

# PCB Editor

The KiCad Team

# Table of Contents

Introduction to the KiCad PCB Editor .....	2
The PCB Editor user interface .....	3
Navigating the editing canvas .....	3
Hotkeys .....	4
Display and selection controls .....	5
Board layers .....	5
The appearance panel .....	5
Selection and the selection filter .....	7
Net highlighting .....	9
Cross-probing from the schematic .....	9
Left toolbar display controls .....	10
Creating a PCB .....	12
Basic PCB concepts .....	12
Capabilities .....	12
Starting from a schematic .....	12
Starting from scratch .....	13
Board setup .....	14
Configuring board stackup and physical parameters .....	14
Configuring text and graphics .....	18
Configuring design rules .....	20
Embedding files .....	33
Importing settings .....	34
Editing a board .....	36

Placement and drawing operations .....	36
Grids and snapping .....	38
Editing object properties .....	41
Board outlines (Edge Cuts) .....	43
Working with footprints .....	43
Working with pads .....	54
Working with zones .....	55
Routing tracks and vias .....	60
Routing differential pairs .....	77
Length tuning .....	77
Teardrops .....	86
Backdrills and hole post-machining (counterbores/countersinks) .....	89
Graphics and text .....	93
Rule areas (keepouts) .....	114
Locking .....	117
Groups .....	117
Aligning objects .....	119
Distributing objects .....	120
Arrays .....	121
Multichannel layout .....	124
Using reference images .....	139
Forward and back annotation .....	141
Update PCB From Schematic (forward annotation) .....	141
Update Schematic from PCB (back annotation) .....	145
Geographical re-annotation .....	147
Inspecting a board .....	151
Design rules checking .....	151
Board Statistics .....	167
Measurement tool .....	169
Find tool .....	170
Search panel .....	171
3D Viewer .....	172
Net inspector .....	175
Importing boards from other EDA tools .....	178
Supported import formats .....	178
How to import a board .....	178
Layer mapping .....	179
Post-import cleanup .....	179
Format-specific notes .....	180
Importing Spectra session files .....	185
Generating outputs .....	186

Plotting (Gerber / PostScript / SVG / DXF / PDF)	186
Drill files	189
IPC-2581	191
ODB++	192
Component placement (position) files	192
Footprint reports	194
IPC-D-356 netlists	194
Bill of Materials	194
Printing	194
Specctra DSN	196
GenCAD	196
VRML	196
IDF	198
3D models (STEP / GLB / BREP / XAO / PLY / STL / STPZ / U3D / PDF)	199
Footprint association (CMP) files	202
Hyperlynx	202
Footprints and footprint libraries	203
Managing footprint libraries	203
Browsing footprint libraries	207
Creating and editing footprints	209
Footprint editor overview	209
Browsing, modifying, and saving footprints	214
Creating a new footprint library	214
Creating a new footprint	214
Editing footprint properties	215
Footprint pads	217
Footprint graphics and text	228
Footprint fields	231
Footprint layers	232
Clearance overrides	234
Pad connections, net ties, and jumper pads	235
3D models	238
Embedding files	239
Rule areas	240
Reference images	241
Footprint wizards	241
Checking footprints	242
Design blocks	244
Creating and managing design blocks	245
Using design blocks in a board	247
Grouped design blocks	248
Advanced topics	250

Configuration and Customization .....	250
Text variables .....	255
Text expressions .....	257
Custom design rules .....	264
Scripting .....	0
IDF component outlines .....	0
Actions reference .....	0
PCB Editor .....	0
3D Viewer .....	0
Common .....	0

## *KiCad 10.0 Reference Manual*

### NOTE

This manual is in the process of being revised to cover the latest stable release version of KiCad. It contains some sections that have not yet been completed. We ask for your patience while our volunteer technical writers work on this task, and we welcome new contributors who would like to help make KiCad's documentation better than ever.

### Copyright

This document is Copyright The KiCad Documentation Contributors. You may distribute it and/or modify it under the terms of either the GNU General Public License (<http://www.gnu.org/licenses/gpl.html>), version 3 or later, or the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), version 3.0 or later.

All trademarks within this guide belong to their legitimate owners.

### Contributors

Jean-Pierre Charras, Fabrizio Tappero, Wayne Stambaugh, Cirilo Bernardo, Jon Evans, Graham Keeth

### Feedback

The KiCad project welcomes feedback, bug reports, and suggestions related to the software or its documentation. For more information on how to submit feedback or report an issue, please see the instructions at <https://www.kicad.org/help/report-an-issue/>

### Software and Documentation Version

This user manual is based on KiCad 10.0.0. Functionality and appearance may be different in other versions of KiCad.

Documentation revision: 90e4b228 .

# Introduction to the KiCad PCB Editor

The KiCad PCB Editor is a PCB layout application distributed as a part of KiCad and available for the following operating systems:

- Linux
- Apple macOS
- Windows

Regardless of the OS, all KiCad files are 100% compatible from one OS to another.

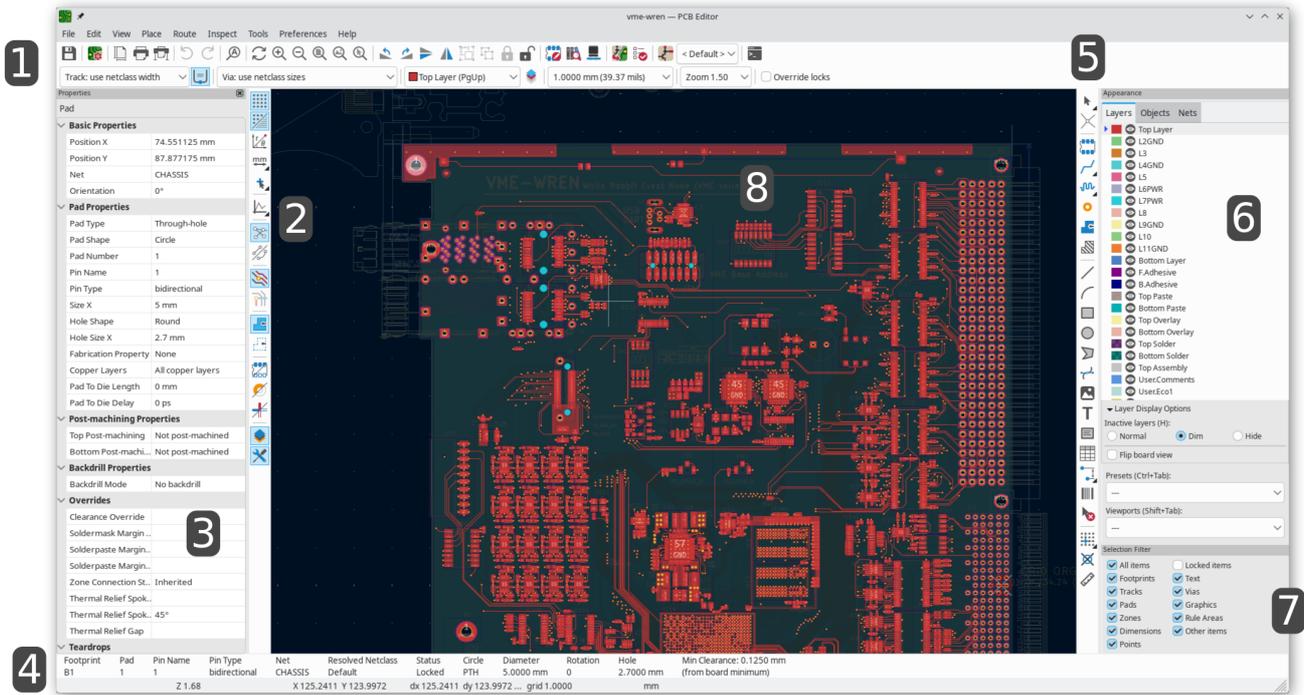
The PCB Editor is an integrated application where all functions of placing footprints, routing tracks, library management, and data transfer to and from the schematic capture software are carried out within the editor itself.

The KiCad PCB Editor is intended to communicate directly with the KiCad Schematic Editor for designing printed circuit boards from schematics without using any intermediate files. It can also import netlist files, which list all the electrical connections, from other packages.

The PCB Editor includes a footprint library editor, which can create and edit footprints and manage libraries. It also integrates the following additional but essential functions needed for modern PCB design software:

- Design rules check (DRC) for automatic detection of design rule violations such as incorrect and missing connections, copper clearance and minimum width violations, and many other design issues
- Scriptable design rules for specifying rules with complex constraints and conditions
- An interactive router with multiple modes of operation (push-and-shove, walkaround, highlight collisions) and support for differential pair routing as well as length and skew tuning
- Export of fabrication and plot files in many formats (Gerber, IPC-2581, ODB++, GenCAD, PDF, PostScript, and SVG)
- A 3D viewer and 3D model generation in many formats (STEP, GLB, BREP, XAO, PLY, STL, IDF, and VRML)

# The PCB Editor user interface



The main PCB Editor user interface is shown above, with some key elements indicated:

1. Top toolbars (file management, zoom tools, editing tools)
2. [Left toolbar \(display options\)](#)
3. [Properties panel](#)
4. Message panel and status bar
5. [Right toolbar \(drawing and design tools\)](#)
6. [Appearance panel](#)
7. [Selection filter panel](#)
8. Editing canvas

## Navigating the editing canvas

The editing canvas is a view onto the board being designed. You can pan and zoom to different areas of the board, and also flip the view to show the board from the bottom.

By default, dragging with the middle or right mouse button will pan the canvas view and scrolling the mouse wheel will zoom the view in or out. You can change this behavior in the Mouse and Touchpad section of the preferences (see [Configuration and Customization](#) for details).

Several other zoom tools are available in the top toolbar:

- zooms in on the center of the viewport.
- zooms out from the center of the viewport.
- zooms to fit the frame around the drawing sheet.

-  zooms to fit the items within the drawing sheet.
-  allows you to draw a box to determine the zoomed area.

The cursor's current position is displayed at the bottom of the window (X and Y), along with the current zoom factor (Z), the cursor's relative position (dx, dy, and dist), the grid setting, and the display units.

The relative coordinates can be reset to zero by pressing `Space`. This is useful for measuring distance between two points or aligning objects.

## Hotkeys

The `Ctrl` + `F1` shortcut displays the current hotkey list. The default hotkey list is included in the [Actions Reference](#) section of the manual.

The hotkeys described in this manual use the key labels that appear on a standard PC keyboard. On an Apple keyboard layout, use the `Cmd` key in place of `Ctrl`, and the `Option` key in place of `Alt`.

Many actions do not have hotkeys assigned by default, but hotkeys can be assigned or redefined using the hotkey editor (**Preferences** → **Preferences...** → **Hotkeys**).

### NOTE

Many of the actions available through hotkeys are also available in context menus. To access the context menu, right-click in the editing canvas. Different actions will be available depending on what is selected or what tool is active.

Hotkeys are stored in the file `user.hotkeys` in KiCad's configuration directory. The location is platform-specific:

- Windows: `%APPDATA%\kicad\10.0\user.hotkeys`
- Linux: `~/.config/kicad/10.0/user.hotkeys`
- macOS: `~/Library/Preferences/kicad/10.0/user.hotkeys`

KiCad can import hotkey settings from a `user.hotkeys` file using the **Import Hotkeys** button in the hotkey editor.

# Display and selection controls

## Board layers

Layers in the PCB Editor represent physical copper layers on a board, as well as graphical layers used for defining things such as silkscreen, solder mask, and the board edge. There is always one layer that is active in the editor. The active layer is drawn on top of other layers and will be the layer assigned to newly-created objects. The active layer is indicated in the layer selector drop-down box in the top toolbar and is also highlighted in the appearance panel. To change the active layer, you can left-click a layer name in the appearance panel, use the drop-down layer selector in the top toolbar, or use a hotkey. Layers can be hidden to simplify the board view. You can hide a layer even if it is the active layer.

## Display order for board layers

The display order for board layers is dynamic and depends on which layer is selected as the active layer. The active layer is always drawn on top of other layers. In addition, layers that are related to the active layer are drawn on top of layers that are unrelated. For example, if you make B.Silkscreen the active layer, then all of the other back layers (B.Cu, B.Adhesive, B.Paste, B.Mask, B.Fab, and B.Courtyard) will be drawn on top of the front, user, and inner copper layers, with B.Silkscreen topmost. If you make Edge.Cuts active, then it will be drawn on top, and the User.\* layers and Margin will also be brought to the front.

**NOTE** | Selected objects are always drawn on top, even if they are not on the active layer.

## The appearance panel

The appearance panel provides controls to manage the visibility, color, and opacity of objects in the PCB Editor's drawing canvas. It has three tabs: the Layers tab contains controls for the board layers, the Objects tab contains controls for different types of graphical objects, and the Nets tab contains controls for the appearance of the ratsnest and copper items.

## Layer controls

In the Layers tab of the appearance panel, each board layer is shown with its color and visibility state. The active layer is shown highlighted with an arrow indicator to the left of the color swatch. Left-click on a layer to choose it as the active layer. Left-click on the corresponding visibility icon to toggle the layer between visible and hidden. Double-click or middle-click on the color swatch to change the layer's color.

**NOTE** | You must first create a custom color theme in Preferences before you can change layer colors in the appearance panel.

Below the list of layers is an expandable panel that contains layer display options. The first setting controls how non-active layers are displayed: normal, dimmed, or hidden. The layer display mode can be used to simplify the view and focus on a single layer. Items on inactive layers cannot be selected when the non-active layer display mode is "Dim" or "Hide". You can use the hotkey **Ctrl + H** to cycle through these display modes quickly.

**Flip board view** will show the board as if you are looking from the bottom (that is, mirrored around the Y-axis). This option is also available in the View menu.

**NOTE**

Flipping the board view does not change the visual layer ordering, the active layer will remain in front followed by the other layers in their normal order.

## Object controls

The Objects tab of the appearance panel is similar to the Layers tab. The main differences are that some objects have no color setting and that four types of objects (tracks, vias, pads, and zones) have opacity control sliders. The opacity setting here will be multiplied with any opacity set in the layer colors. By default, all objects are fully opaque except for zones, which are set to translucent in order to make it easier to see objects through filled zone areas.

## Layer presets

Layer presets store which layers and objects are visible and hidden for easy recall. There are several built-in layer presets and you can save your own custom presets. Custom presets are stored in the project settings for a board, as presets may be specific to a certain board stackup.

To load a preset, choose it from the Presets drop-down menu at the bottom of the appearance panel or use the quick switcher by holding down **Ctrl** and pressing **Tab**. Once the quick switcher window appears, you can press **Tab** and **Shift** + **Tab** to cycle through the available presets. When you let go of the **Ctrl** key, the highlighted preset will be loaded.

To save a custom preset, first use the visibility controls to choose which layers you want visible, then choose **Save preset...** from the Presets drop-down menu. Give your preset a name and it will now be available via the drop-down menu and the quick switcher. To modify a custom preset, follow the same process and save the modified version with the same name to overwrite the existing version. To delete a custom preset, choose the **Delete preset...** option from the drop-down menu and select the preset to be deleted from the list.

## Viewports

Viewports store the current view location and zoom level so you can quickly switch back to it later, or switch between several saved views.

To load a viewport, choose it from the Viewports drop-down menu at the bottom of the appearance panel or use the quick switcher by holding down **Shift** and pressing **Tab**. Once the quick switcher window appears, you can press **Tab** to cycle through the stored viewports. When you let go of the **Shift** key, the highlighted viewport will be loaded.

To save a new viewport, scroll and zoom to show the desired area of the board, then choose **Save viewport...** from the Viewports drop-down menu. Give your viewport a name and it will now be available via the drop-down menu and the quick switcher. To modify an existing viewport, save a new viewport with the same name to overwrite the existing version. To delete a viewport, choose the **Delete viewport...** option from the drop-down menu and select the preset to be deleted from the list.

## Net and net class controls

The Nets tab of the appearance panel shows a list of all nets and net classes in the board. Each net has a visibility control that controls the visibility of that net in the ratsnest. Hiding nets in the ratsnest does not change the connectivity of the board and will not impact the design rule checker; it only is intended to make the ratsnest easier to understand.

Each net and net class can also have a color assigned. By default, this color applies to the ratsnest lines for the net (or for all the nets in the net class). Nets have no color by default; this is indicated by a checkerboard pattern in the color swatch. Double-click or right-click a net or net class color swatch to set the color. To give a net class the same color it has in the schematic, right click the net class and select **Use color from schematic**.

**NOTE**

The Default net class cannot have a color assigned, as nets in this class will just use the default ratsnest color defined by the color theme.

You can also select and highlight nets and net classes via the appearance panel: right-click on a net or net class to show these options in a menu.

Below the list of net classes is an expandable panel that contains net display options. The first option controls how net colors are applied. When "All" is selected, all copper items (pads, tracks, vias, and zones) belonging to a net or net class will take on the chosen color. When "Ratsnest" is selected (the default value), only the ratsnest is affected by net and net class colors. When "None" is selected, net and net class colors are ignored.

The second option controls how ratsnest lines are drawn. "All layers" means that ratsnest lines will be drawn between all unconnected items. "Visible layers" means that no ratsnest lines will be drawn to items that are on hidden layers, even when those items are unconnected.

**NOTE**

You can configure the thickness of ratsnest lines in the PCB Editor Editing Options section of the Preferences dialog, to make the ratsnest more or less visible.

## Selection and the selection filter

Selecting items in the editing canvas is done with the left mouse button. When no tool is active, single-clicking on an object will select it. Pressing **Esc** will always cancel the current tool or operation and return to the selection tool. Pressing **Esc** while the selection tool is active will clear the current selection.

You can also select items using a rectangle or lasso selection. These tools can be useful for selecting many items at once. You can switch between rectangular and lasso selection modes using the  /  palette in the right toolbar.

**NOTE**

To switch between multiple tools in a palette, show the palette by clicking and holding the left mouse button on the tool icon or by clicking and dragging the mouse from the tool icon. Release the mouse button on the desired tool to choose it.

With the rectangular selection mode active (  ), clicking and dragging will perform a rectangular selection. A rectangular selection from left to right will only select items that are fully inside the box. A rectangular selection from right to left will select any items that touch the box. A left-to-right selection box is drawn in yellow, with a cursor that indicates exclusive selection, and a right-to-left selection box is drawn in blue with a cursor that indicates inclusive selection.

With the lasso selection mode active (  ), clicking and dragging will begin a lasso selection. Any items in the existing selection are deselected. Dragging with the left mouse button held draws a freeform shape. Releasing the button stops drawing the freeform shape and starts drawing a straight line. Clicking again completes the straight line. Any number of freeform or straight segments can be drawn by repeatedly

clicking or dragging. Double click to finish drawing the lasso. Similar to a rectangular selection, a lasso drawn in a clockwise direction will only select items that are fully inside the lasso. A lasso drawn in a counter-clockwise direction will select any items that touch the lasso. A clockwise lasso is drawn in yellow, with a cursor that indicates exclusive selection, and a counter-clockwise lasso is drawn in blue with a cursor that indicates inclusive selection.

The selection action can be modified by holding modifier keys while clicking or dragging. Modifiers cannot be used with lasso selections.

The following modifier keys apply when clicking to select single items:

Modifier Keys (Windows)	Modifier Keys (Linux)	Modifier Keys (macOS)	Selection Effect
Ctrl	Ctrl	Cmd	Toggle selection. Note: Ctrl+click can be remapped to highlight net in <b>Preferences → PCB Editor → Editing Options</b> .
Shift	Shift	Shift	Add the item to the existing selection.
Ctrl + Shift	Ctrl + Shift	Cmd + Shift	Remove the item from the existing selection.
long click	long click or Alt	long click or Option	Clarify selection from a pop-up menu.

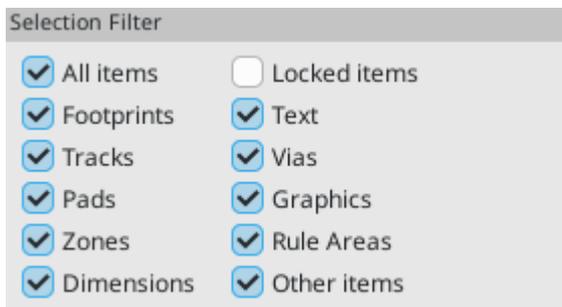
The following modifier keys apply when dragging to perform a rectangular selection:

Modifier Keys (Windows)	Modifier Keys (Linux)	Modifier Keys (macOS)	Selection Effect
Ctrl	Ctrl	Cmd	Toggle selection.
Shift	Shift	Shift	Add item(s) to the existing selection.
Ctrl + Shift	Ctrl + Shift	Cmd + Shift	Remove item(s) from the existing selection.

The selection filter panel in the lower right corner of the PCB Editor window controls which types of objects can be selected with the mouse. Turning off selection of unwanted object types makes it easier to select items in a dense board.

The **All items** checkbox is a shortcut to turn the other items on and off. The **Locked items** checkbox is independent of the rest, and controls whether or not items that have been locked can be selected. You can right-click any object type in the selection filter to quickly change the filter to only allow selecting that type of object.

If you repeatedly try to select an object whose type is disabled in the selection filter, the selection filter will visually flash the checkbox for that object type as a reminder that selecting that object type is disabled.



When a connected copper item is selected, you can expand the selection to other copper items of the same net using the Expand Selection command in the right-click context menu or with the hotkey **U**. The first time you run this command, the selection will be expanded to the nearest pad. The second time, the selection will be expanded to all connected items on all layers. Expanding a selection obeys the selection filter, so the expansion will stop when it reaches an object whose type is disabled in the selection filter. For example, if via selection is disabled in the filter, you can expand a selection until a via is reached.

Selecting an object displays information about the object in the message panel at the bottom of the window. Double-clicking an object opens a window to edit the object's properties.

## Net highlighting

An electrical net (or set of nets) can be highlighted in the PCB editor to visualize how the net is routed across the PCB. Net highlighting can be activated by selecting the net to highlight in the PCB editor or by selecting the corresponding net in the schematic editor when cross-probe highlighting is enabled (see below). When net highlighting is active, the highlighted net or nets will be shown in a brighter color and all other items will be shown in a dimmer color than normal.

There are several ways to select a net or nets to highlight in the PCB editor:

- Use the hotkey **~** after selecting a copper object, or while hovering over a copper object
- Right click a copper object in the editing canvas and select **Net Inspection Tools** → **Highlight Net**
- Right click a net in the **Nets** tab of the Appearance panel and select **Highlight**
- Double click a net in the [Net Inspector](#)

When you press the Highlight Net hotkey, the nets of any selected copper items will be highlighted. If no copper items are selected, the net of the copper item under the editor cursor will be highlighted.

Net highlighting can be cleared by using the Clear Net Highlight action (hotkey **~**) or by using the Highlight net tool on an empty region in the board. By default, **Esc** also clears net highlighting, but this can be disabled if desired in **Preferences** → **PCB Editor** → **Editing Options**.

When a net or nets have been selected for highlighting, the Toggle Net Highlighting action becomes enabled on the left toolbar (also accessible by hotkey, **Ctrl** + **~**). This action will turn the highlighting display on or off without choosing a new net to highlight.

## Cross-probing from the schematic

KiCad allows bi-directional cross-probing between the schematic and the PCB. There are several different types of cross-probing.

**Selection cross-probing** allows you to select a symbol or pin in the schematic to select the corresponding footprint or pad in the PCB (if one exists) and vice-versa. By default, cross-probing will result in the display centering on the cross-probed item and zooming to fit. You can disable the centering and zooming behavior, or disable selection cross-probing entirely, in the Display Options section of the Preferences dialog. Even when selection cross-probing is disabled, you can manually cross-probe from the schematic to the PCB by right-clicking an object and selecting **Select on PCB**, or from the PCB to the schematic by right-clicking an object and choosing **Select** → **Select on Schematic**.

**Highlight cross-probing** allows you to highlight a net in the schematic and PCB at the same time. If the option "Highlight cross-probed nets" is enabled in the Display Options section of the Preferences dialog, highlighting a net or bus in the schematic editor will cause the corresponding net or nets to be highlighted in the PCB editor.

## Left toolbar display controls

The left toolbar provides options to change the display of items in the PCB Editor.

Some toolbar buttons have more than one item available in a palette. These buttons are indicated with a small arrow in the lower-right corner of the button: 

To show the palette, you can click and hold the mouse button on the button or click and drag the mouse. The palette will show the most selected item when it is closed.

The default contents of the left toolbar are shown below.

**NOTE** | You can edit the toolbar's contents in the **Toolbar** page of the PCB Editor Preferences.

	Turns grid display on/off.  <b>Note:</b> by default, hiding the grid does not disable <a href="#">grid snapping</a> . This behavior can be changed in the Display Options section of Preferences.
	Turns item-specific <a href="#">grid overrides</a> on/off.
	Switch between polar and Cartesian coordinate display in the status bar.
	Display/entry of coordinates and dimensions in inches, mils, or millimeters. Clicking the button toggles to the next unit, or you can choose a unit directly by expanding the palette (click and hold/drag).
	Switches the cursor crosshair between small, fullscreen, and 45-degree-rotated fullscreen. Clicking the button toggles to the next crosshair type, or you can choose a crosshair directly by expanding the palette (click and hold/drag).
	Switches the line mode between free angle, 90 degree mode, and 45 degree mode for placement of new tracks, zones, graphical shapes, dimensions, and other objects. Clicking the button toggles to the next line mode, or you can choose a line mode directly by expanding the palette (click and hold/drag). You can also toggle between line modes using <code>Shift + Space</code> .
	Turns the ratsnest display on/off.
	Switches between straight and curved ratsnest lines.
	Switches the non-active layer display mode between Normal and Dim.  <b>Note:</b> this button will be highlighted when the non-active layer display mode is either Dim or Hide. In both cases, pressing the button will change the layer display mode to Normal. The Hide mode can only be accessed via the controls in the Appearance Panel or via the hotkey <code>Ctrl + H</code> .
	When a net has been selected for <a href="#">highlighting</a> , switches the highlighting on or off.  <b>Note:</b> this button will be disabled when no net has been highlighted. To highlight a net, use the hotkey <code>[H]</code> , right-click any copper object in the net and choose Highlight Net from the Net Tools menu, or right-click the net in the list in the Nets tab of the Appearance panel.
	Show zone filled areas.
	Show zone outlines only.
	Switches display of pads between filled and outline mode.
	Switches display of vias between filled and outline mode.
	Switches display of tracks between filled and outline mode.
	Shows or hides the <a href="#">Appearance</a> and <a href="#">Selection Filter</a> panels on the right side of the editor.
	Shows or hides the <a href="#">Properties Manager</a> panel on the left side of the editor.

