值中交织的前缀（例如 1k5）。

16.1.2 模型

Model 选项卡用于分配外部库文件中定义的半导体或复杂模型。Spice 模型库通常由设备制造商提供。

主文本小部件显示所选的库文件内容。将模型描述放在库文件中是常见的做法，包括节点顺序。

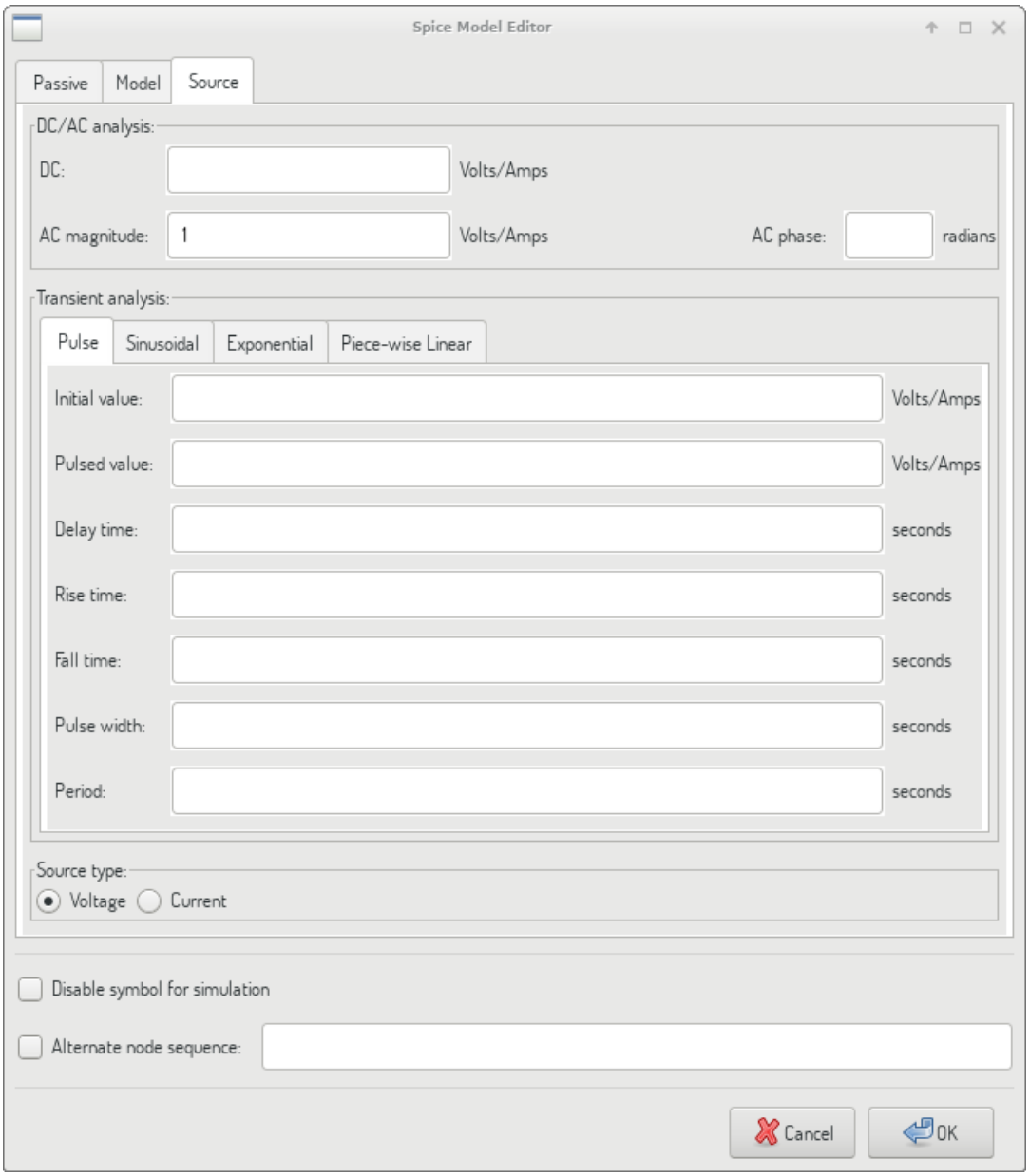


文件	Spice 库文件的路径。该文件将由模拟器使用，因为它是使用*.include* 指令添加的。
型号	所选设备型号。选择文件后，列表将填充可用模型可供选择。
类型	选择型号类型（子电路，BJT，MOSFET 或二极管）。通常是设定的选择模型时自动选择。