# Creating a PCB

## Basic PCB concepts

A printed circuit board in KiCad is generally made up of **footprints** representing electronic components and their pads, **nets** defining how those pads connect to each other, **tracks**, **vias**, and **filled zones** that form the copper connections between pads in each net, and various graphic shapes defining the board edge, silkscreen markings, and any other desired information.

KiCad normally keeps the information about nets on a PCB synchronized with an associated schematic, but nets can also be created and edited directly within the PCB editor.

## Capabilities

KiCad is capable of creating printed circuit boards with up to 32 copper layers, 14 technical layers (silkscreen, solder mask, component adhesive, solder paste, etc), and 13 general-purpose drawing layers.

The internal measurement resolution of all objects in KiCad is 1 nanometer, and measurements are stored as 32-bit integers. This means it is possible to create boards up to approximately 4 meters by 4 meters.

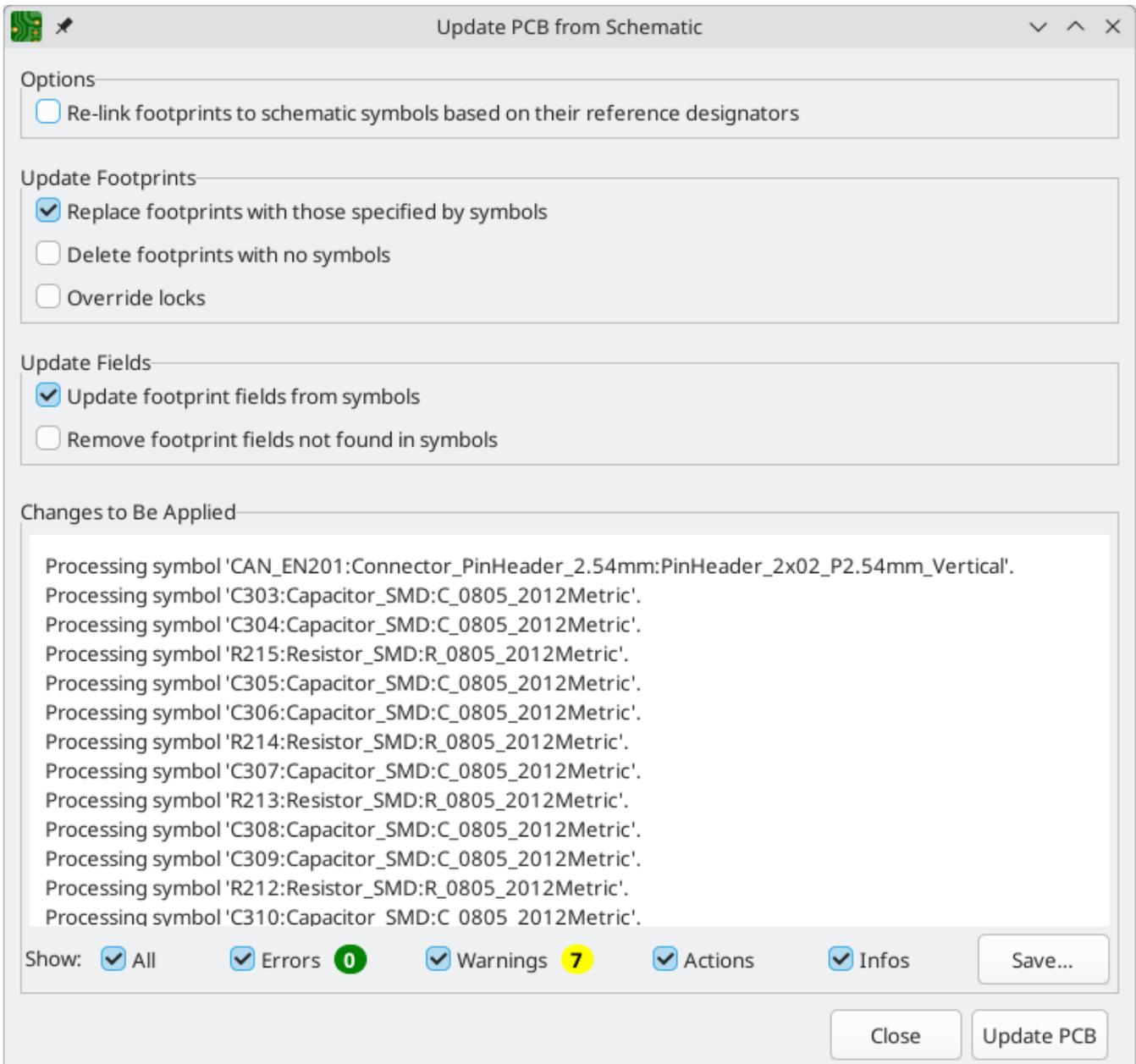
KiCad currently supports one board file per project / schematic.

## Starting from a schematic

Creating a board from a schematic is the recommended workflow for KiCad. When you create a new project, KiCad will generate an empty board file with the same name as the project. To start designing the board after you have created a schematic, simply open the board file. You can do this either from the KiCad project manager, or by clicking the "Open PCB in board editor" button in the schematic editor. To import the schematic design information into the board editor, including footprints and net connections, use the **Tools** → **Update PCB from Schematic...** action (  ). You can also use the  icon in the top toolbar.

### NOTE

Update PCB from Schematic is the preferred way to transfer design information from the schematic to the PCB. In older versions of KiCad, the equivalent process was to export a netlist from the Schematic Editor and import it into the Board Editor. It is no longer necessary to use a netlist file.



For more information about the Update Schematic from PCB tool, see the [forward annotation section of the manual](#).

## Starting from scratch

It is also possible to create a board with no matching schematic, although this workflow has some limitations and is not recommended for most users. To do this, you must start the PCB editor standalone (not from the KiCad project manager). Before beginning your design, it is a good idea to save the board file, which will also create a project file to store board settings. Use "Save As..." from the File menu to choose where to save your board file. A project file with the same name will be created in the same location you choose to save the board file in.

# Board setup

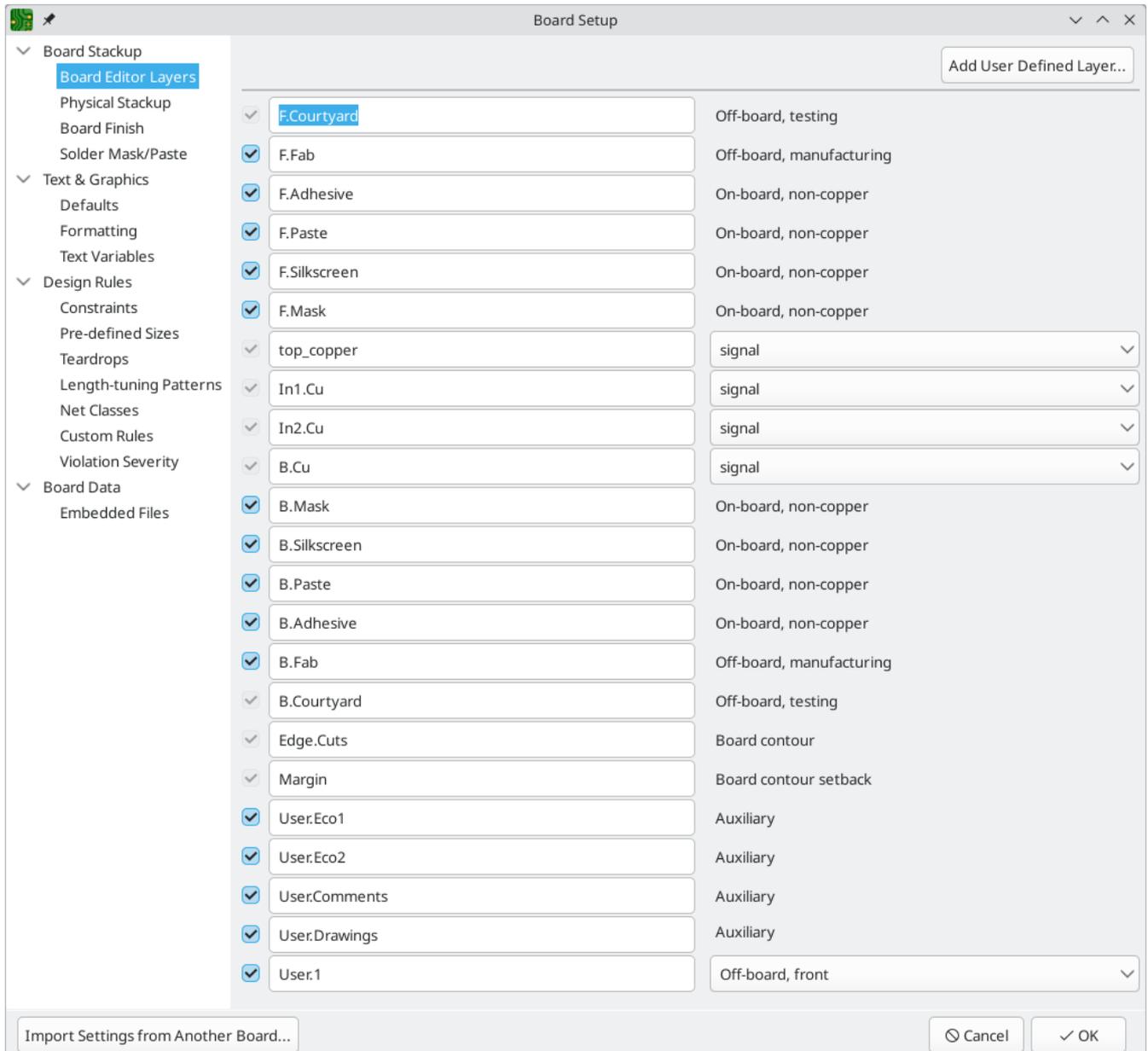
Before beginning your board design, use the Board Setup dialog to configure the basic parameters of the board. To open Board Setup, click the  icon in the top toolbar or choose "Board Setup..." from the File menu.

## Configuring board stackup and physical parameters

The Board Stackup section of Board Setup is used to configure the stackup and layers of the board.

### Board editor layers

The Board Editor Layers page lets you rename layers, disable non-copper layers that will not be used in the design, and add additional user-defined layers for documentation or other purposes. For example, if you will not use a back silkscreen on the design, uncheck the box next to the `B.Silkscreen` layer. Some layers, like copper layers, courtyard layers, and `Edge.Cuts`, are required layers and therefore cannot be disabled.



## NOTE

Copper layers can be designated as signal, power plane, mixed, or jumper in the Board Editor Layers section. This designation is intended as a guide for the user only. Tracks and zones can be routed on any copper layer, no matter what the type is configured to in this dialog.

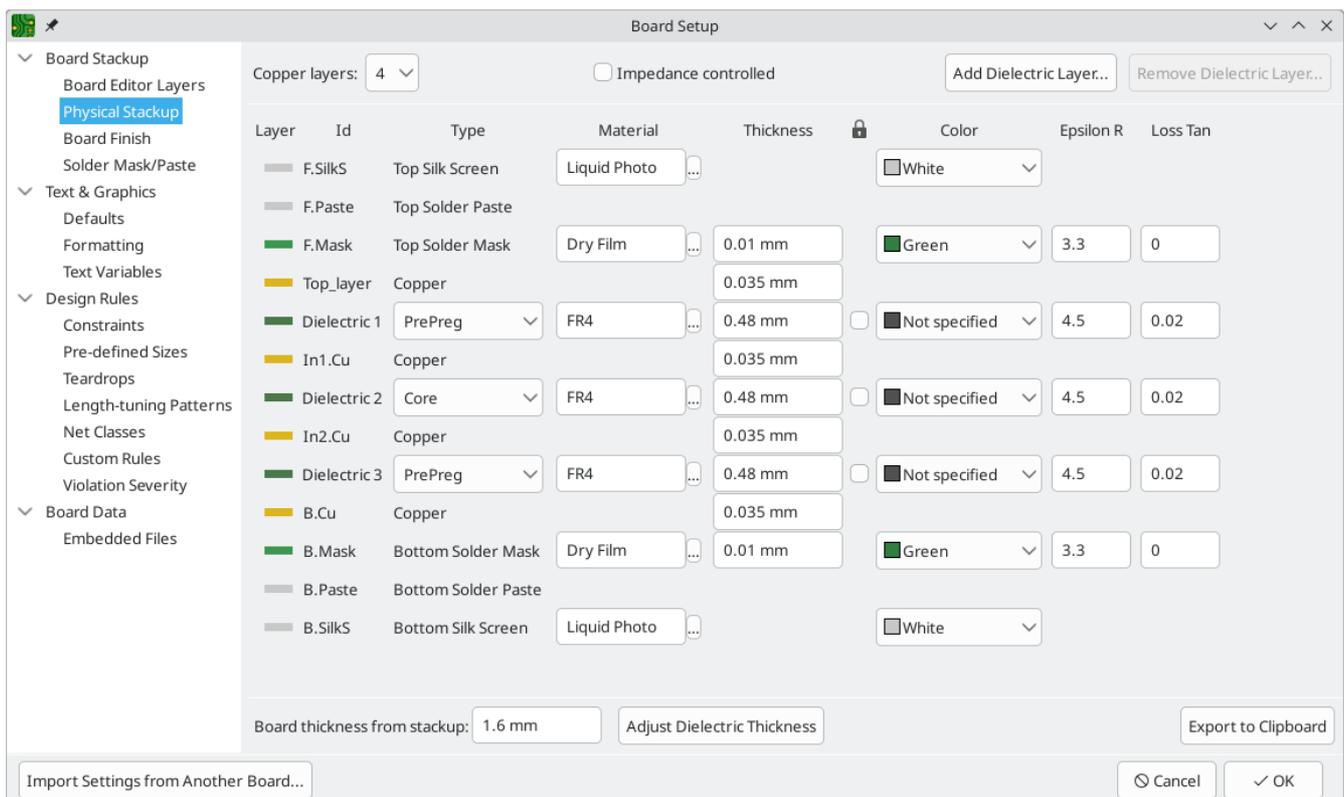
You can add additional user-defined layers ( User . 1 , User . 2 , etc.) by clicking the **Add User Defined Layer...** button in the top right. User-defined layers can't be used for routing, but they can contain arbitrary graphics or other information. By default, user layers are **auxiliary** layers, meaning that whatever information they contain does not correspond to either the front or back of the board. User layers can instead be set to **Off-board, front** or **Off-board, back**, in which case they correspond to the selected side of the board. Items on such layers can be flipped from front to back in the same way as objects on physical front/back layers. Adjacent front/back layers are treated as paired: if User . 2 is defined as a front layer and User . 3 is defined as a back layer, flipping an object on User . 2 will move it to User . 3 , and vice versa.

## Physical stackup

The Physical Stackup page controls the board layers that are part of the PCB layer stackup: copper layers, dielectric layers, solder mask, and silkscreen.

## NOTE

Use the **Board Editor Layers** page to add non-physical layers, configure names for all layers, and enable or disable optional layers.



Set the number of copper layers in the upper left corner and then enter the physical parameters of the stackup if desired. These parameters may be left at their default values, but note that the board thickness value will be used when exporting a 3D model of the board, and layer thicknesses will be included in net length calculations for any nets that include vias. If you plan to use these features, it is a good idea to ensure that the stackup thickness is correct. Dielectric, soldermask, and silkscreen layers can have colors assigned to them, which affects the board's appearance in the 3D viewer and in 3D model exports.

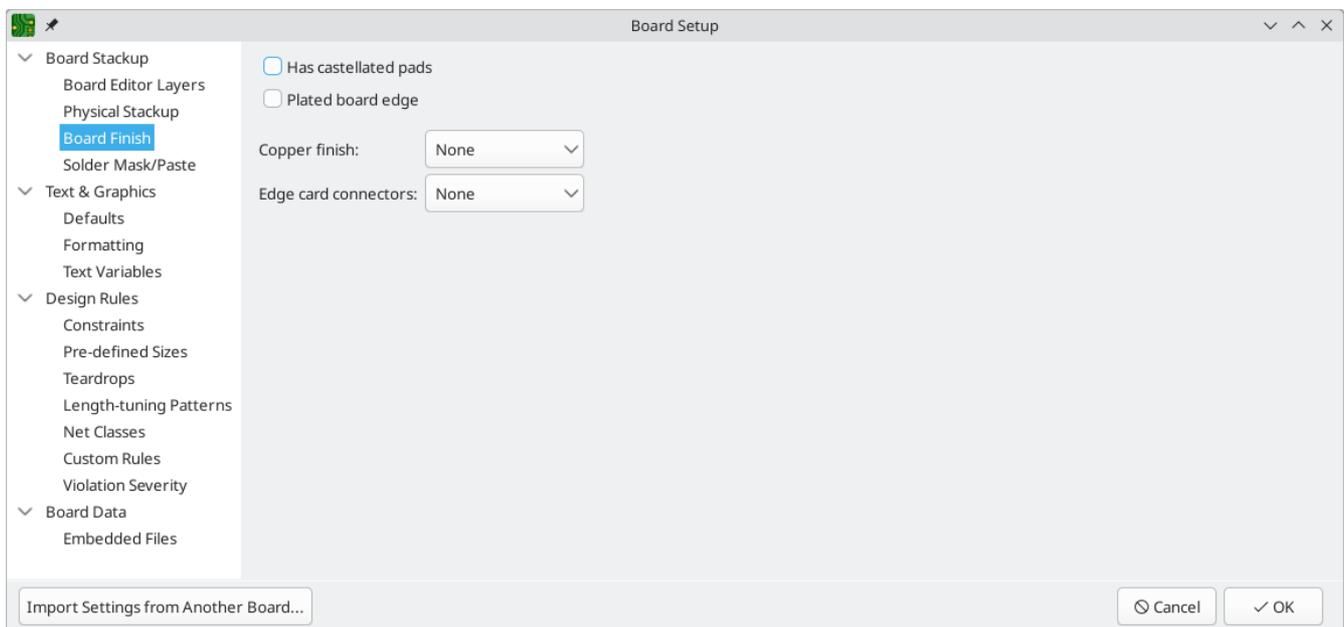
## NOTE

KiCad currently only supports stackups with an even number of copper layers. To create designs with an odd number of layers (for example, flexible printed circuits or metal-core printed circuits), simply choose the next highest even number and ignore the extra layer.

The **Board thickness from stackup** value at the bottom of the page is automatically calculated based on the stackup parameters in the table. You can automatically adjust the thickness of dielectric layers by pressing the **Adjust Dielectric Thickness** button and entering an overall thickness for the PCB. The thickness of the dielectric layers will be adjusted to meet the overall PCB thickness. Any dielectric layers that are locked (the  column is checked) will not be adjusted.

## Board finish

The Board Finish section has settings for defining the copper finish and special features such as castellations or edge plating. Note that these settings only impact the board attributes output as part of Gerber job files at this time.

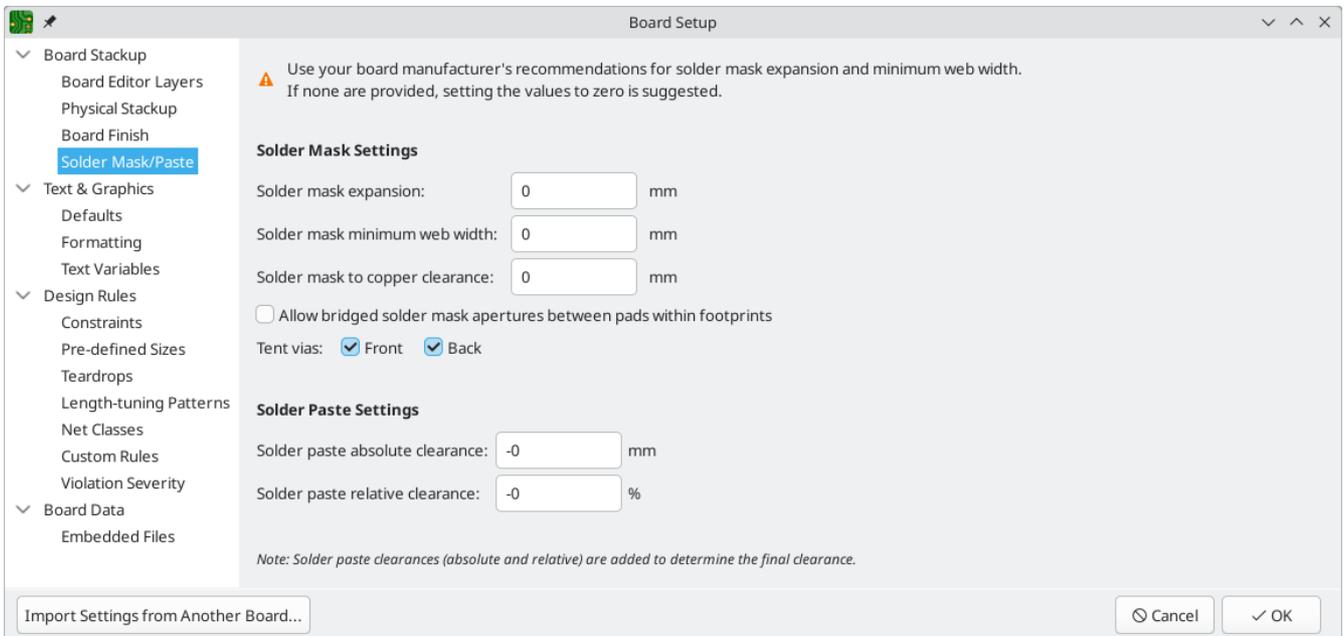


## Solder mask/paste

The Solder Mask/Paste section allows global adjustment of the clearance (positive or negative) between solder mask / solder paste shapes and the copper shapes of the parent pads. These values are global settings, but they will be superseded by any clearance overrides set on individual footprints or pads. Positive clearance values will result in the shape of the solder mask or paste opening being *larger* than the copper shape. Negative clearance values will result in the opening being *smaller* than the copper shape.

## WARNING

Most commercial PCB fabricators expect these values to be zero and make their own adjustments to solder mask and paste openings as part of their CAM process. It is usually best to leave these values at their default of zero unless you are making the PCB yourself or have specific advice from your fabricator to use different values.



- **Solder mask expansion** is a global setting to specify the size of a solder mask opening relative to the parent pad size. If it is 0, solder mask openings will be the same size as the pad. Positive values mean solder mask openings will be larger than pads. Negative values mean solder mask openings will be smaller than pads. This global value is overridden by expansion settings in individual footprints or pads.
- **Solder mask minimum web width** is the minimum width of webs between solder mask openings, or in other words, the minimum distance between solder mask openings. Any solder mask openings that are closer than this minimum distance will be plotted as a single merged opening.
- **Solder mask to copper clearance** is the minimum distance between a solder mask opening and copper with a different net than the opening's parent copper. Distances smaller than this minimum will result in a DRC error.
- **Allow bridged solder mask apertures between pads within footprints** controls whether a DRC violation occurs when multiple pads in the same footprint share a single solder mask opening. This situation can occur when multiple solder mask openings are merged due to the minimum solder mask web width setting.
- **Tent vias** controls whether vias are tented (covered with solder mask) on the top and bottom layers of the board.
- **Solder paste clearance** is a global setting to specify the solder paste shape relative to the parent pad size (the size difference between the pad shape and the aperture shape on the F.Paste and B.Paste layers). This can be specified as an absolute offset from the pad edge (e.g. -0.1mm), a value relative to the pad dimension (e.g. -5%), or both (e.g. -0.1mm - 5%). If it is 0 or blank, the solder paste aperture will be the same size as the pad. Positive values mean solder paste aperture larger than the pad. Negative values mean solder paste aperture smaller than the pad. This global value is overridden by paste clearance settings in individual footprints or pads.

## Zone hatch offsets

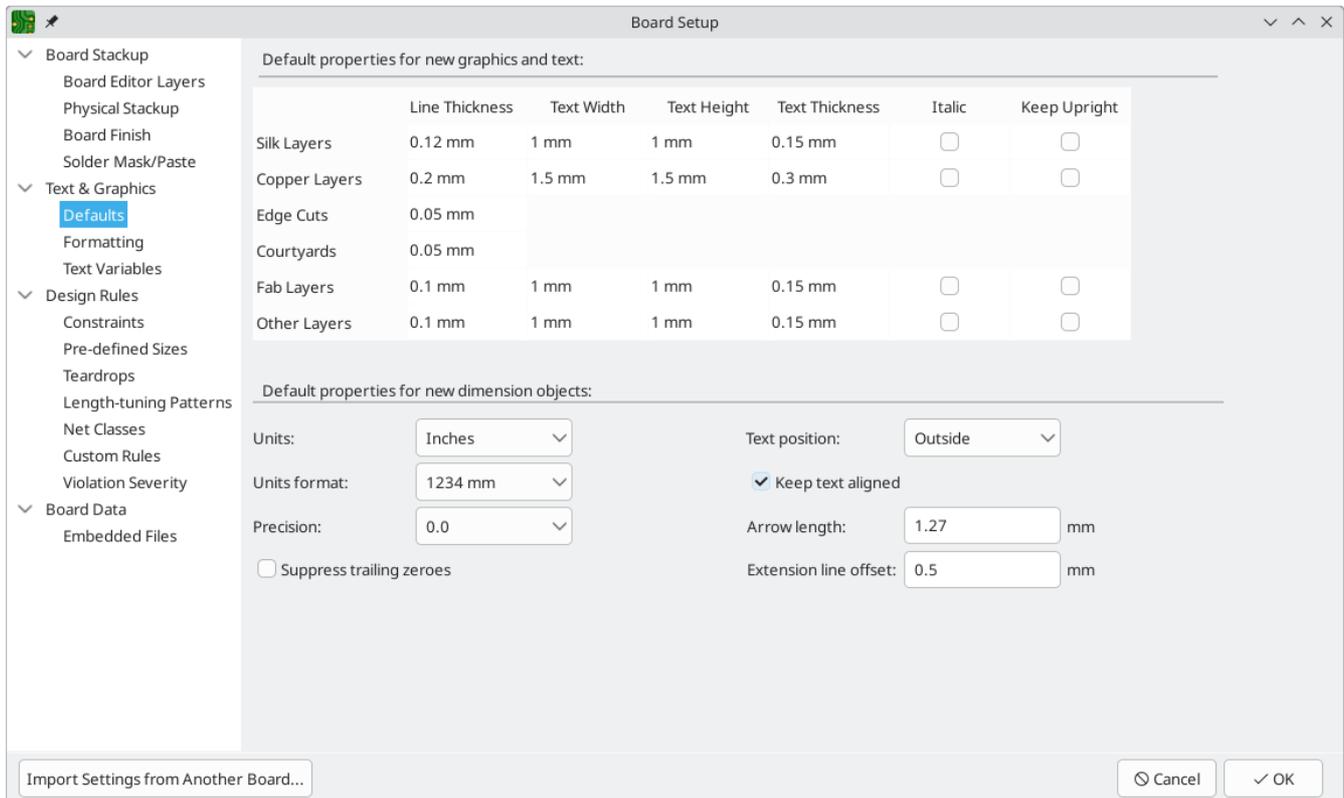
The Zone Hatch Offsets page lets you configure per-layer offsets for zone hatch patterns. Hatched zones on each layer will have their hatch patterns offset in X and Y relative to the patterns on other layers based on the settings on this page.

## Configuring text and graphics

The Text & Graphics section contains formatting settings for text objects, graphic shapes, and dimension objects. It also allows configuring project text variables.

### Defaults

The Defaults section of the Board Setup dialog is used to configure the properties that will be used for new text and graphic shapes that are placed on the board.



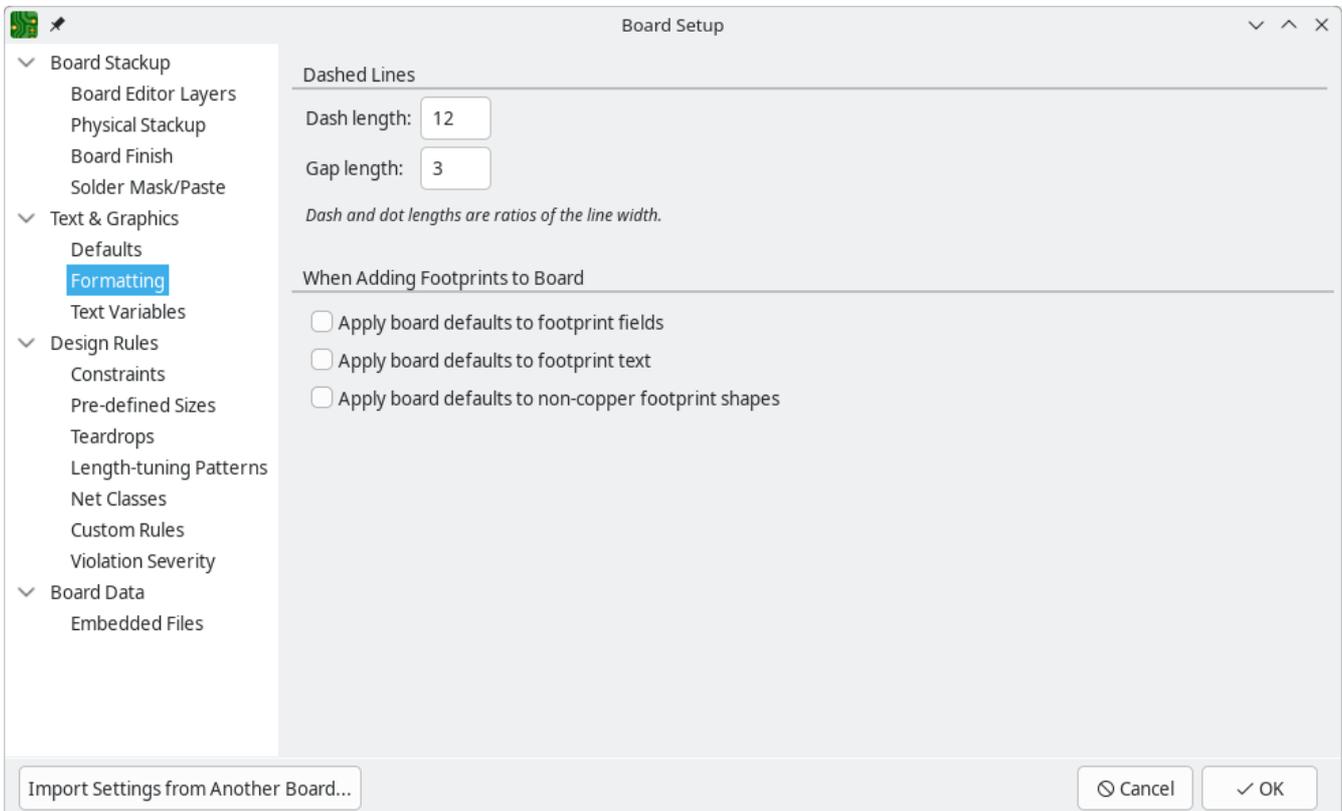
Line thickness, text size, and text appearance can be configured for the six different categories of layers shown in the dialog. These default settings are automatically applied to new text and graphic objects based on the new object's layer. These settings can be overridden on a per-object basis in that object's properties, however.

Additionally, the default properties for dimension objects can be configured for all layers. For more details about dimension properties, see the [dimensions section](#).

The **Defaults** page also contains default settings for new [zones](#). The settings configured here are applied to newly-created zones, but can be overridden on a per-zone basis in the zone's properties.

## Formatting

The Formatting section contains controls for how to format certain board items.



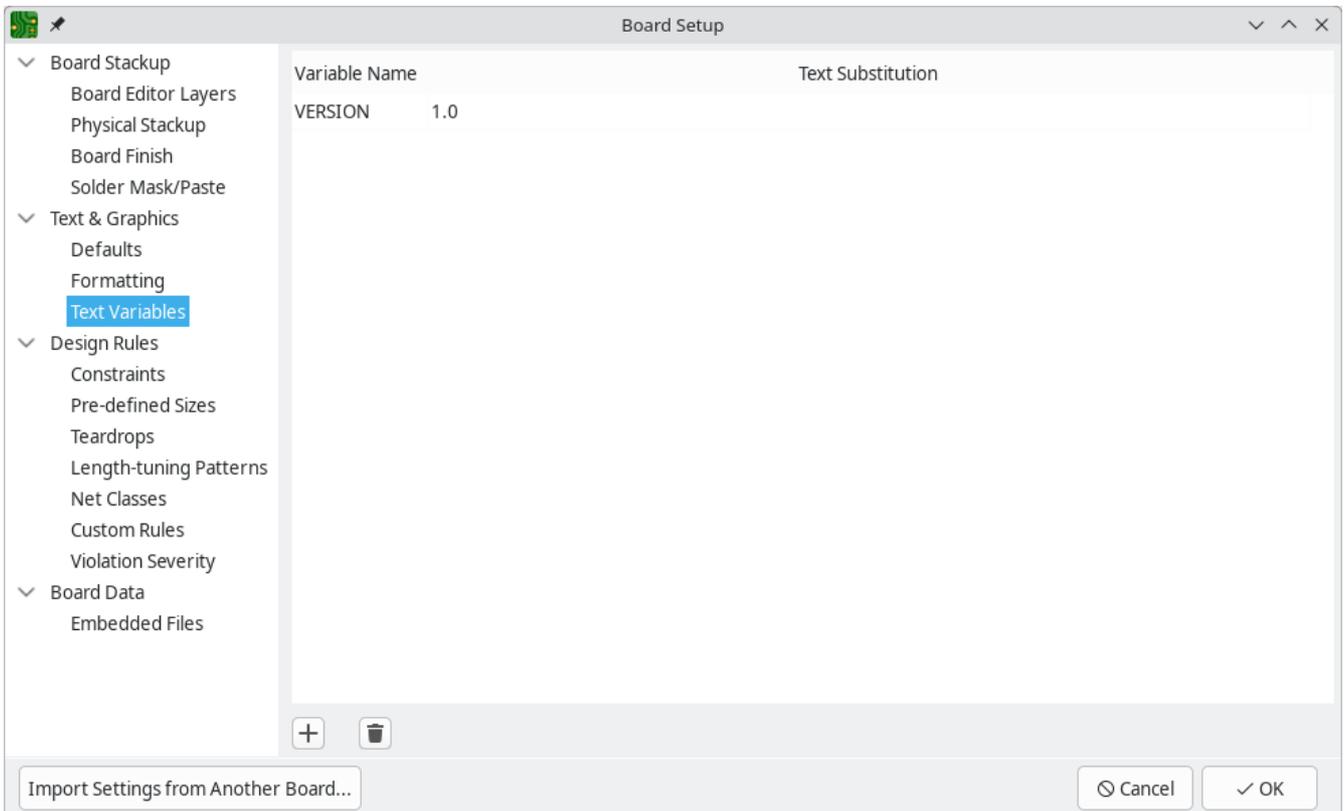
The Dashed Line section controls the appearance of dashed lines. **Dash length** controls the length of dashes, while **Gap length** controls the spacing between dashes and dots. The dash and gap lengths are relative to the line width: a gap length of 2 means twice the width of the line.

The checkboxes at the bottom of the page control how the settings from the [Defaults page](#) are automatically applied to footprints that are added to the board.

- **Apply board defaults to footprint fields:** if checked, default settings will be applied to footprint fields.
- **Apply board defaults to footprint text:** if checked, default settings will be applied to footprint text objects.
- **Apply board defaults to non-copper footprint shapes:** if checked, default settings will be applied to graphic shapes on non-copper layers in footprints. Graphic shapes on copper layers will not be modified.
- **Apply board defaults to footprint dimensions:** if checked, default settings will be applied to footprint [dimensions](#).
- **Apply board defaults to footprint barcodes:** if checked, default settings will be applied to footprint [barcodes](#).

## Text variables

Project text variables can be created in the Text Variables section. KiCad will substitute the variable name with the text string assigned to the variable. This substitution happens anywhere the variable name is used inside the variable replacement syntax of `${VARIABLENAME}`.



For example, you could create a variable named `VERSION` and set the text substitution to `1.0`. Now, in any text object on the PCB, you can enter `${VERSION}` and KiCad will display this as `1.0`. If you change the value to `2.0`, every text object that includes `${VERSION}` will be updated automatically. You can also mix regular text and variables. For example, you can create a text object with the text `Version: ${VERSION}` which will be displayed as `Version: 1.0`.

Text variables can also be created in [Schematic Setup](#). Text variables are project-wide; variables created in the schematic editor are also available in the board editor, and vice versa.

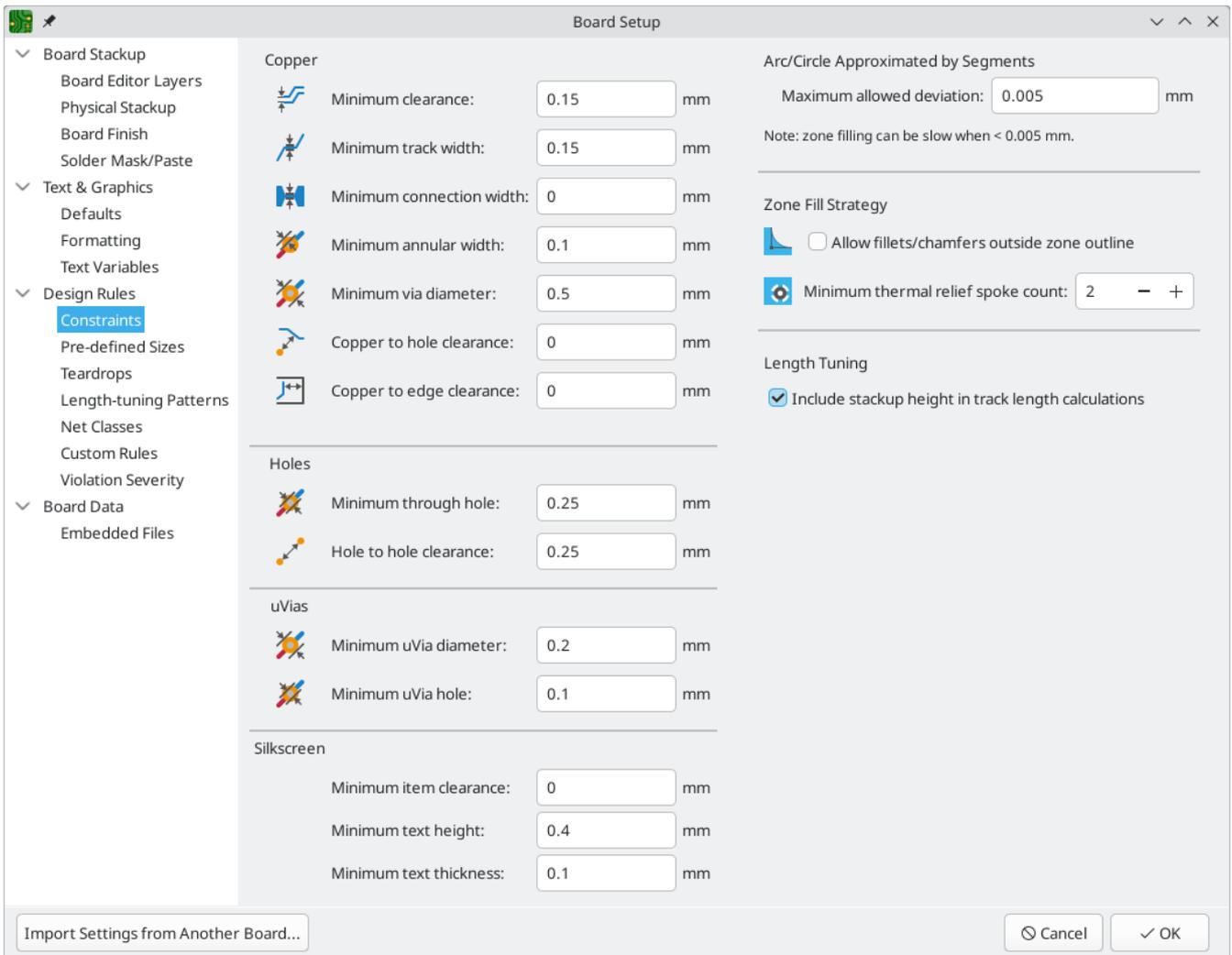
There are also a number of [built-in system text variables](#).

## Configuring design rules

Design rules control the behavior of the interactive router, the filling of copper zones, and the [design rule checker](#). Design rules can be modified at any time, but we recommend that you establish all known design rules at the beginning of the board design process.

### Constraints

Basic design rules are configured in the Constraints section of the Board Setup dialog. Constraints in this section apply to the entire board and should be set to the values recommended by your board manufacturer. Any minimum value set here is an *absolute* minimum and cannot be overridden with a more specific design rule. For example, if you need the copper clearance on part of a board to be 0.2mm and in the rest 0.3mm, you must enter 0.2mm for the minimum copper clearance in the Constraints section and use a net class or custom rule to set the larger 0.3mm clearance.

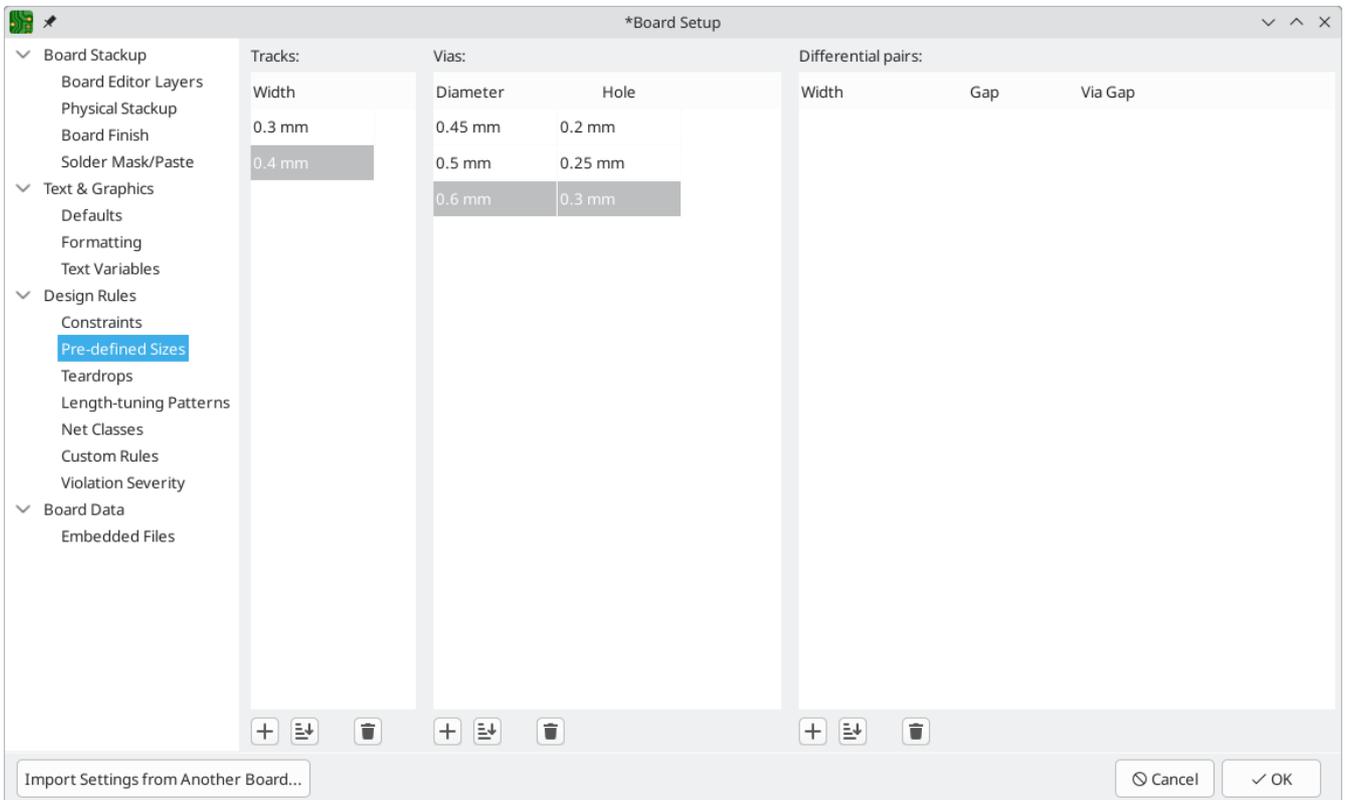


In addition to setting minimum clearances, a number of features can be configured here:

Setting	Description
Arc/circle approximated by segments	In some situations, KiCad must use a series of straight line segments to approximate round shapes such as those of arcs and circles. This setting controls the maximum error allowed by this approximation: in other words, the maximum distance between a point on one of these line segments and the true shape of the arc or circle. Setting this to a lower number than the default value of 0.005mm will result in smoother shapes, but can be very slow on larger boards. The default value typically results in arc approximation error that is not detectable in the manufactured board due to manufacturing tolerances.
Allow fillets outside zone outline	Zones can have fillets (rounded corners) added in the Zone Properties dialog. By default, no zone copper, including fillets, is allowed outside the zone outline. This effectively means that inside corners of the zone outline will not be filleted even when a fillet is configured. By enabling this setting, inside corners of the zone outline will be filleted even though this results in copper from the zone extending outside the zone outline.
Minimum thermal relief spoke count	This sets the minimum acceptable number of thermal relief spokes connecting a pad to a zone. A DRC violation will be generated if this constraint is violated.
Include stackup height in track length calculations	By default, the length tuner uses the height of the stackup to calculate the additional length of a track that travels through vias from one layer to another. This calculation relies on the board stackup height being correctly configured. In some situations, it is preferable to ignore the height of vias and just calculate the track length assuming that vias add no length. Disabling this setting will exclude via length from length tuner track length calculations.

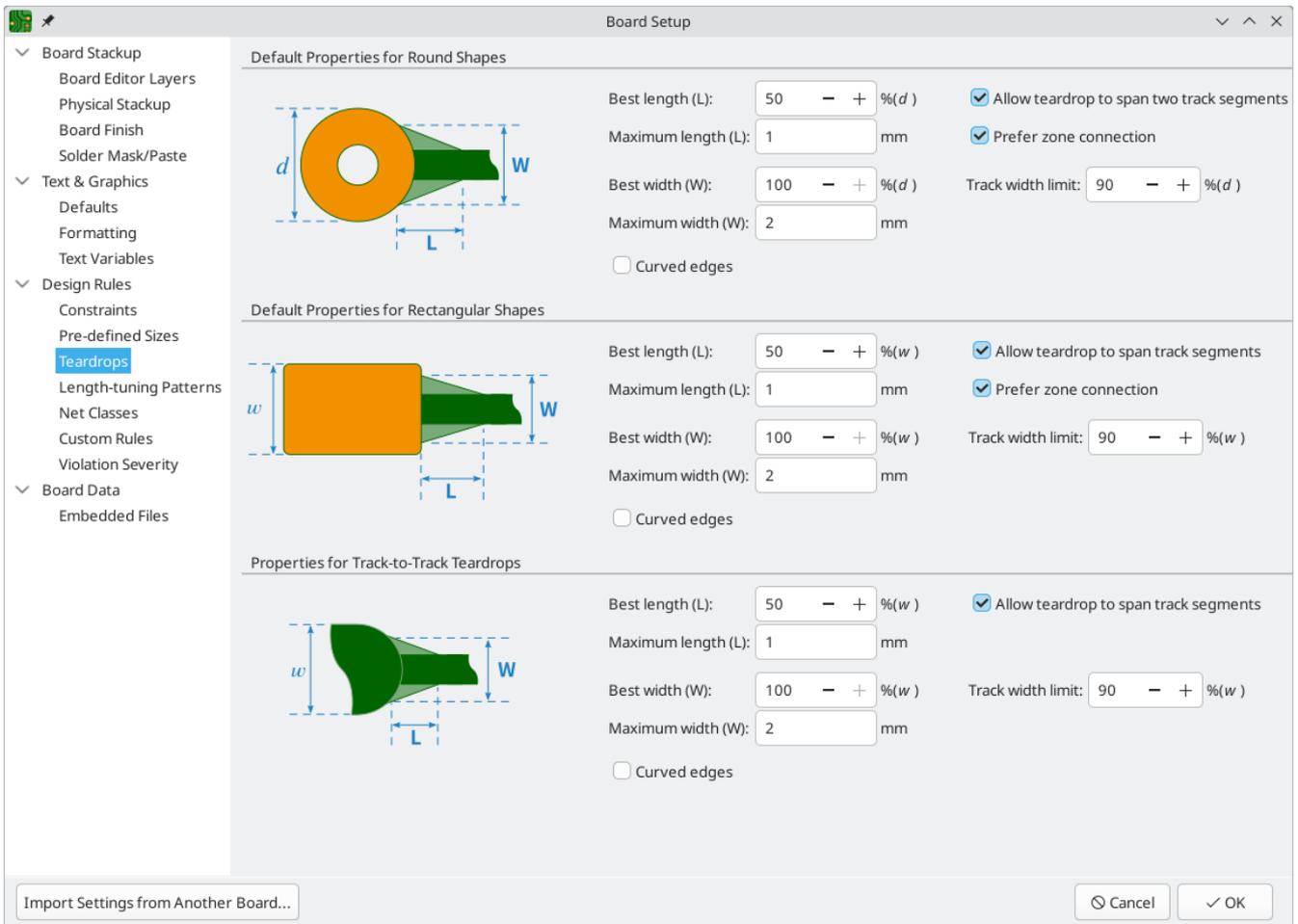
## Pre-defined sizes

The Pre-defined Sizes section allows you to define the track and via dimensions you want to have available while routing tracks. Net classes can be used to define the default dimensions for tracks and vias in different nets (see below) but defining a list of sizes in this section will allow you to step through these sizes while routing. For example, you may want the default track width on a board to be 0.2 mm, but use 0.3 mm for some sections that carry more current, and 0.15 mm for some sections where space is limited. You can define each of these track widths in the Board Setup dialog and then switch between them when routing tracks.