16.1.3 源

Source 选项卡用于分配电源或信号源模型。有两个部分：“DC/AC 分析”和“转换分析”。每个都定义了相应模拟类型的源参数。

Source type 选项适用于所有模拟类型。



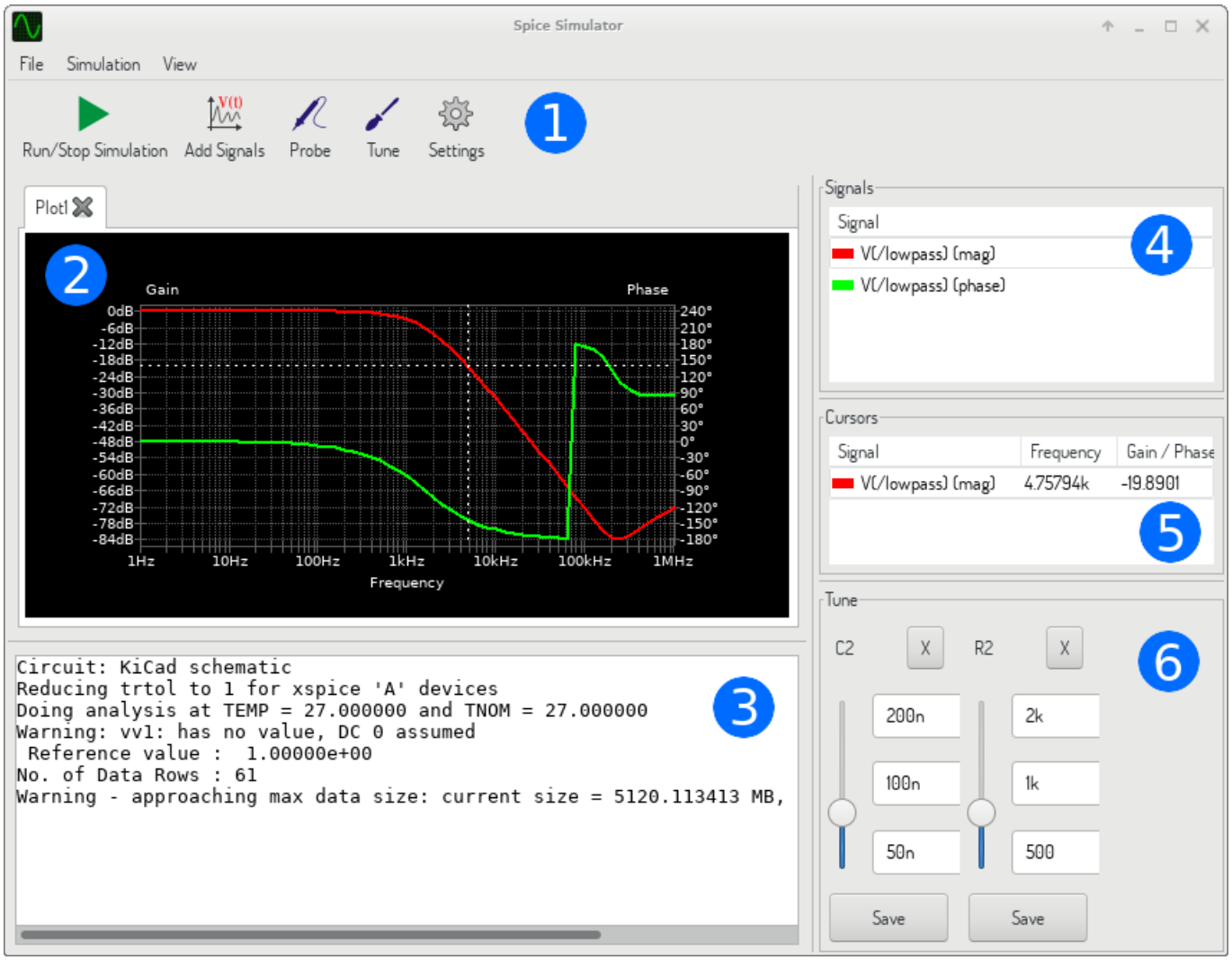
有关源的更多详细信息，请参阅 ngspice 文档，第 4 章（电压和电流源）。