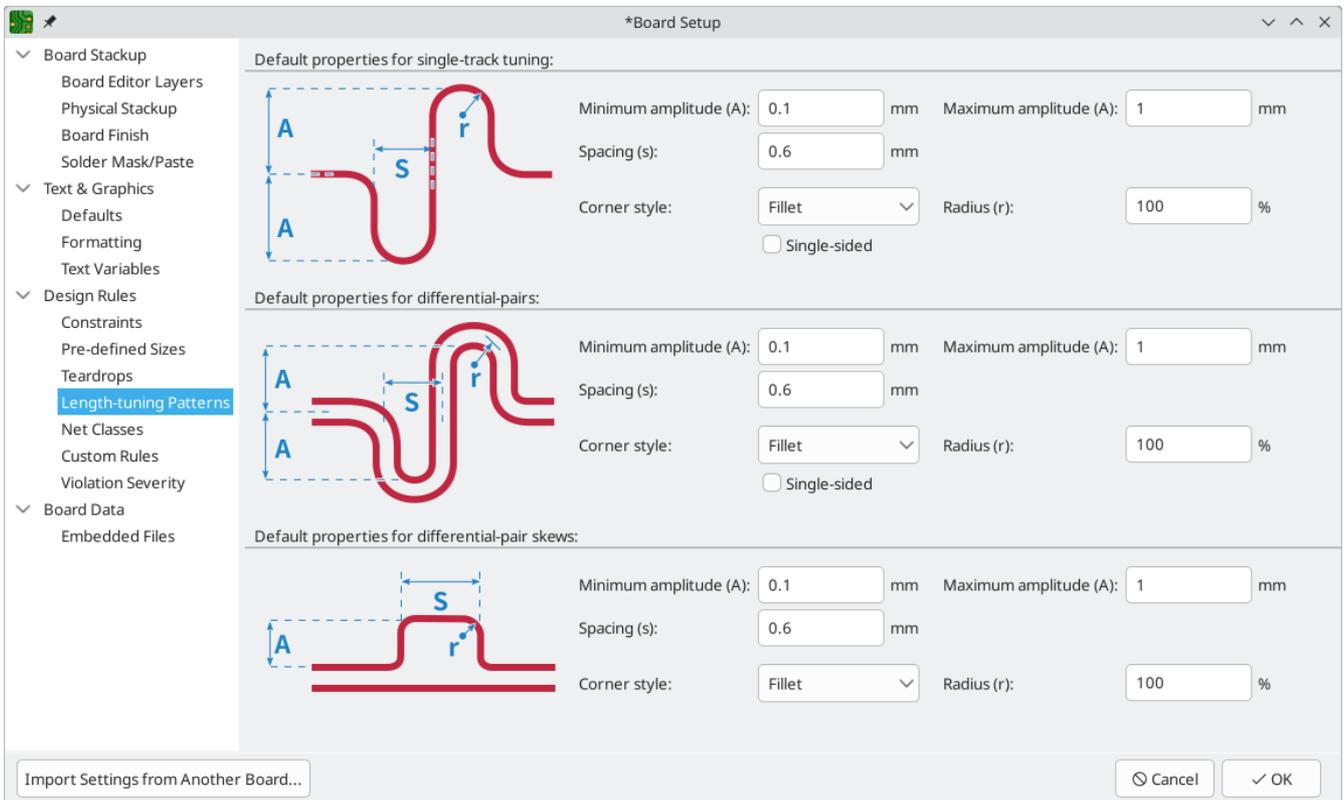
## Teardrops

The teardrops section lets you set default parameters for various types of teardrops. There are different settings for teardrop connections to round objects, rectangular objects, and teardrop connections between tracks. The default teardrop parameters can be overridden when teardrops are added, and also changed in the properties for individual connected items. See the [teardrops documentation](#) for more information about each setting.



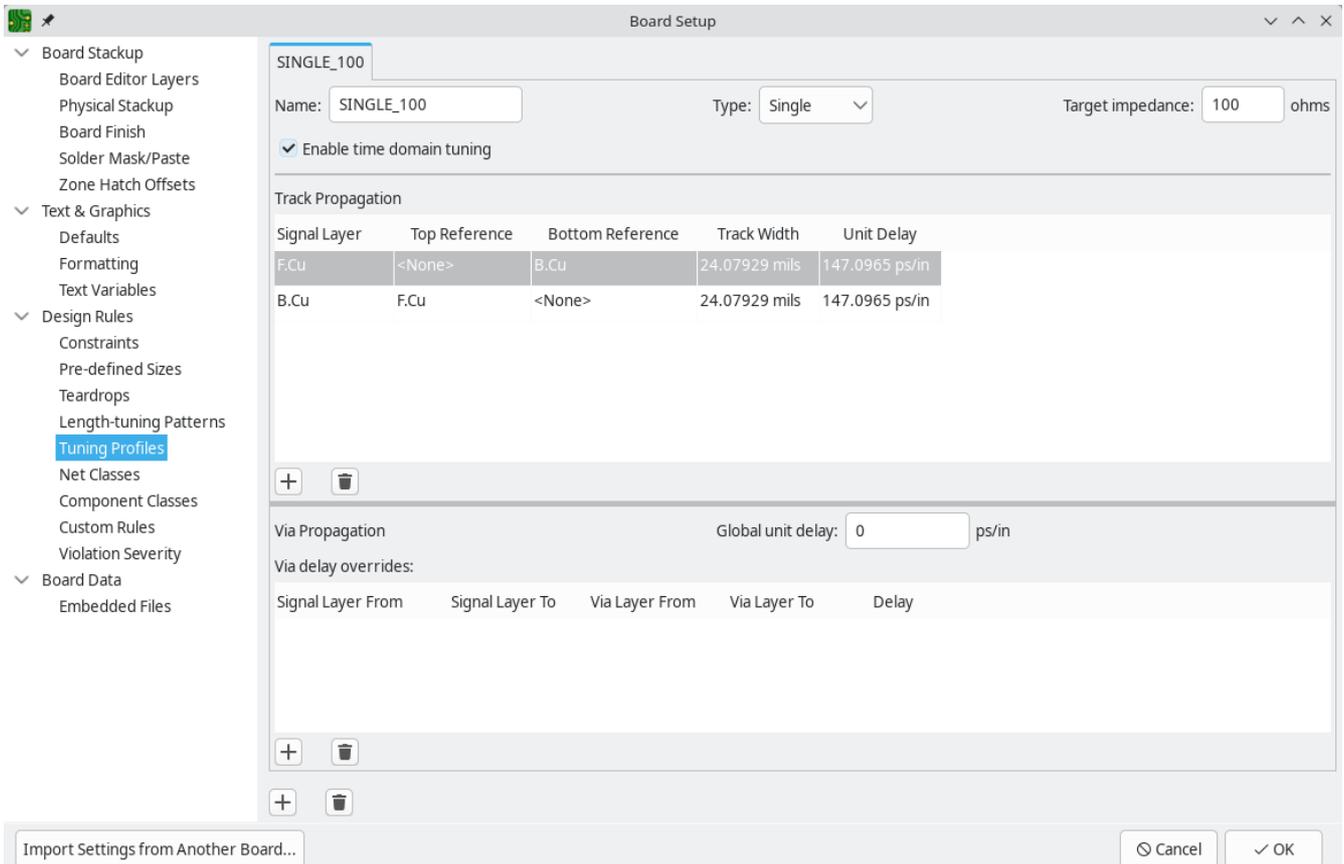
## Length-tuning patterns

The length-tuning patterns section lets you set default parameters for each type of length-tuning pattern (single-track length, differential-pair length, and differential-pair skew). These defaults can be overridden in the properties of each tuning pattern added to the board. See the [length tuning documentation](#) for more information.



## Tuning profiles

Tuning profiles let you define per-layer routing parameters for single-ended and differential tracks with specific impedance requirements. You can automatically calculate track width and gap based on the board stackup and target signal impedance. Tuning profiles can be assigned to net classes, after which the interactive router will use the specified values. [DRC can optionally generate violations](#) for tracks that don't match their tuning profile's parameters.



To add a tuning profile, press the **+** button. You must specify a name for the tuning profile and select whether it applies to single tracks (**Single**) or differential tracks (**Differential**). The **Target impedance** is optional, but required for the tool to calculate track geometry. If **Enable time domain tuning** is checked, the length tuner will use time domain tuning when tuning tracks assigned this tuning profile; otherwise the tuner will tune based on track length.

The **Track Propagation** section lets you define per-layer track geometries and propagation delays. Each row in the table represents tracks on a different layer. Press the **+** button below the table to add a new row, or press the **🗑** button to remove the selected row.

For each row, you can select the **Signal Layer**, the **Bottom Reference** layer, and optional **Top Reference** layer (for stripline geometries). Each row (signal layer) also has **Track Width** and **Unit Delay** values, and differential pairs additionally have a **Diff Pair Gap** value.

You can enter width, gap, and delay values manually (after calculating them with an external tool) or press the **↻** button in each field to calculate them automatically. The width and gap values are calculated based on the target impedance and the stackup thickness between the signal and reference layer(s). For differential pairs, at least one of the width and gap must be manually entered; the other can be calculated. The delay value is calculated based on the track geometry. The calculator assumes a microstrip or stripline geometry, depending on whether one or two reference layers, respectively, are specified. To use the calculators, you must enter a **Target Impedance** for the profile and configure the [board's stackup](#).

The **Via Propagation** section lets you define propagation delays for vias. The **Global unit delay** field is the general propagation delay per unit distance for vias. You can override this general unit delay by adding overrides for individual layer pairs in the table below.

Each row in the table represents vias between specific layer pairs. Press the **+** button below the table to add a new row, or press the **🗑** button to remove the selected row.

Each override has **Signal Layer From**, **Signal Layer To**, **Via Layer From**, and **Via Layer To** layers. Signal Layer From/To are the layers on which the signal tracks connect to the via. Via Layer From/To are the start and end layers of the full via structure. Through vias start and end at the top and bottom layers of the board. Other types of vias can start and end on inner layers, depending the stackup and via geometry.

Each override also has a **Delay** value, with which you specify the propagation delay for that type of via.

To remove an entire tuning profile, press the **🗑** button at the bottom of the page.

You can use tuning profiles to route tracks with specific geometries by assigning the tuning profile to a net class. The interactive router will then use the tuning profile's geometry values as net class width and differential pair gap values when you route tracks that belong to that net class, the length tuner will use the propagation delays for time-domain tuning, and [DRC violations can optionally be generated](#) for tracks that do not match the specified geometry.

To assign a tuning profile to a net class, go to the [Net Classes page of Board Setup](#), then choose a tuning profile in that net class's **Tuning Profile** column. You may need to scroll the table or unhide the column in order to see it.

## When to enable time-domain tuning

The **Enable time domain tuning** checkbox controls whether the length tuner operates in time-domain mode (matching propagation delay) or length-domain mode (matching physical trace length) for nets assigned to this tuning profile.

Enable time-domain tuning when:

- Matched nets are routed on multiple layers with different propagation velocities (e.g., a mix of microstrip and stripline routing).
- Your design specification defines timing budgets in picoseconds rather than millimeters.
- Via transitions between layers with different dielectric properties contribute meaningful delay differences that length matching alone cannot capture.

When time-domain tuning is disabled, the tuning profile still provides per-layer track geometry (width and gap) for the interactive router and DRC, but the length tuner matches physical length rather than propagation delay.

### NOTE

You can override the tuning mode on a per-net basis using custom DRC rules. A `length` or `skew` constraint written with time units (e.g., `500ps`) will force the length tuner into time-domain mode for matching nets, regardless of the tuning profile setting.

## Understanding track propagation parameters

Each row in the **Track Propagation** table represents a signal layer and its reference plane(s). The combination of signal layer and reference layer(s) determines the transmission line geometry:

- **One reference layer** (bottom reference only): The calculator assumes a **microstrip** geometry. This is typical for outer layers where the trace sits above or below a single reference plane.
- **Two reference layers** (bottom and top reference): The calculator assumes a **stripline** geometry. This is typical for inner layers where the trace is sandwiched between two reference planes.

The propagation velocity (and therefore the unit delay) differs between microstrip and stripline geometries because the effective dielectric constant is different. Stripline traces are fully enclosed in dielectric material and have a higher effective dielectric constant, resulting in slower propagation (higher unit delay per mm). Microstrip traces are partially exposed to air, resulting in faster propagation (lower unit delay per mm).

The **Unit Delay** value for each layer is expressed as propagation delay per unit distance (for example, ps/mm). You can enter this value manually if you have calculated it with an external impedance calculator, or press the  auto-calculate button to have KiCad compute it from the board stackup and trace geometry.

### TIP

After making changes to the board stackup (layer thicknesses, dielectric materials), press the  button in each delay field to recalculate the unit delays. The delay values are not automatically updated when the stackup changes.

## Understanding via propagation parameters

The **Via Propagation** section controls how via delays are calculated in time-domain mode.

The **Global unit delay** is the default propagation delay per unit distance for all vias. This value is multiplied by the via's electrical height (the stackup distance between the layers the signal actually uses, not the full via span) to produce the via's delay contribution.

For more precise control, you can add **per-layer-pair overrides** in the table below the global delay field. Each override specifies an absolute delay value for a via transition between specific layers. Overrides take priority over the global unit delay calculation.

Each override row has four layer fields:

- **Signal Layer From** and **Signal Layer To**: The copper layers on which the routed tracks connect to the via.
- **Via Layer From** and **Via Layer To**: The start and end layers of the full via structure (e.g., F.Cu and B.Cu for a through via).

This four-layer specification allows you to define different delays for the same physical via structure depending on which layers the signal actually transitions between.

## How tuning profiles, net classes, and the length tuner work together

The relationship between these three features is:

1. A *tuning profile* defines the per-layer track geometry and propagation delay parameters for a specific impedance target.
2. A *net class* groups nets that share common routing parameters. Assigning a tuning profile to a net class applies the profile's geometry and delay values to all nets in that class.
3. The *length tuner* reads the tuning profile associated with a net's net class. If the profile has time-domain tuning enabled, the tuner uses the profile's per-layer delay values to calculate and match propagation delay. If time-domain tuning is not enabled, the tuner falls back to matching physical length.

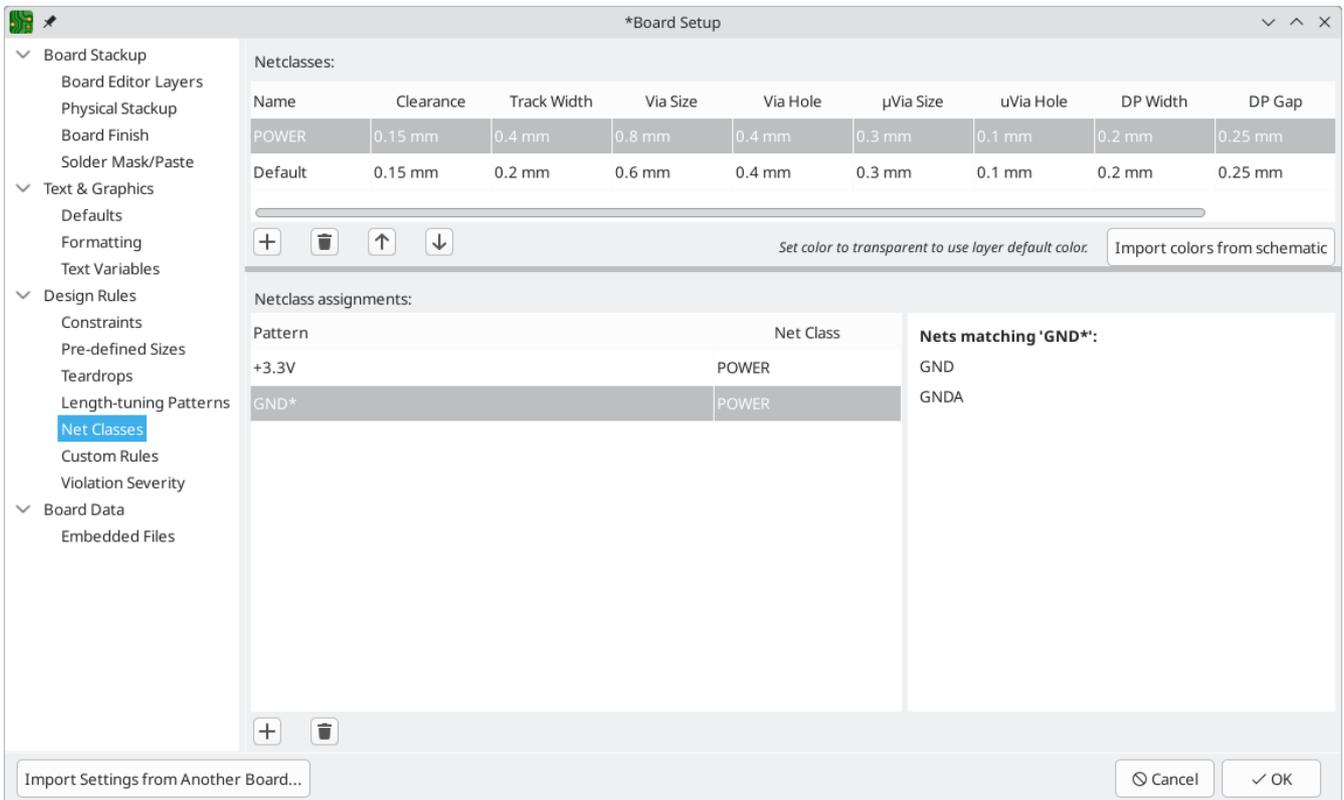
When the interactive router routes a track belonging to a net class with an assigned tuning profile, it automatically uses the track width and differential pair gap defined in the tuning profile for the current layer. This ensures that the routed geometry matches the assumptions used in the propagation delay calculations.

## Net classes

The Net Classes section allows you to configure routing and clearance rules for different classes of nets.

More than one net class can be assigned to a net. For nets with multiple net classes assigned, an effective aggregate net class is formed, taking any net class properties from the highest priority net class which has that property set. Net class priority is determined by the ordering in the Schematic or Board Setup dialogs. The `Default` net class is used as a fallback for any missing properties after all explicit net classes have been considered; this means that nets may be part of the `Default` net class even if they have other net classes explicitly assigned.

[Net classes may be created and edited](#) in either the Schematic or Board Setup dialogs.



The upper portion of the Net Classes section contains a table showing the net classes in the design and the design rules that apply to each net class. Some columns in the table may be hidden or scrolled to the side. You can show or hide columns in the table by right-clicking on the table header and checking or unchecking columns in the menu.

Every class has values for copper clearance, track width, via sizes, and differential pair sizes. These values will be used when creating tracks and vias unless a more specific rule overrides them (see Custom Rules below).

#### NOTE

No rule may override the minimum values set in the Constraints section of Board Setup. For example, if you set a net class clearance to 0.1 mm, but the Minimum Clearance in the Constraints section is set to 0.2 mm, nets in that class will have a clearance of 0.2 mm.

The track widths and via sizes defined for each net class are used when the track width and via size controls are set to "use netclass values" in the PCB editor. These widths and sizes are considered the default, or optimal, sizes for that net class. They are not minimum or maximum values. Manually changing the track width or via size to a different value from that defined in the Net Classes section will not result in a DRC violation. To restrict track width or via size to specific values, use [Custom Rules](#).

You can assign a [tuning profile](#) to a net class in the **Tuning Profile** column. This assigns per-layer track geometry (track width and differential pair gap) and propagation delays for tracks and vias belonging to that net class. Like other net class values, the interactive router uses these values for routing tracks. The length tuner also uses the propagation delays from the tuning profile when tuning tracks in time-domain mode. [DRC violations can optionally be generated](#) for tracks that don't match their tuning profile's geometry by setting the severity of the "Tuning profile track geometries" violation to either Warning or Error.

Each net class can also have a color assigned to it using the **PCB Color** column. Depending on how net colors are configured in the [appearance panel](#), net class colors can override the default color for ratsnest lines or copper objects. In addition to arbitrary colors for each net class, you can set all net classes to use the same

color as configured for them in the schematic editor by clicking the **Import colors from schematic** button. To use a layer's default color instead of overriding it with a custom net class color, set the net class color to transparent.

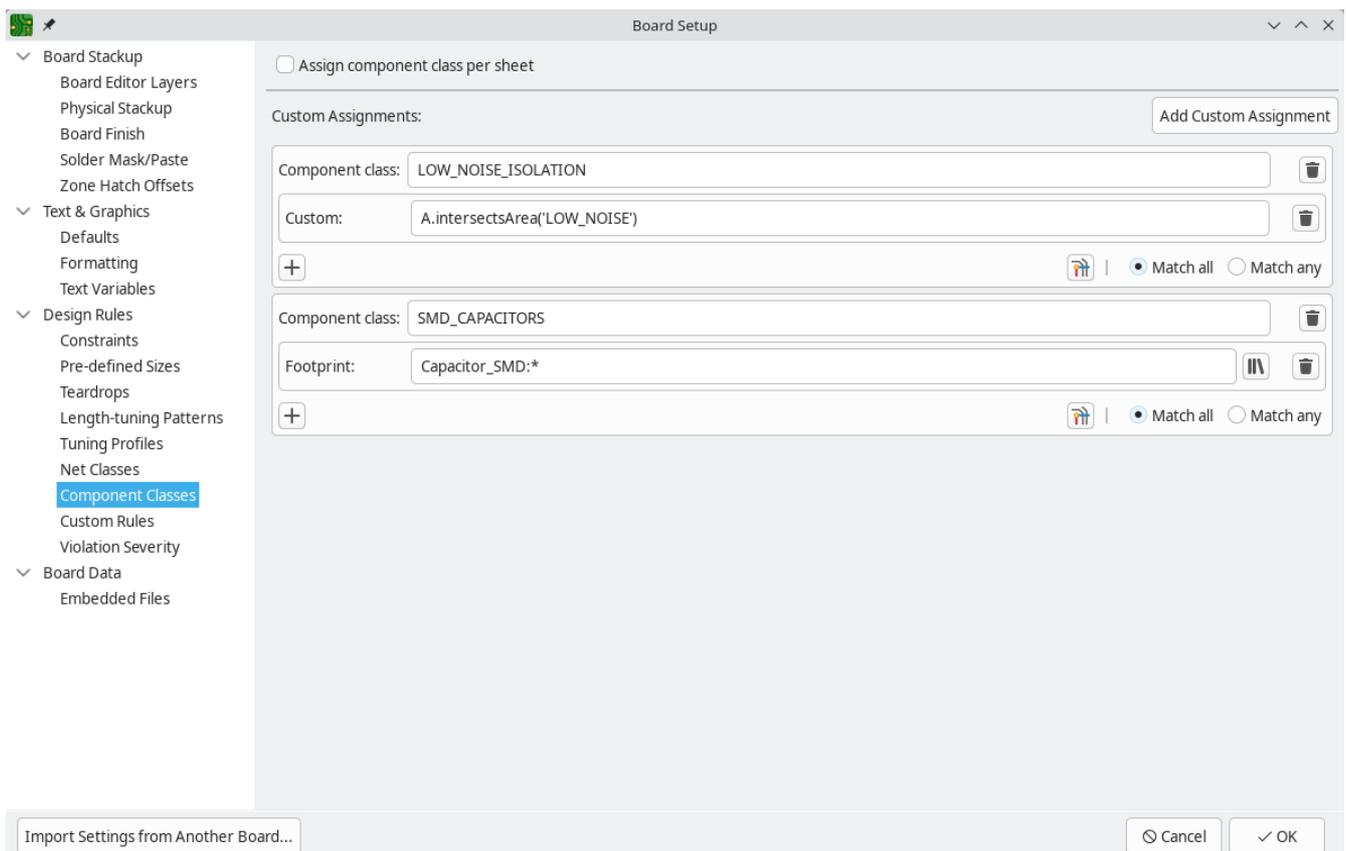
The lower portion of the Net Classes section lists pattern-based net class assignments. Working with pattern-based net class assignments is explained in the [Schematic Editor documentation](#); pattern-based assignments can be edited in either the Board or Schematic Setup windows.

Note that pattern-based assignments can be created directly from the PCB editing canvas by right clicking a copper track or zone and clicking **Assign netclass...** Net classes can also be assigned in the schematic using [net class directives or labels](#) instead of pattern-based assignments.

## Component classes

The Component Classes section allows you to create rules that automatically assign components to component classes. In addition to these automatic assignments, you can [manually assign component classes in the Schematic Editor](#).

Component classes are named groupings of components: they are assigned to symbols in the schematic or to footprints in the board, but however they are assigned they apply to both the symbols and the corresponding footprints. They can be used to group symbols into channels for [multichannel designs](#) and can also be used to group footprints in [custom DRC rules](#). Components can have more than one class.



Enabling the **Assign component class per sheet** will create a component class for every sheet in the schematic and assign each component in the sheet to that sheet's component class.

You can add a rule for assigning a component class by clicking the **Add Custom Assignment** button. Enter the name of the component class you want to assign in the **Component class** textbox, then add a condition

for when to assign the class by pressing the **+** button and selecting a type of condition from the menu that appears.

The following types of conditions are available:

- **Reference:** matches by footprint reference designator. More than one reference designator can be given as a comma-separated list. The reference field support wildcards: **\*** matches any number of any characters, including none, and **?** matches any single character. Pressing the  button uses the selected footprints' reference designators in the condition.
- **Footprint:** matches by footprint library and identifier. The footprint field support wildcards: **\*** matches any number of any characters, including none, and **?** matches any single character. Pressing the  button opens a window to choose a footprint from your libraries.
- **Side:** matches by side of the PCB (front, back, or any).
- **Rotation:** matches by footprint rotation angle.
- **Footprint Field Value:** matches by the value of a specified footprint field. The field name and value support wildcards: **\*** matches any number of any characters, including none, and **?** matches any single character.
- **Sheet:** matches by the name of the schematic sheet containing the footprint's linked symbol.
- **Custom Expression:** matches by a [custom DRC rule condition clause](#), which matches footprints that satisfy the clause. For example, `A.intersectsArea('some_area_name')` matches any footprints that intersect the named area `some_area_name`.