16.2 Spice 指令

可以通过将 Spice 指令放在原理图工作表的文本字段中来添加它们。此方法便于定义默认模拟类型。此功能仅限于以点开头的 Spice 指令（例如 ".tran 10n 1m"），无法使用文本字段放置其他元件。

16.3 仿真

要启动模拟，请在原理图编辑器窗口中选择菜单 *工具* → *仿真*打开 *Spice 仿真*对话框。



该对话框分为几个部分：

- 《模拟-工具栏，工具栏》
- 《模拟面板，绘图面板》
- 《模拟输出控制台，输出控制台》
- 《模拟信号列表，信号列表》
- 《模拟游标列表，游标列表》
- 《模拟面板，绘图面板》

16.3.1 菜单

16.3.1.1 文件

新绘制	在绘图面板中创建一个新选项卡。
打开工作簿	打开绘制信号列表。
保存工作簿	保存绘制信号列表。
另存为图像	将活动图导出为.png 文件。
另存为.csv 文件	将活动绘图原始数据点导出到.csv 文件。
退出模拟	关闭对话框。

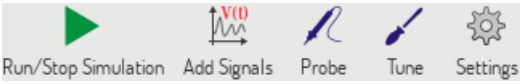
16.3.1.2 仿真

运行模拟	使用当前设置执行模拟。
添加信号……	打开一个对话框以选择要绘制的信号。
原理图探测	启动原理图“模拟探针工具，探针”工具。
调整元件值	启动“模拟调谐工具，调谐”工具。
显示 SPICE 网表…	打开一个对话框，显示生成的网表模拟电路。
设置…	打开“模拟设置，模拟设置对话框”。

16.3.1.3 视图

缩小	缩小活动图。
适合屏幕	调整缩放设置以显示所有绘图。
显示网格	切换网格可见性。
显示长度	切换图表图例可见性。

16.3.2 工具栏



顶部工具栏提供对最常执行的操作的访问。

运行/停止模拟	启动或停止模拟。
添加信号	打开一个对话框以选择要绘制的信号。
探针	启动原理图“模拟探针工具，探针”工具。
调谐	启动”模拟调谐工具，调谐“工具。
设置	打开“模拟设置，模拟设置对话框”。

16.3.3 绘图面板

将模拟结果可视化为图。可以在单独的选项卡中打开多个图，但只有在执行模拟时才会更新活动图。这样就可以比较不同运行的模拟结果。

可以使用“模拟菜单视图，视图”菜单切换网格和图例可见性来自定义绘图。当图例可见时，可以拖动它来改变其位置。

绘图面板交互:

- 滚动鼠标滚轮放大/缩小
- 右键单击打开上下文菜单以调整视图
- 绘制选择矩形以放大所选区域
- 拖动光标以更改其坐标

16.3.4 输出控制台

输出控制台显示来自模拟器的消息。建议检查控制台输出以确认没有错误或警告。

16.3.5 信号列表

显示活动图中显示的信号列表。

信号列表交互:

- 右键单击打开上下文菜单以隐藏信号或切换光标
- 双击以隐藏信号

16.3.6 游标列表

显示游标列表及其坐标。每个信号可以显示一个光标。使用“模拟信号列表，信号”列表设置游标可见性。

16.3.7 调谐面板

显示使用“模拟调谐工具，调谐“工具选取的元件。调谐面板允许用户快速修改元件值并观察它们对模拟结果的影响 - 每次更改元件值时，都会重新运行模拟并更新图形。

对于每个元件，有一些控件关联:

- 顶部文本字段设置最大元件值。
- 中间文本字段设置实际的元件值。
- 底部文本字段设置最小元件值。
- 滑块允许用户以平滑的方式修改元件值。
- *Save* 按钮将原理图上的元件值修改为使用滑块选择的元件值。
- *X* 按钮从调谐面板中删除元件并恢复其原始值。