You can add multiple conditions to a single rule. If **Match all** is selected, the component class will be assigned to any components that match all of the conditions. If **Match any** is selected, the component class will be assigned to components that match any of the conditions.

You can test a component class rule by pressing the  button, which highlights all footprints that match the rule in the editing canvas.

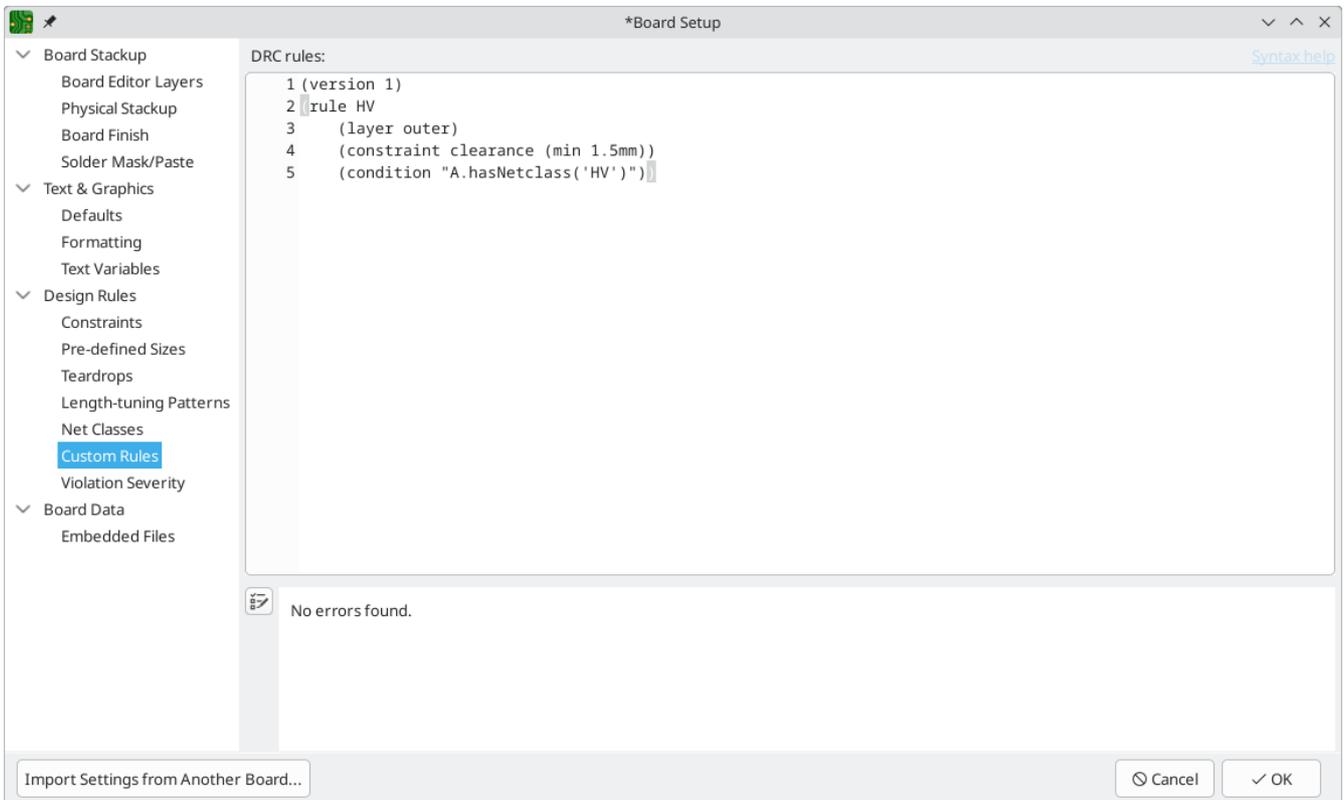
To delete a condition, press the  button next to the condition. To delete a rule and all of its conditions, press the  button next to the component class name.

## Custom rules

The Custom Rules section contains a text editor for creating design rules using the custom rules language. Custom rules are used to create specific design rule checks that are not covered by the basic constraints or net class settings.

Custom rules will only be applied if there are no errors in the custom rules definitions. Use the Check Rule Syntax button to test the definitions and fix any problems before closing Board Setup.

See [Custom Design Rules](#) in the Advanced Topics chapter for more information on the custom rules language as well as example rules.

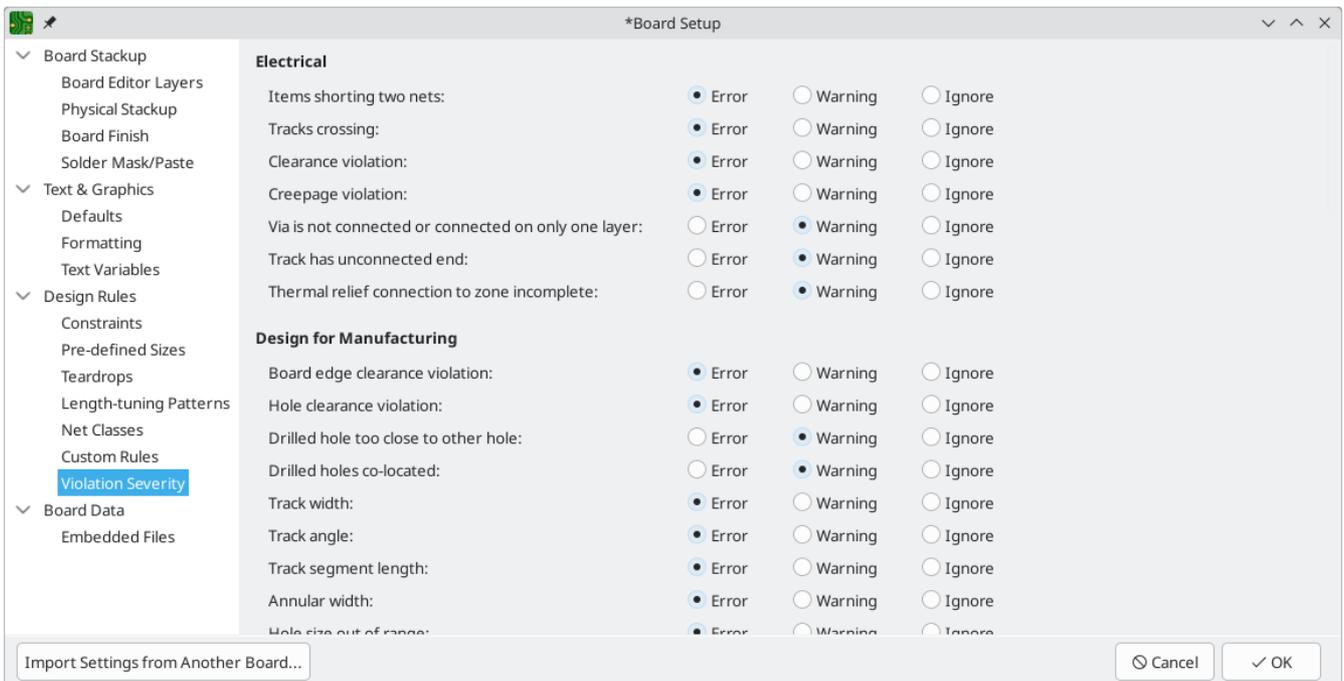


## Violation severity

The Violation Severity section allows you to configure the severity of each type of design rule check. Each rule may be set to create an error marker, a warning marker, or no marker (ignored).

### NOTE

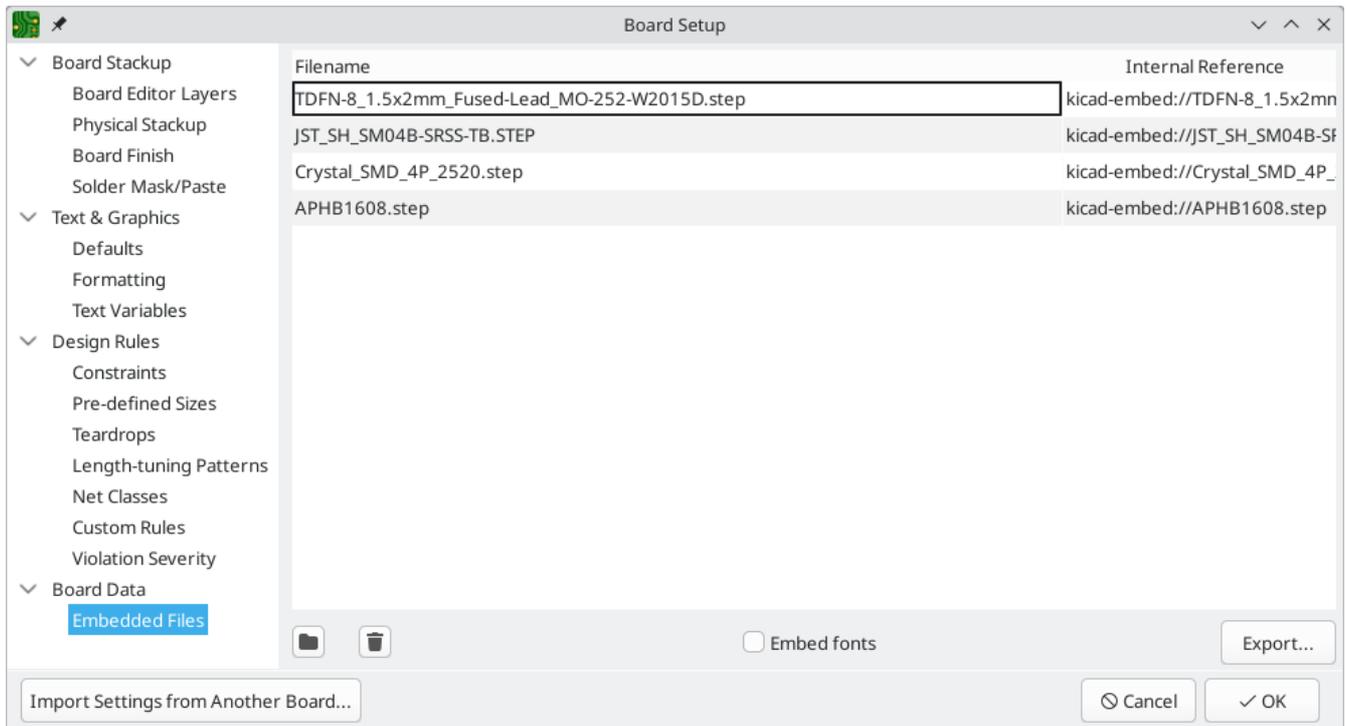
Individual rule violations may be ignored in the Design Rule Checker. Setting a rule to Ignore in the Violation Severity section will completely disable the corresponding design rule check. Use this setting with caution.



For descriptions of each violation type, and how to ignore individual violations without disabling all violations of that type, see the [DRC documentation](#).

## Embedding files

External files can be embedded within a board file. Embedding a file stores a copy of the file inside the board file. The design can then refer to the embedded copy of the file instead of the external file, which makes the project more portable as it doesn't rely on an external file. Fonts, datasheets, drawing sheets, SPICE models, and footprint 3D models can be embedded and used within KiCad. Other arbitrary files can also be embedded to store them in the project for later export, but they are not used by any KiCad functionality. Files embedded in a board necessarily increase the board's file size, although files are compressed before being embedded to minimize the space required.



Embedded files are managed in the Embedded Files section of Board Setup. All files embedded in a board are shown here. To embed a file inside a board, click the  button and select the file. The file is then embedded inside the PCB and is listed in the embedded files list along with its *embedded reference*. The embedded reference is a unique identifier for the embedded file that begins with `kicad-embed://`. You can use the embedded reference elsewhere in the Board Editor to refer to the embedded file as if it were an external file path. You can copy the embedded reference by right clicking and selecting **Copy Embedded Reference**. To remove an embedded file, click the  button. Any remaining links to the removed file will become invalid.

### NOTE

3D models and drawing sheets can be embedded directly using the file browser when you add them to a footprint ([3D models](#)) or to a board ([drawing sheets](#)) by enabling the **Embed Files** option in the file browser. This is a single-step shortcut for adding the files in Board Setup and then referring to them by their embedded reference; the result is the same.

To embed any fonts used in a board, check the **Embed fonts** checkbox. All fonts used in the board design will be embedded, so text using that font can be edited on any computer regardless of whether the font file is installed.

You can also [embed files in a footprint](#), either in the board copy of a footprint or in a library. Such files will be available within the footprint instance but not within the larger board design or within other footprints. Files embedded in a footprint are deduplicated when the footprint is added to a board: if a file is embedded in a footprint, and multiple instances of that footprint are added to the board, only one copy of the file will be embedded, and all of the footprint instances will refer to the same embedded file.

As an example, to embed a 3D model in a project and use it within several footprints, you could embed the model using the Board Setup dialog, copy the internal reference, and paste the internal reference as a 3D model path in each footprint that uses that model. Alternatively, you could embed the model within a single footprint, either in the board or in the source footprint library. In this case, the footprint itself is portable if you export the footprints from the board, and the model embedding is managed in the footprint's properties rather than Board Setup. A more convenient way to achieve the same thing, however, is to open the footprint's properties dialog, add a 3D model file, and enable the **Embed File** option in the file browser. Again, this could be done for a footprint in the board or for a footprint in the source footprint library.

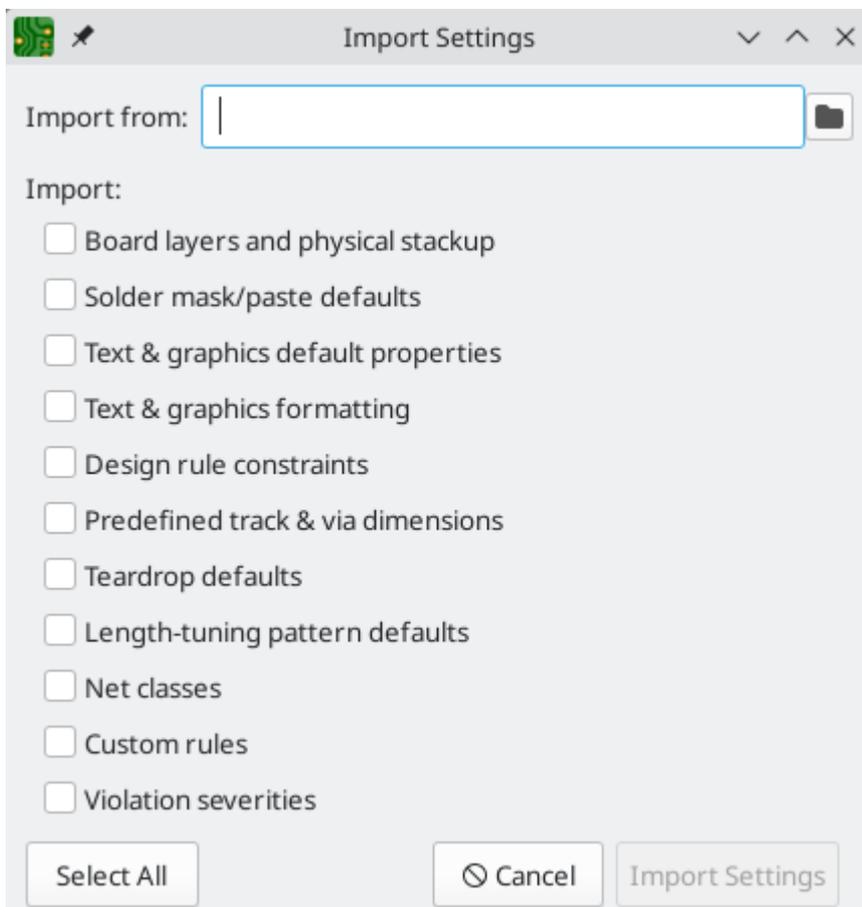
**NOTE**

You can embed all of your board's footprints at once using **Tools** → **Collect and Embed 3D Models**. This takes every external 3D model referenced by the board's footprints and embeds the models in the board. The 3D model references in each footprint are replaced by references to the corresponding embedded files.

Files can also be embedded in [schematics](#).

## Importing settings

You can import part or all of the board setup from an existing board. This technique can be used to create a "template" board that has the settings you want to use on multiple designs, and then importing these settings from the template board into each new board rather than entering them manually.



To import settings, click the **Import Settings from Another Board...** button at the bottom of the Board Setup dialog and then choose the `kicad_pcb` file you want to import from. Select which settings you want to import and the current settings will be overwritten with the values from the chosen board.

The settings that are available to import are:

- Board layers and physical stackup
- Solder mask/paste defaults
- Text and graphics default properties
- Text & graphics formatting
- Design rule constraints
- Predefined track & via dimensions
- Teardrop defaults
- Length-tuning pattern defaults
- Net classes
- Custom rules
- Violation severities

# Editing a board

## Placement and drawing operations

Placement and drawing tools are located in the right toolbar. When a tool is activated, it stays active until a different tool is selected or the tool is canceled with the `Esc` key. The selection tool is always activated when any other tool is canceled.

Some toolbar buttons have more than one tool available in a palette. These tools are indicated with a small arrow in the lower-right corner of the button:



To show the palette, you can click and hold the mouse button on the tool or click and drag the mouse. The palette will show the most recently used tool when it is closed.

The default contents of the right toolbar are shown below.

### NOTE

You can edit the toolbar's contents in the **Toolbar** page of the PCB Editor Preferences.

	<b>Selection tool</b> (the default tool).
	When the rectangular selection mode is active, clicking and dragging performs a rectangular selection.  When the lasso selection mode is active, clicking and dragging performs a lasso selection.  Only one selection mode is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
	Local ratsnest tool: when the board ratsnest is hidden, selecting footprints with this tool will show the ratsnest for the selected footprint only. Selecting the same footprint again will hide its ratsnest. The local ratsnest setting for each footprint will remain in effect even after the local ratsnest tool is no longer active.
	<b>Footprint placement tool</b> : click on the board to open the footprint chooser, then click again after choosing a footprint to confirm its location.
 	<b>Route tracks / route differential pairs</b> : These tools activate the interactive router and allow placing tracks and vias. The interactive router is described in more detail in the <a href="#">Routing Tracks</a> section. Only one router type is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
 	<b>Tune length</b> : These tools allow you to tune the length of single tracks or the length or skew of differential pairs, after they have been routed. Only one tuner type is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
	<b>Add vias</b> : place a standalone ("free") via without routing tracks.
	<b>Add filled zone</b> : Click to set the start point of a zone, then configure its properties before drawing the rest of the zone outline. Zone properties are described in more detail below.

	<b>Add rule area:</b> Rule areas, formerly known as keepouts, can restrict item placement and zone fills. You can also define named areas and apply specific custom DRC rules to them.
	<b>Draw lines.</b> <b>Note:</b> Lines are graphical objects and are not the same as tracks placed with the Route Tracks tool.
	<b>Draw arcs:</b> pick the center point of the arc, then the start and end points. By right clicking this button, you can change the arc editing mode between a mode that maintains the existing arc center and a mode that maintains the arc radius.
	<b>Draw rectangles.</b> Rectangles can be filled or outlines.
	<b>Draw circles.</b> Circles can be filled or outlines.
	<b>Draw graphical polygons.</b> Polygons can be filled or outlined. <b>Note:</b> Filled graphical polygons are not the same as filled zones: graphical polygons cannot be assigned to a net and will not keep clearance from other items.
	<b>Draw bezier curves.</b> Each click alternates between fixing a curve node and fixing the control handle for the node that was just placed.
	<b>Add bitmap image</b> for reference. Reference images are not included in fabrication outputs.
	<b>Add text.</b>
	<b>Add a textbox.</b>
	<b>Add a table.</b>
	<b>Add dimensions.</b> Only one dimension type is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
	<b>Add barcodes.</b>
	Deletion tool: click objects to delete them.
	<b>Set grid origin or drill/place origin (used for fabrication outputs).</b> Only one origin type is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
	Add a point. Points are nonphysical, dimensionless objects that can be used for snapping and moving other objects.
	<b>Interactively measure</b> the distance between two locations.

## Grids and snapping

When moving, dragging, and drawing board elements, you can make these operations snap to a grid or to snapping points on pads and other items. In complex designs, snap points can be so close together that it makes the current tool action difficult. Both grid and object snapping can be disabled while moving the mouse by using the modifier keys in the table below.

**NOTE** | On Apple keyboards, use the **Cmd** key instead of **Ctrl**.

Modifier Key	Effect
<b>Ctrl</b>	Disable grid snapping.
<b>Shift</b>	Disable object snapping.

Tools only snap to objects on visible layers. You can reduce unwanted snapping points by hiding unneeded layers or using the single-layer view mode. Additionally, you can toggle between snapping to objects on all layers or only snapping to objects on the current layer by pressing **Shift** + **S**.

Snapping to different types of objects (pads, tracks, and graphics) can be configured in the Editing Options section of the PCB Editor preferences.

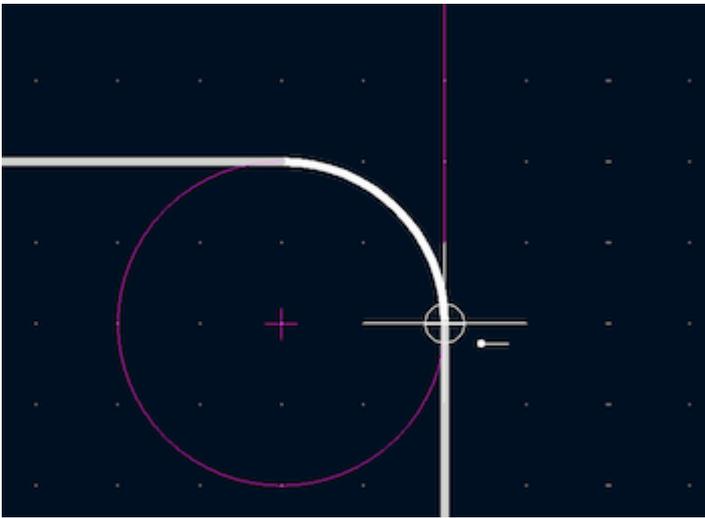
## Snapping to graphical shapes

When working with [graphic shapes](#) like rectangles or arcs, such as when drawing shapes or when selecting a reference point for a move operation, many additional snapping points are available that let you snap to features of existing graphic shapes.

Available snapping points for graphic shapes include:

- Endpoints and corners
- Midpoints
- Centers
- Intersection points

When you hover over a snap point with a shape tool active, a graphical icon will be shown that indicates a snapping point is active and explains the type of snapping point. Clicking will use that snapping point. Some shapes display auxiliary snapping lines that appear when you snap to part of that object. For example, line segments display an auxiliary line that continues the segment beyond its endpoint, and arcs display an auxiliary circle that completes the arc's circumference. Auxiliary shapes can be used for snapping just like the original shape. An auxiliary line and circle are shown as a solid purple line in the screenshot below. The cursor indicates that the active snapping point is the endpoint of a line.



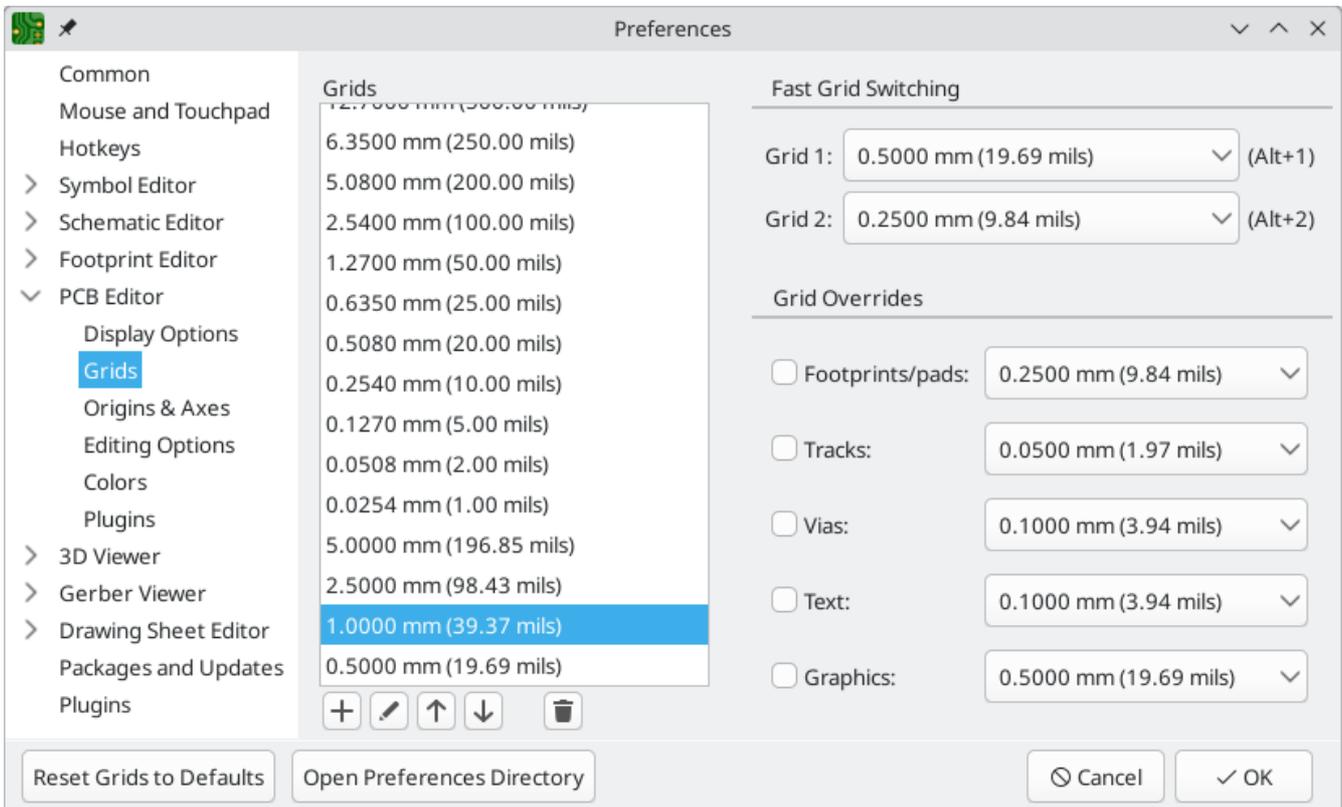
If you move the cursor away from the snapping point, a horizontal or vertical dashed line will appear, depending on the direction of motion. This indicates a horizontal or vertical projection from the snapping point, respectively. Following the line will maintain a position that is horizontally or vertically aligned to the original snap point. This projection is shown as a purple dashed line in the screenshot below.



## Grid settings

Interactive editing operations are snapped to the active grid. You can adjust the grid size using the grid dropdown in the top toolbar or by right-clicking and selecting a new grid from the list in the **Grid** submenu. Pressing the **n** or **N** hotkeys will cycle to the next and previous grid in the list, respectively.

You can also select a new grid or edit the available grids in the **Grids** pane of the preferences dialog. As a shortcut to reach this dialog, right click the  button on the left toolbar and select **Edit Grids...**



In this dialog you can select an active grid from the list of grids, reorder the list of grids ( $\uparrow$  /  $\downarrow$ ), and add ( $+$ ), remove ( $\text{trash}$ ), or edit ( $\text{pencil}$ ) grids. Grids defined in this dialog can have unequal X and Y spacing as well as an optional name. The grid spacing and name are specified when you create or edit a grid.

This dialog also lets you designate two grids from the list as "Fast Grids", which can be quickly selected using  $\text{Alt} + 1$  and  $\text{Alt} + 2$ .

Finally, you can configure grid overrides for different types of objects. Grid overrides let you set particular grid sizes for different types of objects which will be used instead of the default grid when working with those objects. For example, you can set a 100 mil grid for footprints and pads while using smaller grids to finely position tracks, vias, and text. Grid overrides can be individually enabled and disabled in this dialog, or globally enabled and disabled using the  $\text{grid}$  button on the left toolbar ( $\text{Ctrl} + \text{Shift} + \text{G}$ ).

To change the origin (zero point) of the grid, use **Place**  $\rightarrow$  **Grid Origin** and click to place the origin in the canvas. This function is also available with the  $\text{grid}$  button in the right toolbar. Alternatively, you can enter explicit coordinates for the grid origin with **Edit**  $\rightarrow$  **Grid Origin....**

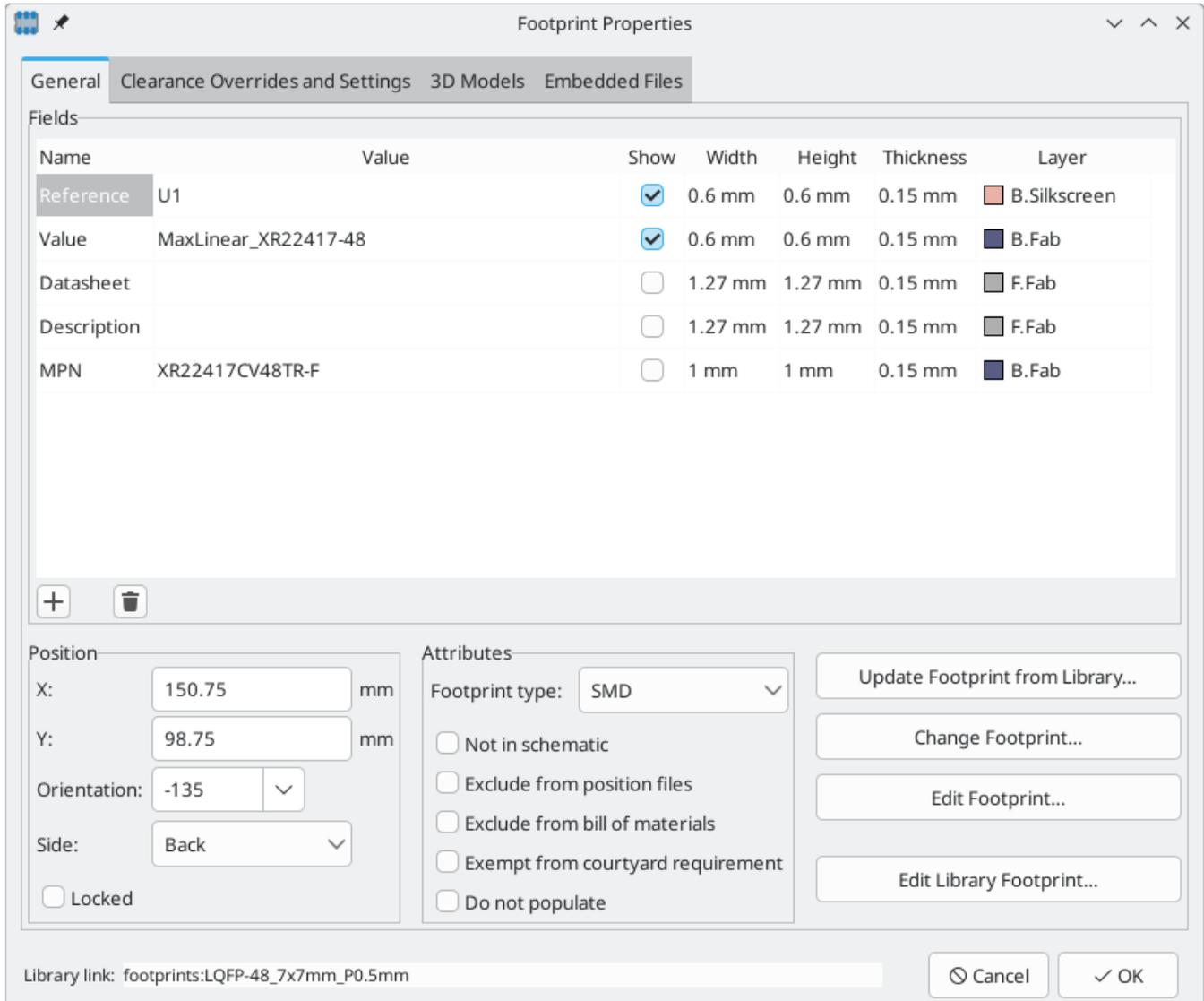
**NOTE** The grid origin is one of several different origins in KiCad, which aren't necessarily set to the same point. The grid origin is the point that the grid aligns to; shifting the grid origin also shifts every grid point. The page origin is an absolute origin which is always the top left corner of the drawing sheet. The drill/place file origin is a configurable point that can be used for fabrication outputs (**Place**  $\rightarrow$  **Drill/Place File Origin**). Finally, the local origin is a quickly settable relative origin that current cursor location by pressing  $\text{Space}$ ; the cursor coordinates relative to the local origin are displayed in the status bar.

The visual appearance of the grid can also be customized in several ways. You can change the thickness of the grid markings, switch their shape (dots, lines, or crosses), and set the minimum displayed spacing in the **Display Options** page of the preferences dialog, and you can change the grid color in the **Colors** page of the preferences dialog.

The grid can be shown or hidden using the  button on the left-hand toolbar. By default the grid is still active even if it is hidden, but this is configurable in the **Display Options** preferences page. There you can set the grid to be disabled when it is hidden or even disable the grid entirely.

## Editing object properties

All objects have properties that are editable in a dialog. Use the hotkey **E** or select **Properties** from the right-click context menu to edit the properties of selected item(s). You can only open the properties dialog if all the items you have selected are of the same type. For many object types, like footprints, you can only edit the properties of a single item at one time. To edit the properties of multiple items at once, including items with different types, you can use the Properties Manager.



You can also view and edit item properties using the Properties Manager. The Properties Manager is a docked panel that displays the properties of the selected item or items for editing. If multiple types of items are selected at once, the properties panel displays only the properties shared by all of the selected item types.

Properties

Footprint

Basic Properties

Position X	150.75 mm
Position Y	98.75 mm
Locked	<input type="checkbox"/>
Layer	B.Cu
Orientation	-135°

Fields

Reference	U1
Value	MaxLinear_XR22417-48
Library Link	footprints:LQFP-48_7x7mm_P0.5mm
Library Description	LQFP, 48 Pin ( <a href="https://www.analog.com/me">https://www.analog.com/me</a> )
Keywords	LQFP QFP
Component Class	

Attributes

Not in Schematic	<input type="checkbox"/>
Exclude From Position Files	<input type="checkbox"/>
Exclude From Bill of Materials	<input type="checkbox"/>
Do not Populate	<input type="checkbox"/>

Overrides

Exempt From Courtyard Requirement	<input type="checkbox"/>
Clearance Override	
Solderpaste Margin Override	
Solderpaste Margin Ratio Override	-0.125
Zone Connection Style	Inherited

Editing a property in the Properties Manager immediately applies the change. When multiple items are selected, property modifications are applied to each selected item individually, not to the whole selection as a group. For example, when changing the orientation of multiple items, each item is individually rotated around its own origin, not the group's origin.

Show the Properties Manager with **View** → **Panels** → **Properties** or the  button on the left toolbar.

Several tools are available for editing properties of specific types of objects in bulk. For text and graphical items, including footprint fields and dimensions, you can use the [Edit Text and Graphics Properties tool](#). Tracks and vias can be bulk-edited using the [Edit Track and Via Properties tool](#). Teardrop properties can be edited with the [Edit Teardrops tool](#).

In properties dialogs and many other dialogs, any field that contains a numeric value can also accept a basic math expression that results in a numeric value.

For example, a dimension may be entered as `2 * 2mm`, resulting in a value of `4mm`. Basic arithmetic operators as well as parentheses for defining order of operations are supported. Units can also be specified, and unit conversions are performed automatically, so `1in + 1mm` evaluates to `26.4mm`.

## Board outlines (Edge Cuts)

KiCad uses graphical objects on the `Edge.Cuts` layer to define the board outline. The outline must be a continuous (closed) shape, but can be made up of different types of graphical object such as lines and arcs, or be a single object such as a rectangle or polygon. If no board outline is defined, or the board outline is invalid, some functions such as the 3D viewer and some design rule checks will not be functional.

For the board outline to be considered valid, the endpoints of any shapes in the outline must coincide *exactly*. If any endpoints are not coincident with another endpoint, the outline will not be considered closed. Outline shapes also cannot intersect each other or overlap. In such cases, [DRC will report a "Board has malformed outline" violation](#) that points to the problematic parts of the outline.

### NOTE

You can use the [grid or the snapping tools](#) to ensure outline endpoints exactly coincide. The [Heal Shapes](#) tool can also be used to fix small gaps between endpoints.

If there are multiple closed shapes on the `Edge.Cuts` layer, each shape acts as an independent board outline. When an outline shape completely encloses another outline, the outermost shape is considered the outside edge of the board. Any closed shapes inside the outer shape are considered interior cutouts in the board. Each closed outline cannot intersect or overlap with other outlines.

Zones only fill when they are within the board outline. Any portion of a zone that is outside of the board outline, including inside an interior cutout, will not be filled.

## Working with footprints

### Adding footprints to the board

Footprints are automatically added to the board when the PCB is [updated from the schematic](#). The footprint associated with each schematic symbol is added to the board if it is not already present, and each footprint pad is associated with the corresponding symbol pin's net. Symbol pins are matched to footprint pads by pin/pad number.

When footprints are added to the board after an update from the schematic, they are clustered by schematic sheet and by geographical location in the schematic. They are initially attached to the cursor; you can place them by clicking in the desired location.

You can also add footprints to the board manually using the [Add Footprint tool](#) ( or the  button).

### NOTE

Footprints added in this way will not be automatically associated with a symbol or have nets assigned to their pads, and subsequent updates from the schematic will remove these unassociated footprints unless the footprint is locked or the **Delete footprints with no symbols** option is unchecked in the Update PCB From Schematic dialog. For these reasons, it is usually recommended to avoid manually adding footprints to the board. Manually adding footprints is necessary for [PCB-only workflows](#), and can also be useful for adding logos or other footprints that do not need a corresponding schematic symbol.

## Placing and moving footprints

Once footprints have been added to the board, you can reposition them in many ways.

The Move command (**M**) moves a footprint or a selection of footprints, ignoring any connected track segments that are not selected. No DRC checking is done when moving footprints with the Move command, although any footprint courtyards that collide with the moved footprint's courtyard will be highlighted.

There is a reference point for the move operation, which is the point in the footprint which attaches to the cursor and therefore the point in the footprint that snaps to the grid and to other objects. The reference point during a move is determined by the location of the cursor when the Move command is initiated. If the cursor is over a pad, the pad's center will be used as the reference point. If the cursor is not over a pad, the footprint's anchor (coordinate origin point) will be used. To select an arbitrary snapping point, you can use the Move With Reference command instead of the regular Move command (right click → **Move with Reference**). After initiating the command, click on the desired reference point; KiCad will then begin the move with that point as the reference.

You can also use the Drag command (**D**) to move the selected footprint using the interactive router, maintaining all track connections to the footprint. Dragging footprints behaves like the Highlight Collisions router mode: obstacles will not be avoided or shoved, only highlighted. Ordinarily the router will prevent you from dragging a footprint into a position that violates DRC: when you click to commit a drag in a position that violates DRC, the footprint will return to its original position. To force a drag to be committed even if it violates DRC, **Ctrl**-click to commit the drag. Like the Move command, colliding courtyards are highlighted.

### NOTE

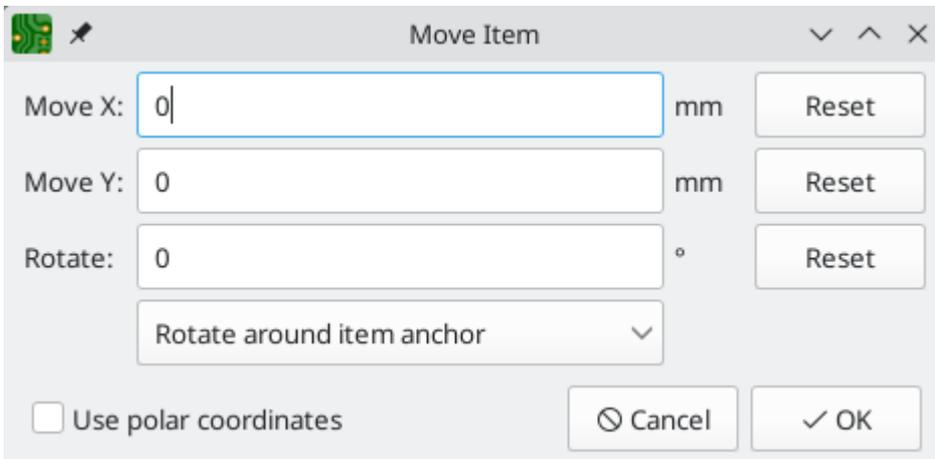
Only tracks that end at the origin of the footprint's pads will be dragged. Tracks that simply pass through the pad or that end on the pad at a location other than the origin will not be dragged.

You can move a footprint to the opposite side of the board with the Flip command (**F**). Any parts of the footprint on a front layer will be swapped to the corresponding back layer, and vice versa.

Footprints can be rotated counter-clockwise using the **R** hotkey, or clockwise using **Shift** + **R**. By default, footprints are rotated by 90 degrees every time the rotate command is used, but you can configure the rotation angle step in **Preferences** → **PCB Editor** → **Editing Options**.

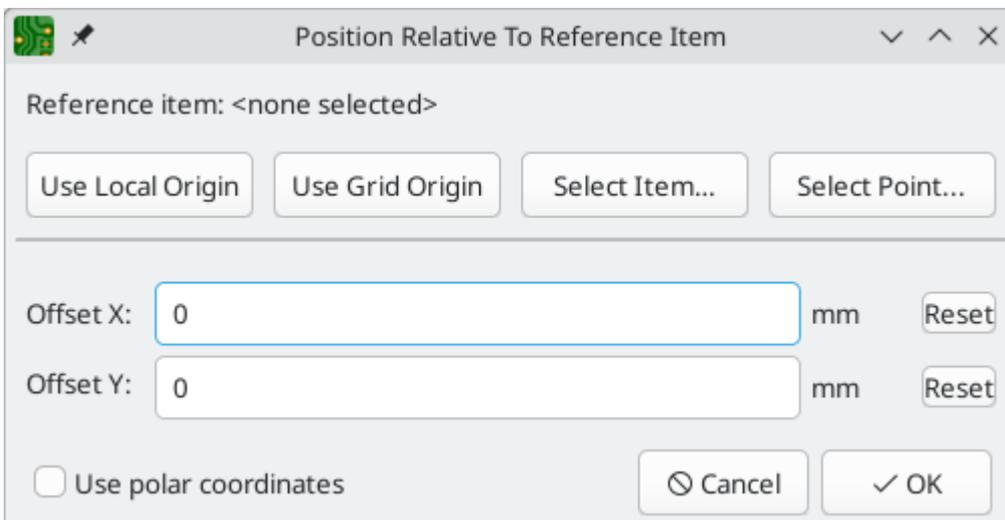
You can directly set a footprint's exact absolute position, rotation angle, and PCB side using either the Footprint Properties dialog or the Properties panel.

To reposition a footprint relative to its current position, use the Move Exactly tool (**Shift** + **M**) or right click → **Position** → **Move Exactly...**. The dialog lets you specify an X and Y translation, as well as a rotation, that will be applied to the footprint. The rotation can be performed relative to either the footprint's anchor, the local coordinate origin, or the drill/place origin. You can also use polar coordinates instead of Cartesian coordinates.



To position a footprint relative to another object, you can use the Position Relative tool (**Shift** + **P** or right click → **Position** → **Position Relative To...**). With this tool, you select a reference point for the move and specify an offset. The footprint is moved to the specified offset relative to the reference point. The reference point can be one of the following:

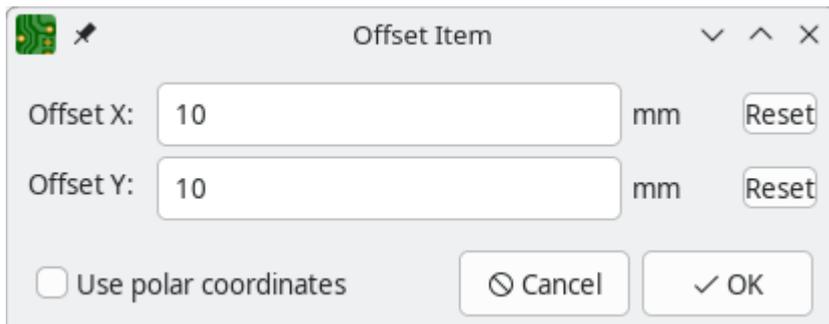
- The local origin, which is set to the cursor position when you press **Space**.
- The grid origin, which is configured in the Grids dialog.
- The location of an arbitrary item on the board, such as a specific pad in a footprint. After clicking the **Select Item...** button, click on the desired board item in the canvas to set the reference point.
- An arbitrary point in the canvas. After clicking the **Select Point...** button, click at the desired location to set the reference point. You can use object snapping to select a specific point in an object, such as the end of a graphic line.



To position a footprint such that an arbitrary point in the footprint is positioned a certain distance from another arbitrary reference point, you can use the Interactive Offset tool (right click a footprint → **Position** → **Interactive Offset Tool...**).

This tool lets you interactively select two points that form the start and end of a position vector. The first point is a reference point in the footprint, and will move along with the footprint. The second point is a fixed reference that will remain stationary when the footprint is moved. The vector from the first point to the second point is shown graphically in the editing canvas. You can then give new X and Y (or polar) dimensions for the vector, which will move the footprint reference relative to the fixed reference such that the fixed

reference is the specified distance from the footprint reference point. The dialog initially contains the vector dimensions before any move is performed, or in other words the initial distance between the footprint reference point to the fixed reference.



You can swap the position of two selected footprints using the Swap command (**Alt** + **S**). The first footprint is assigned the location, rotation, and board side of the second footprint, and vice versa. If there are more than two footprints selected, the locations are cycled: the last footprint gets the position of the first footprint, the first footprint gets the location of the second, and so on.

There are several convenience features that make it easier to find, select, and move specific footprints or footprints related to another footprint.

The Get and Move Footprint command (**T**) prompts you to choose a footprint from a list or by typing a reference designator. KiCad then attaches the chosen footprint to your cursor for a move operation.

There are two commands to select other footprints that need to be connected to the selected footprint but don't yet have routed connections. The Select All Unconnected Footprints command (**U**) selects all footprints that have ratsnest lines to the currently selected footprints. The command can be executed repeatedly to further expand the selection based on the newly selected items. The Grab Nearest Unconnected Footprint command (**Shift** + **U**) selects the closest footprint with ratsnest lines to the currently selected footprint, and additionally begins to move it. If there are multiple footprints initially selected, the command will act like the Move Individually command described below, individually moving the closest unconnected footprint for each of the initially selected footprints.

You can select footprints based on their schematic sheet using the right click → **Select** → **Items in Same Hierarchical Sheet** command, which selects all other footprints that are in the same schematic sheet as the originally selected footprint.

If you want to move multiple selected footprints in sequence, use the Move Individually command (**Ctrl** + **M**). After triggering the command, KiCad will begin moving the first selected footprint. After you click to place the footprint, KiCad will immediately start moving the next footprint, in the same order that you selected the footprints. You can skip moving a footprint by pressing **Tab**, commit the current move and skip any remaining moves by double-clicking, or cancel all moves (including those already completed) by pressing **Esc**.

If you want to move a collection of footprints at once into one area, the Pack and Move Footprints command (**P**) closely packs the selected footprints together and moves them as a block.

**TIP**

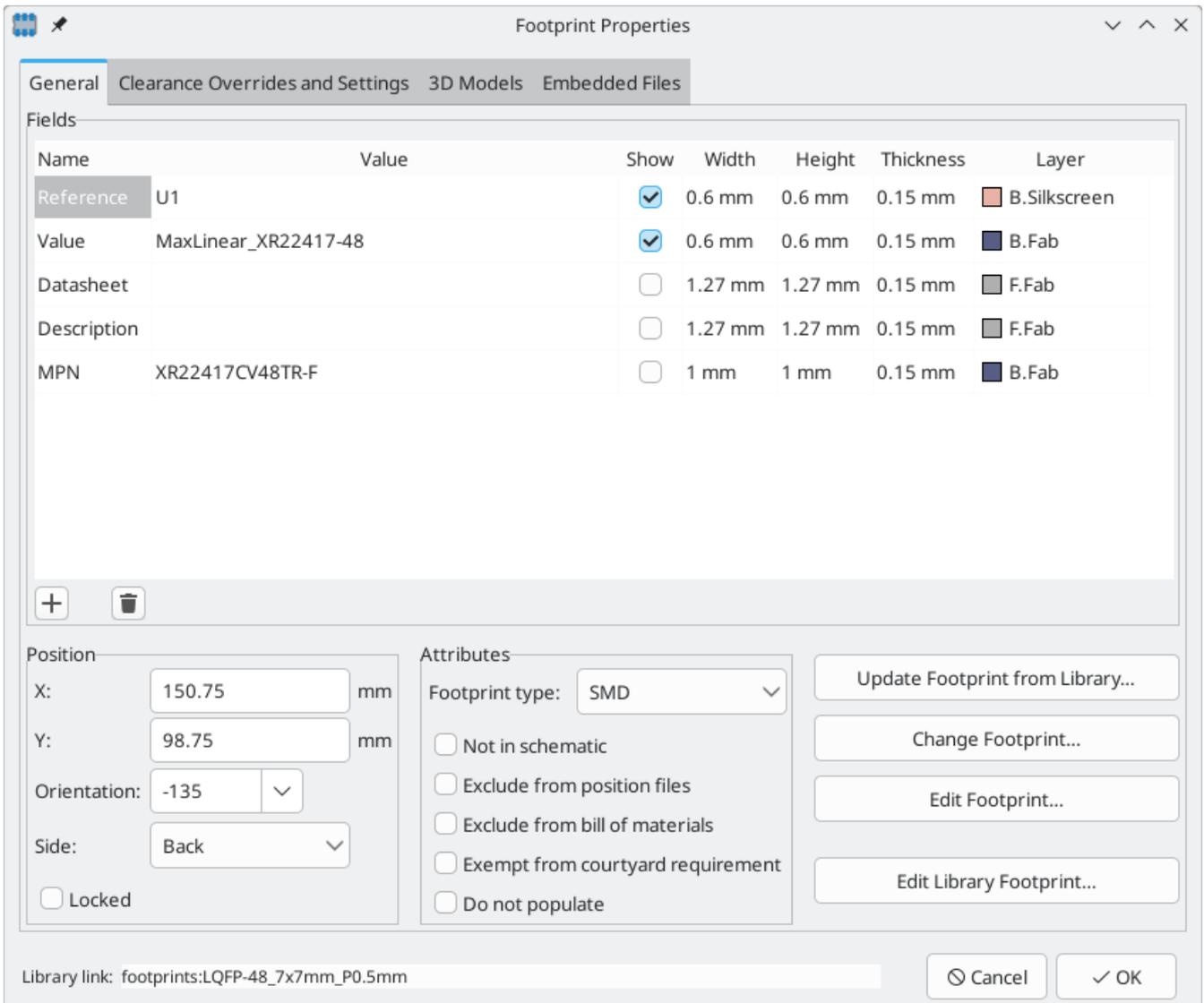
Move Individually and Pack and Move Footprints are useful in combination with other selection convenience features, such as cross-selection from the schematic or the advanced footprint selection features described above. For example, you could select a group of bypass capacitors in the Schematic Editor, switch to the PCB Editor where the corresponding footprints are now selected, and then use Move Individually to quickly place all of the bypass capacitor footprints close to their respective ICs. Alternatively, you could use one of the other selection tools, such as Select All Unconnected Footprints, to select many footprints from all over the board, then use Pack and Move Footprints to quickly put them all into a small area.