三个文本字段识别 Spice 单元前缀。

16.3.8 调谐工具

调谐器工具允许用户选择要调整的元件。

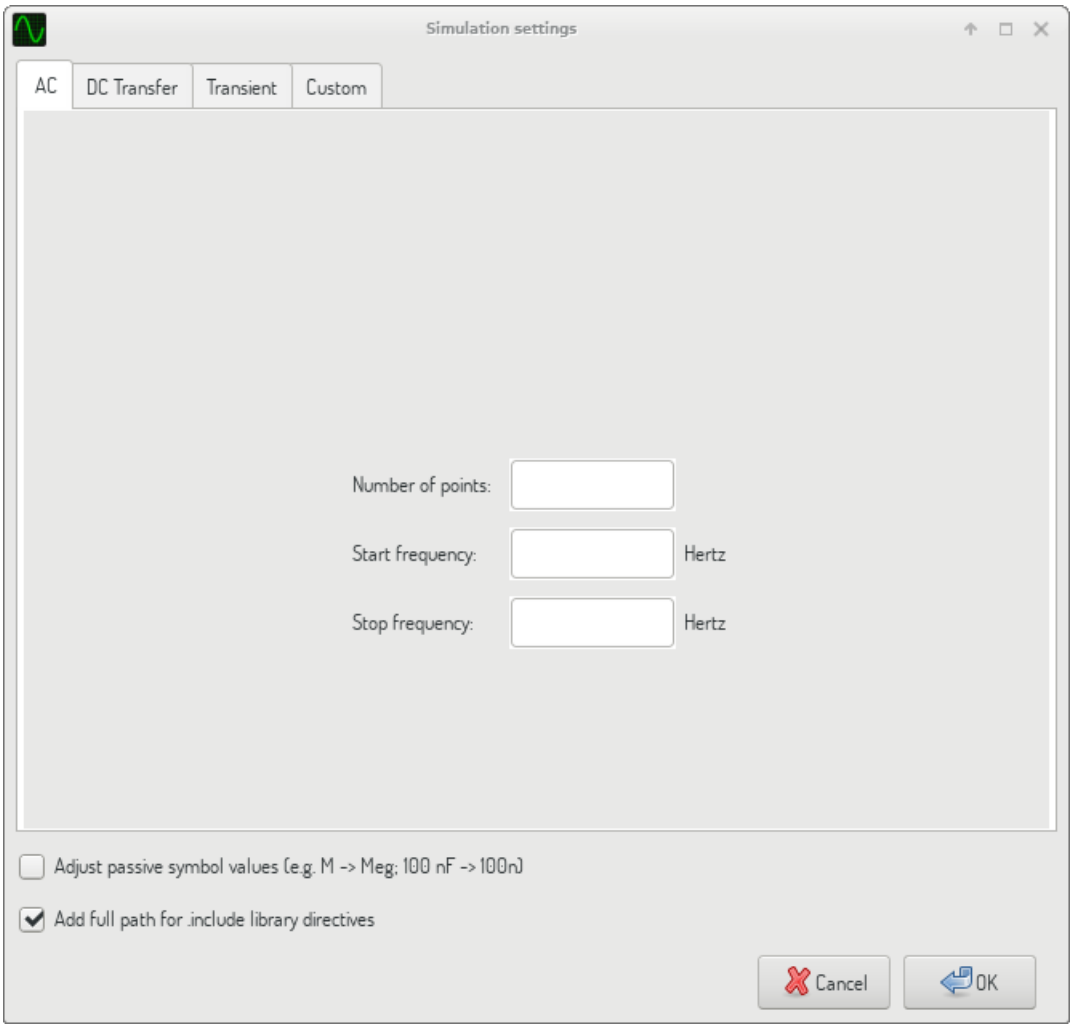
要选择要调整的元件，请在工具处于活动状态时单击原理图编辑器中的一个元件。所选元件将出现在“模拟调谐工具，调谐”面板中。只能调整被动元件。

16.3.9 探针工具

探针工具提供了一种用户友好的方式来选择用于绘图的信号。

要向绘图添加信号，请在工具处于活动状态时单击原理图编辑器中的相应导线。

16.3.10 仿真设置



模拟设置对话框允许用户设置模拟类型和参数。有四个选项卡：

- 交流
- 直流转换

- 短暂的
- 自定义

前三个选项卡提供可以指定模拟参数的表单。最后一个选项卡允许用户键入自定义 Spice 指令以设置模拟。有关仿真类型和参数的更多信息，请参见 ngspice 文档 ，第 1.2 章。

配置模拟的另一种方法是在原理图上的文本字段中键入“模拟指令，Spice 指令”。与模拟类型相关的任何文本字段指令都会被对话框中选择的设置覆盖。这意味着一旦开始使用模拟对话框，该对话框将覆盖原理图指令，直到重新打开模拟器。

所有模拟类型共有两个选项：

调整被动符号值	替换被动符号值以转换常见元件值符号表示 Spice 表示法。
为.include 库指令添加完整路径	Prepend Spice 模型库文件名为完整路径。通常，ngspice 需要完整路径才能访问库文件。