Finally, KiCad can automatically place footprints onto the board. The auto-place function attempts to optimally place footprints to simplify ratsnest connections to other footprints. You can auto-place the selected footprints with **Place** → **Auto-Place Footprints** → **Place Selected Footprints**, or auto-place all footprints outside of the board outline with **Place** → **Auto-Place Footprints** → **Place Off-Board Footprints**.

## Editing Footprints

Footprints in the board can be individually edited, both in terms of their properties (fields, attributes, clearance settings, etc.) and in terms of their physical pads and graphics. Editing a footprint in the board only affects that particular instance of the footprint; it does not affect any other copies of that footprint in the board, and it does not affect the library footprint.

To edit the properties of a footprint in the board, open its properties dialog ()



The majority of the settings in this dialog are the same as in the [footprint editor](#). You can edit the footprint's fields, attributes, clearance and zone connection settings, 3D models, and [embedded files](#), as in the footprint editor. However, here you can also set the footprint's position, orientation, and side. You can also update the footprint from the library, exchange it for a different footprint, or edit the footprint itself in the footprint editor.

To edit the footprint's physical form, i.e. its pads and graphics, you need to use the [footprint editor](#). There are two buttons for opening a footprint in the editor, depending on whether you want to edit a single copy of a footprint in the board or a footprint's source copy in the library.

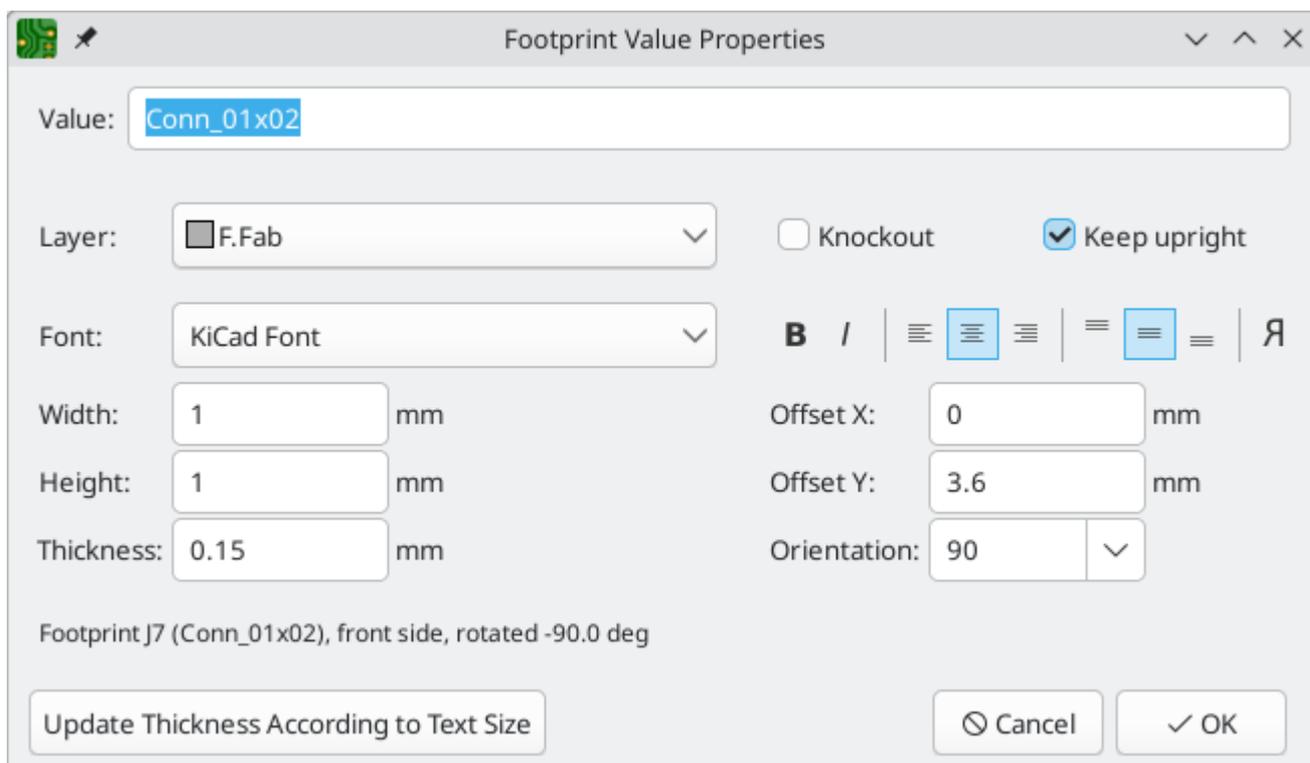
- **Edit Footprint...** will open the specific instance of the footprint in the footprint editor. Editing this footprint will only affect this one instance of the footprint in the board. It will not affect other instances of the footprint in the board, and it will not affect the library copy of the footprint. You can also open a board footprint in the footprint editor by right clicking the footprint in the board and selecting **Open in footprint editor** (**Ctrl** + **E**).
- **Edit Library Footprint...** will open the library copy of the footprint in the footprint editor. Editing the library copy of the footprint will edit the footprint in the footprint library, but will not immediately affect any instances of that footprint in the board. To update footprints in the board with changes to the library footprint, use the **Update Footprint from Library...** tool. Editing the library footprint in this way

is equivalent to opening the footprint editor, opening the appropriate footprint in its library, and editing it.

The **Update Footprint from Library...** button is used to update the board's copy of the footprint to match the copy in the library. The **Change Footprint...** button is used to swap the current footprint to a different footprint in the library. These functions are described [later](#).

## Editing footprint fields

An individual symbol text field can be edited directly with the **E** hotkey (with a field selected instead of a footprint) or by double-clicking on the field.



The options in this dialog are the same as those in the full Footprint Properties dialog, but are specific to a single field.

Only footprint fields can be edited this way in the board editor. Unlike fields, Footprint text is a graphic object that can only be edited or moved in the footprint editor.

### NOTE

In versions of KiCad before version 8.0, footprint fields did not exist. Instead, footprint text could be edited directly in the board editor. In KiCad 8.0, footprint text is not editable in the board editor and can only be edited in the footprint editor.

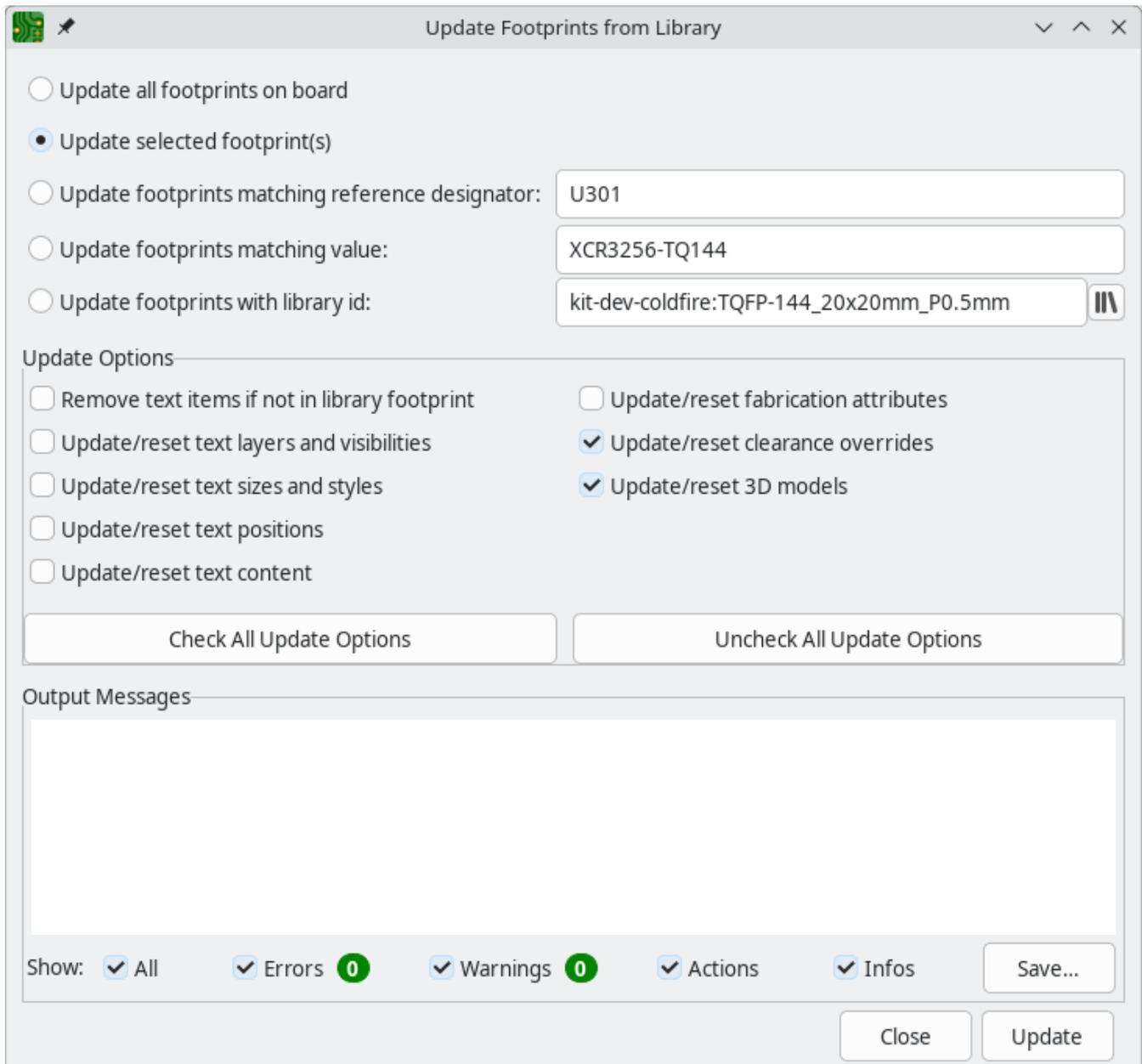
## Updating and exchanging footprints

When a footprint is added to the board, KiCad embeds a copy of the library footprint in the board so that the board is independent of the system libraries. Footprints that have been added to the board are not automatically updated when the library changes. Library footprint changes are manually synced to the board so that the board does not change unexpectedly.

### NOTE

You can use the [Compare Footprint with Library tool](#) to inspect the differences between a footprint in a board with its corresponding library footprint.

To update footprints in the board to match the corresponding library footprint, use **Tools** → **Update Footprints from Library...**, or right click a footprint and select **Update Footprint...**. You can also access the tool from the [footprint properties dialog](#).

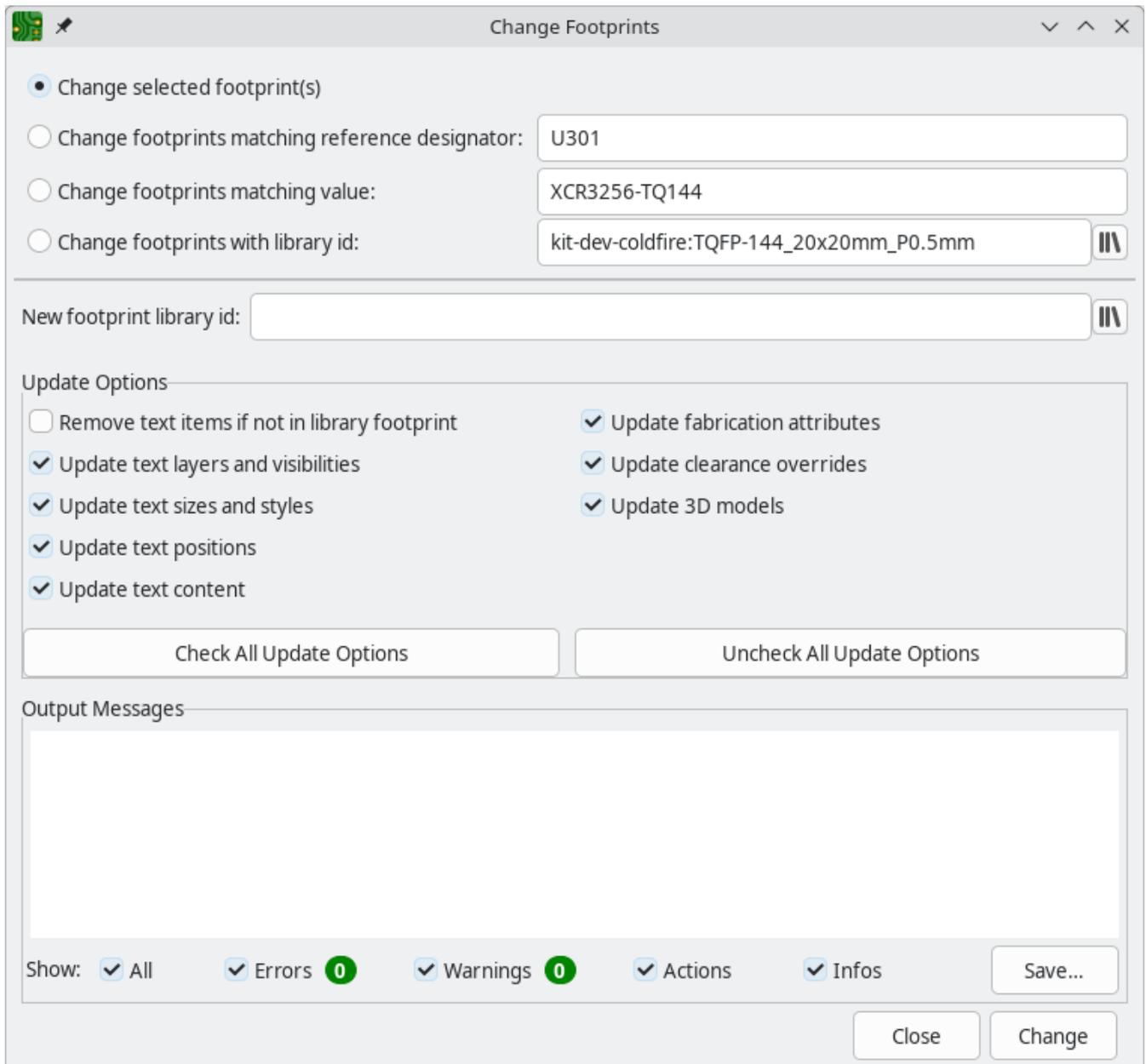


The top of the dialog has options to choose which footprints will be updated. You can update all footprints on the board, update only the selected footprints, or update only the footprints that match a specific reference designator, value, or library identifier. The reference designator and value fields support wildcards: **\*** matches any number of any characters, including none, and **?** matches any single character.

The middle of the dialog has options to control what parts of the footprint will be updated. You can select specific fields to update or not update, which properties of the fields to update (text content, visibility, size and style, and position), and how to handle fields that are missing or empty in the library footprint. You can also choose whether to update clearance overrides and footprint attributes, such as footprint type, **not in schematic, exclude from position files / bill of materials, exempt from courtyard requirement, and do not populate**.

The bottom of the dialog displays messages describing the update actions that have been performed.

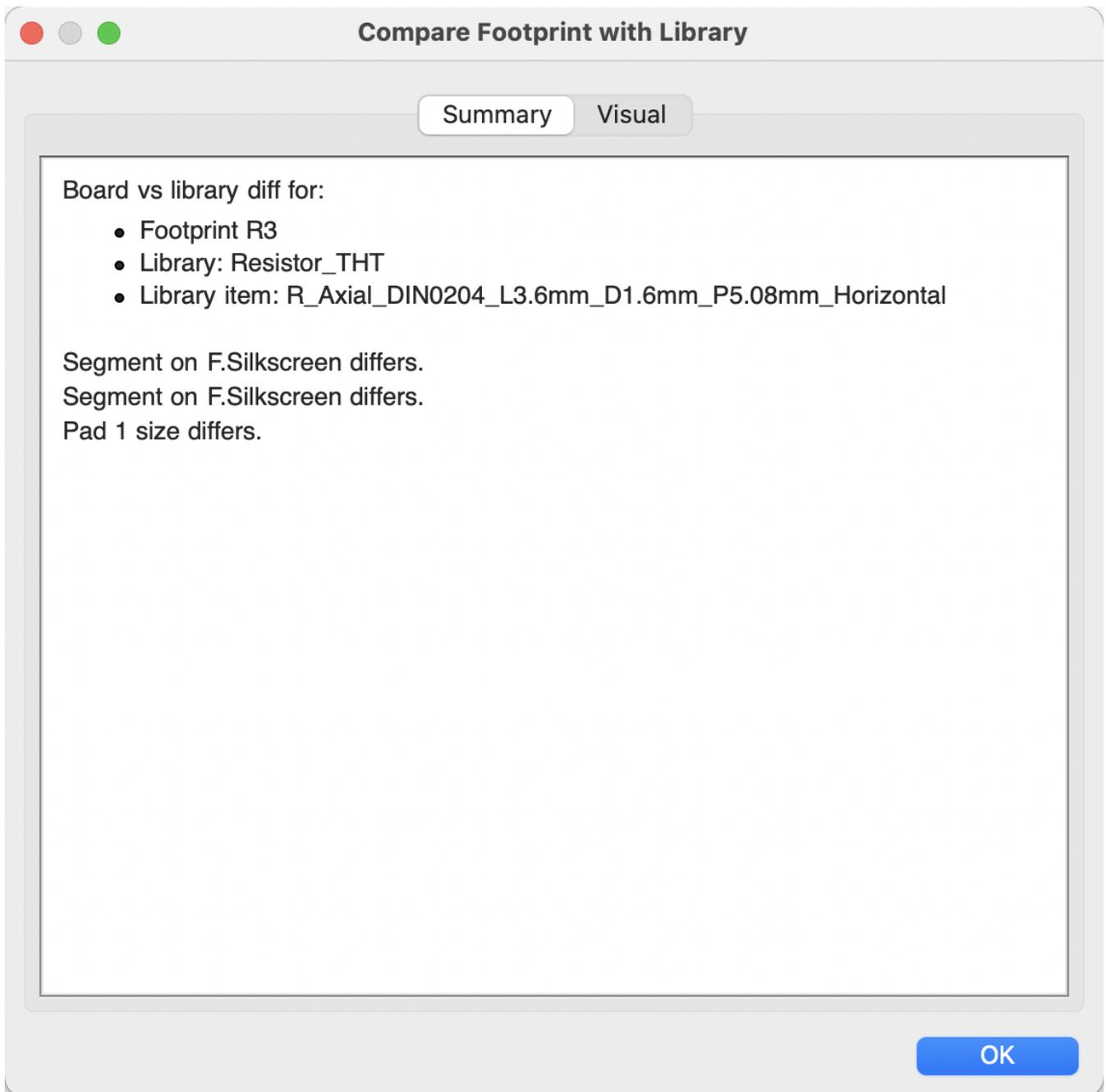
To change an existing footprint to a different footprint, use **Edit** → **Change Footprints...**, or right click an existing footprint and select **Change Footprint...**. This dialog is also accessible from the [footprint properties dialog](#).



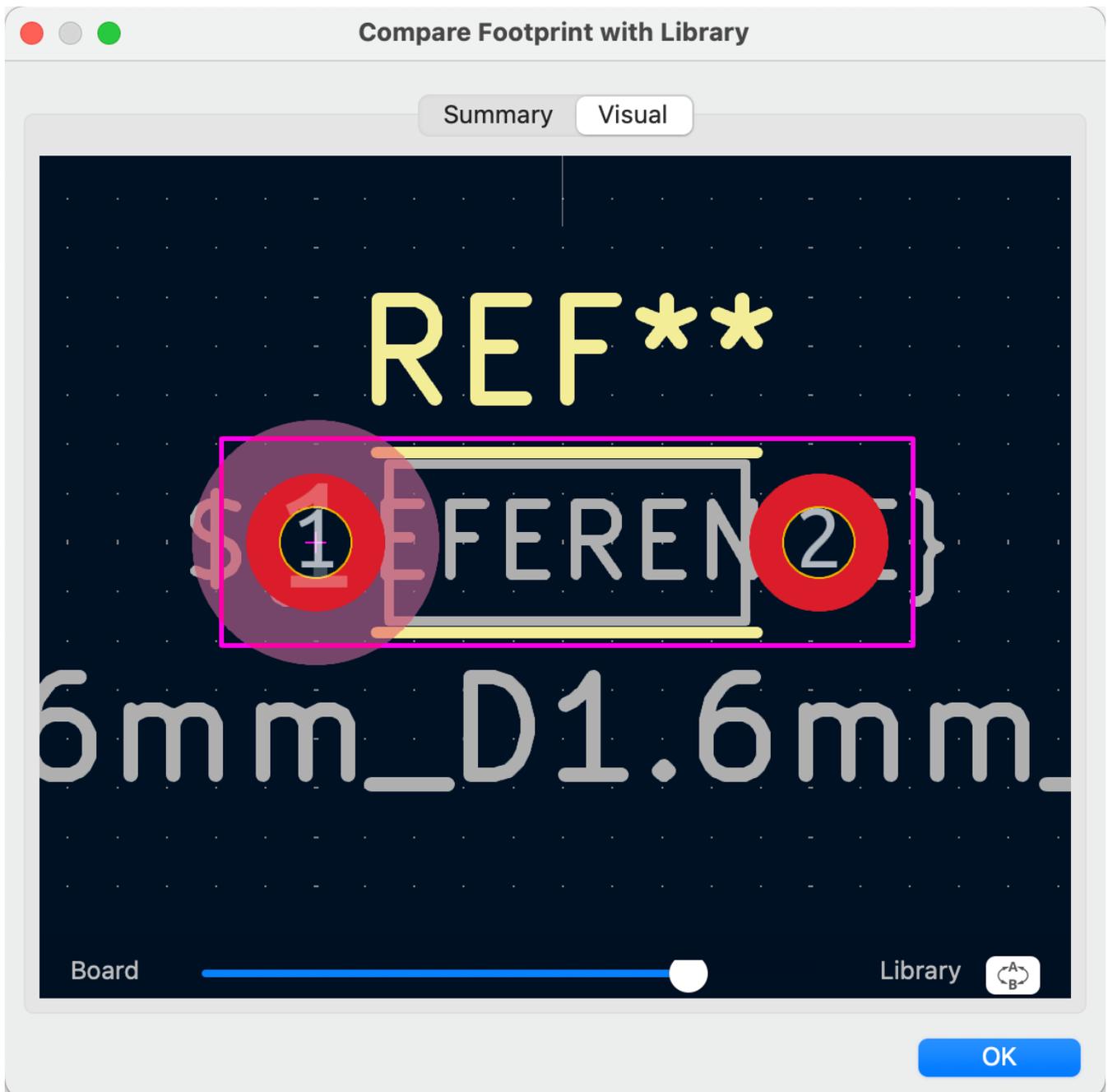
The options for the Change Footprints dialog are very similar to the Update Footprints from Library dialog.

## Comparing footprints between board and library

When a footprint in a board diverges from the corresponding footprint in the original footprint library, you can use the Compare Footprint with Library tool to inspect the differences between the two versions of the footprint. Run the tool using **Inspect** → **Compare Footprint With Library**.



The **Summary** tab shows the name of the footprint, including its library and board reference designator, and provides a list of the differences between the board and library versions of the footprint.



The **Visual** tab shows a visual comparison of the board and library versions of the footprint. This can be used as a visual diff tool.

By default, the comparison displays both versions of the footprint superimposed on each other. To see the changes more easily, you can drag the slider at the bottom of the tab to the right to emphasize the library version of the footprint in the superimposed view (making the board version of the footprint more transparent) or drag it to the left to emphasize the board version (making the library version more transparent). At the far right and left ends of the slider, the board and library versions of the footprint, respectively, are fully hidden. It may be helpful to drag the slider back and forth to see the changes more clearly.

You can press the **A/B** button, or use the  hotkey, to quickly toggle back and forth between the board and library versions.

The screenshot above shows a visual comparison with the board version of the footprint deemphasized. Looking at pad 1 on the left, you can see a large, partially transparent pad (from the board footprint)

surrounding a fully opaque, smaller pad (from the library footprint). This indicates that the pad was enlarged in the board version of the footprint, or shrunk in the library version of the footprint.

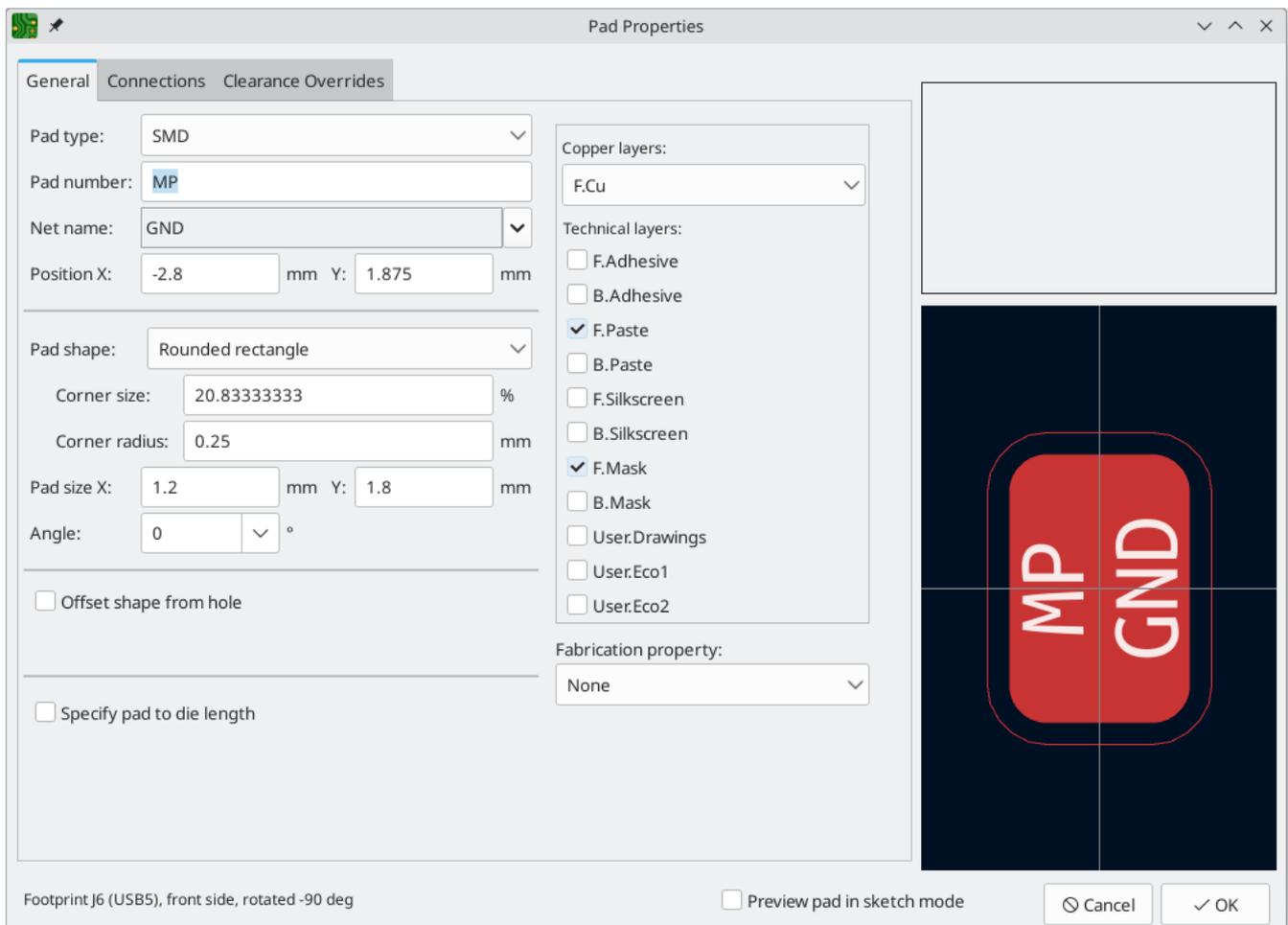
## Working with pads

The properties of each individual pad of a footprint can be inspected and edited after placing the footprint on the board. In other words, it is possible to override the design of an individual footprint pad in a specific instance of the footprint on the board, if the footprint design in the library is not appropriate. For example, you may wish to remove the solder paste aperture for a pad that needs to remain unsoldered in a specific design, or you may wish to move the location of a through-hole pad for an axial-lead resistor in order to fit a specific design.

### NOTE

By default, the position of all footprint pads are locked, so it is possible to edit the pad properties but not move the pad's location relative to the rest of the footprint. Pads may be unlocked to allow free movement, which can be useful for certain applications (such as through-hole footprints with varying lead positions) but is generally never recommended for surface-mount footprints.

The pad properties dialog is opened through the context menu or default hotkey **E** when a pad is selected. Note that KiCad assumes that if you click near a pad, you are probably trying to select the entire footprint rather than a single pad. To select a single pad, make sure to click inside the pad area, or turn off the Footprints setting in the selection filter (and make sure the Pads setting is turned on) to prevent accidental selection of the entire footprint rather than a specific pad.



This dialog lets you edit the physical properties of the pad, including size and shape. You can also modify how the pad connects to other objects on the board, including clearance properties, teardrops, and thermal reliefs.

This dialog is the same as the pad properties dialog in the footprint editor, except that here you can also manually assign a net to a pad using the **net name** selector. The remaining options are explained in the [Footprint Editor documentation](#).

**NOTE**

While you can manually assign nets to pads in the PCB editor, this is not a typical workflow. Usually net-to-pad connections are defined by the schematic and then [transferred to the PCB editor](#).

## Working with zones

Copper zones, also sometimes called copper pours or fills by other EDA tools, are solid or hatched areas of copper assigned to a particular net that automatically keep clearance from other copper objects. Zones are commonly used to fill in all free space on a board layer (or a portion of a layer) in order to create ground and power planes, carry high currents, or to provide shielding.

**NOTE**

Some EDA tools have separate tools for creating "plane layers" and for creating copper zones on signal layers. In KiCad, the Copper Zone tool is used for both these applications.

Zones are defined by a polygonal **outline** that defines the maximum extent of the filled copper area. This outline does not represent physical copper and will not appear in exported manufacturing data. The actual copper areas of the zone must be **filled** each time the outline, or any objects inside the outline, are modified. The filling process may be run on a single zone, or on all zones in a board (default hotkey **B**). Zones may be **unfilled** (default hotkey **Ctrl + B**) to improve performance and reduce visual clutter while editing large boards.

**NOTE**

By default, zone filling is a manual process rather than occurring every time an object changes that would result in a change to the zone copper. This is because zone filling can be a slow process on older computers or very large designs. It is important to make sure zone fills are up-to-date before generating outputs. KiCad will check that zones have been updated and warn you before generating outputs or running DRC when zones have not yet been refilled. You can optionally enable automatic zone-filling in the Preferences dialog (**PCB Editor** → **Editing Options** → **Miscellaneous** → **Automatically refill zones**).

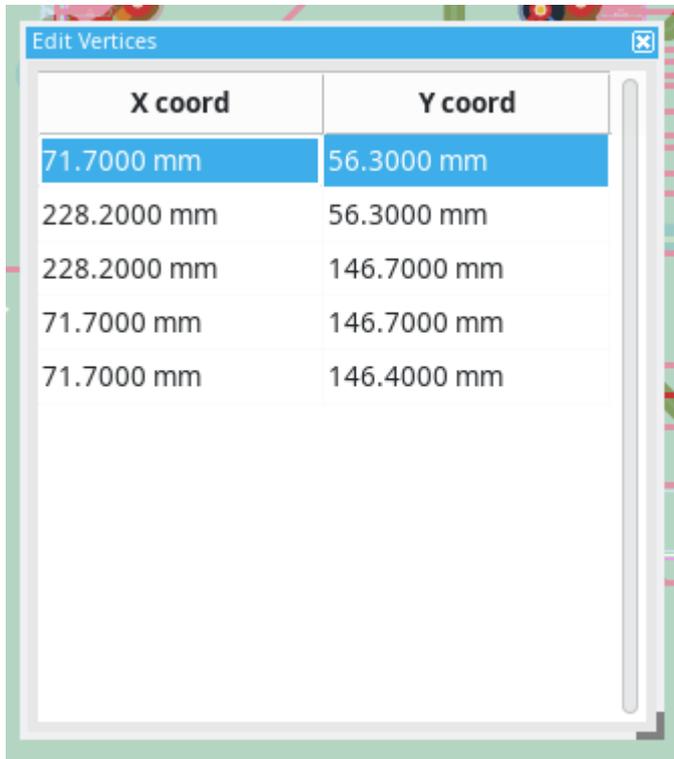
A zone fill occupies any unused space within the zone outline, automatically maintaining a specified clearance to board edges, holes, and copper objects on different nets. Zones do not fill outside of the [board outline](#) or within interior cutouts.

To draw a zone, click the Add Filled Zone tool () on the right toolbar, or use default hotkey **Ctrl + Shift + Z**. Click to choose the first point of the zone outline. The Zone Properties dialog will appear, allowing you to choose the zone net and other properties. These properties may be edited at any time, so it is not critical to choose them all correctly at first. Accept the dialog and continue placing points to define the zone outline. To finish the zone, double-click to set the last point.

To modify an existing zone outline, select it, then drag its editing handles to change the shape. Moving a handle at the corner of a zone will move that corner. Moving a handle on the edge of a zone will move that edge in a direction perpendicular to the edge. Normally, dragging an edge maintains the angles of the corners

adjacent to the edge while allowing the edge's length to vary. Holding **Ctrl** instead holds the edge's length constant and allows the adjacent corner angles to vary.

To precisely position a corner, right click the corner's handle and choose **Shape Modification** → **Move Corner To...**, then enter new X and Y coordinates for the corner. You can also edit the coordinates of every outline corner by right clicking the zone and choosing **Shape Modification** → **Edit Corners...** This opens a floating dialog with a table containing the coordinates of every corner. Editing the coordinates of a corner immediately updates the zone outline.



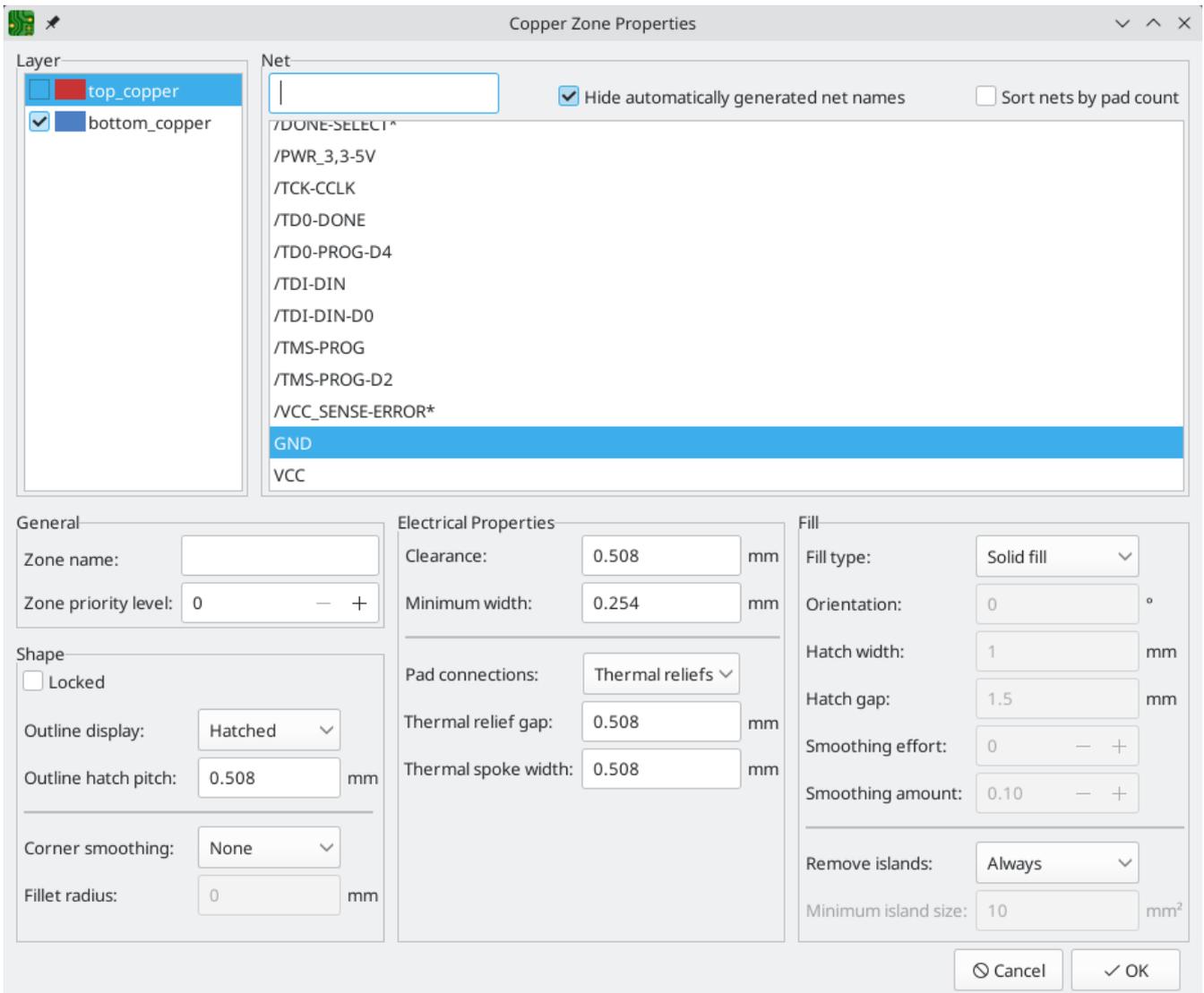
The image shows a floating dialog box titled "Edit Vertices" with a close button in the top right corner. It contains a table with two columns: "X coord" and "Y coord". The table lists five vertices with their respective coordinates in millimeters. The first row is highlighted in blue.

X coord	Y coord
71.7000 mm	56.3000 mm
228.2000 mm	56.3000 mm
228.2000 mm	146.7000 mm
71.7000 mm	146.7000 mm
71.7000 mm	146.4000 mm

**NOTE**

You can also create zones by [converting an existing graphic shape to a zone](#). This can be useful, for example, for creating a zone with a shape that would otherwise be difficult to draw with the zone tool, such as a circle. To convert a shape to a zone, right click the shape, then select **Create from Selection** → **Create Zone from Selection...**

To edit the zone's properties, use hotkey **E** or select Properties from the context menu.



**Layer:** A single zone object can create filled copper on one or more copper layers. Check the box next to each copper layer that this zone outline should fill on. The copper on each layer will be filled independently, but all layers will share the same net.

**Net:** Select the electrical net that the zone copper should be connected to. It is possible to create zones with no net assignment. Zones with no net will keep clearance from any copper objects on any net.

**Zone name** can be used to assign a specific name to a zone. This name can be used to refer to the zone in custom DRC rules.

**Zone priority level** determines the order in which multiple zones on a single layer are filled. The highest priority level zone on a given layer will be filled first. Lower-priority zones will keep clearance to the filled areas of higher-priority zones. Two zones on the same layer with the same priority level will overlap (short-circuit) with each other, unless they are assigned different nets. When two zone outlines with the same priority and different nets touch, one zone will maintain clearance to the other so that they don't short.

**Locked** controls whether or not the zone outline object is **locked**. Locked objects may not be manipulated or moved, and cannot be selected unless the **Locked Items** option is enabled in the Selection Filter panel.

**Outline display** controls how the zone outline is drawn on screen. In **Line** mode, only the border lines of the outline are drawn. In **Hatched** mode, hatch lines are drawn on the inside of the outline border for a

short distance, to make the zone outline more apparent. In **Fully Hatched** mode, hatch lines are drawn across the entire inside of the zone outline.

**Corner smoothing** controls the behavior of the filled copper areas at corners of the outline. Corners can be smoothed by a chamfer or fillet, or can extend all the way to the outline corner if smoothing is disabled. The chamfer or fillet size is configurable when those modes are selected.

**NOTE**

By default, chamfers and fillets are not added to **inside corners** of the zone outline, because this would result in filled copper extending *outside* the outline. If smooth inside corners are desired, enable the **Allow fillets outside zone outline** option in the Constraints section of the Board Setup dialog.

**Clearance** controls the minimum clearance the filled areas of this zone will keep from other copper objects. Note that if two clearance values are in conflict, the larger clearance value will be used. For example, if a zone is set to use 0.2mm clearance but its netclass is set to use 0.3mm clearance, the result will be an 0.3mm clearance.

**Minimum width** controls the minimum size of narrow necks of copper created inside the zone. Any copper areas that would be below this minimum width are removed during the filling process.

**Pad connection** controls the way that the filled zone areas will connect to footprint pads on the same net. **Solid** connections will result in the copper completely overlapping the pads. **Thermal reliefs** will result in small copper spokes connecting the pad to the rest of the copper zone, increasing the thermal resistance between the pad and the rest of the zone. This can be useful for hand soldering. **Reliefs for PTH** will apply thermal reliefs to plated through-hole pads and use solid connections for surface mount pads. **None** will result in the zone not connecting to any pads on the same net.

**Thermal relief gap** controls the distance maintained between any pad and the copper zone when the pad connection mode is set to generate thermal reliefs.

**Thermal spoke width** controls the width of the "spokes", or short copper segments connecting the pad to the rest of the copper zone.

**Fill type** controls how the copper zone is filled: the default is **solid fill**, which will result in copper filling in all available space within the zone outline. The zone can also be set to fill a **hatch pattern**, which will fill the area with a pattern that contains less copper. This can be useful for flexible printed circuits and other specialty applications.

**Orientation** controls the angle of the hatch pattern lines. An orientation of 0 degrees will result in the hatch pattern using horizontal and vertical lines.

**Hatch width** controls the width of each line in the hatch pattern.

**Hatch gap** controls the distance between each line in the hatch pattern.

**Smoothing effort** controls the style of smoothing applied to the hatch pattern. A value of 0 will result in no smoothing, and a value of 3 will result in the finest smoothing. Higher values will result in longer processing time and larger Gerber files.

**Smoothing amount** is a ratio that controls the size of the smoothing chamfers or fillets that are generated when **smoothing effort** is set to a value other than 0. An amount of 0.0 results in no smoothing, and a value

of 1.0 results in maximum smoothing (in other words, a chamfer or fillet equal to half of the hatch gap).

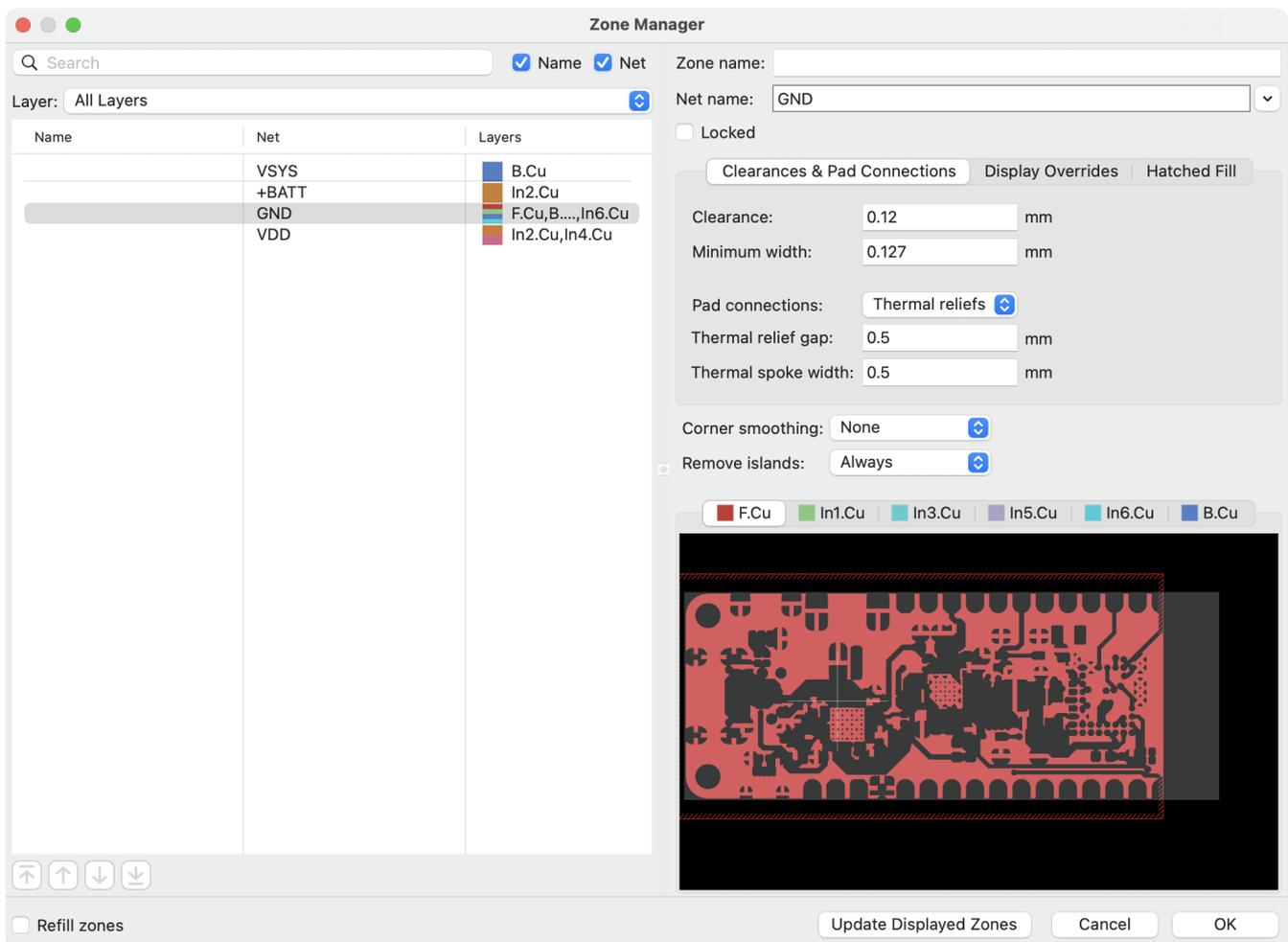
**Remove islands** controls the behavior of isolated copper areas, also called islands, after the initial zone fill. When this is set to **always**, isolated areas inside the zone are removed. When set to **never**, isolated areas are left alone, and will result in copper areas that are not connected to the rest of the net. When set to **below area limit**, a **minimum island size** can be specified, and islands below this threshold will be removed.

**NOTE**

Regardless of the **remove islands** setting, islands are never removed from zones that are electrically unconnected. In other words, islands are only removed from zones that have at least one electrical connection.

## Managing zones

Instead of editing a single zone with the Zone Properties dialog, you can use the Zone Manager tool to view, edit, and prioritize all zones in the board at once. To run the Zone Manager, click **Tools** → **Zone Manager**.



The left side of the dialog shows a list of all zones in the board, displaying the name (if any), net, and layers for each zone. The order of the zones in the list reflects the priority of each zone: higher priority zones are higher in the list. To change the priority of a zone, use the **↑** and **↓** buttons to move it up or down in the list. Use the **↕** and **↕** buttons to move it to the very highest or lowest priority.

You can filter the list of zones by typing into the filter box. The filter matches against the zones' name and/or net, depending on which filter options are enabled. You can also filter the list by zone layer using the **Layer** dropdown menu.

Selecting a zone in the list shows a preview of that zone in the bottom right. The preview can be zoomed and panned using the same controls as the PCB Editor canvas.

**TIP**

You can reset the preview to show the entire zone by right clicking the preview and choosing **Zoom to Fit**.

If the selected zone spans multiple layers, each layer is shown individually. You can preview each layer by clicking the appropriate layer tab above the preview.

The right side of the dialog shows the settings for the selected zone, which are explained [above](#). You can preview the new settings by clicking the **Update Displayed Zones** button, which updates the zone preview without affecting the board. Changing the properties of a zone in the Zone Manager will not update the board until you press **OK**. If the **Refill zones** option is enabled, all zones will be refilled when you accept the dialog. If **Refill zones** is not enabled, zones will not be refilled until you manually refill them.

## Routing tracks and vias

KiCad features an interactive router that:

- Allows manual or guided (semi-automatic) routing of single tracks and differential pairs
- Enables modifications of existing designs by:
  - Re-routing existing tracks when they are dragged
  - Re-routing tracks attached to footprint pads when the footprint is dragged
- Allows tuning of track lengths and differential pair skew (phase) by inserting serpentine tuning shapes for designs with tight timing requirements

By default, the router respects the configured design rules when placing tracks: the size (width) of new tracks will be taken from the design rules and the router will respect the copper clearance set in the design rules when determining where new tracks and vias can be placed. It is possible to disable this behavior if desired by using the Highlight Collisions router mode and turning on the Allow DRC Violations option in the router settings (see below).

The router has three modes that can be selected at any time in the [Interactive Router Settings dialog](#). The router mode is used for routing new tracks, but also when dragging existing tracks using the Drag (hotkey **D**) command. These modes are:

- **Highlight Collisions:** in this mode, most of the router features are disabled and routing is fully manual. When routing, *collisions* (clearance violations) will be highlighted in green and the newly-routed tracks cannot be fixed in place if there is a collision unless the Allow DRC Violations option is turned on. In this mode, up to two track segments may be placed at a time (for example, one horizontal and one diagonal segment).
- **Shove:** in this mode, the track being routed will walk around obstacles that cannot be moved (for example, pads and locked tracks/vias) and *shove* obstacles that can be moved out of the way. The router prevents DRC violations in this mode: if there is no way to route to the cursor position that does not violate DRC, no new tracks will be created.
- **Walk Around:** in this mode, the router behaves the same as in Shove mode, except no obstacles will be moved out of the way.

Which mode to use is a matter of preference. For most users, we recommend using Shove mode for the most efficient routing experience or Walk Around mode if you do not want the router to modify tracks that are not being routed. Note that Shove and Walk Around modes always create horizontal, vertical, and 45-degree (H/V/45) track segments. If you need to route tracks with angles other than H/V/45, you must use Highlight Collisions mode and enable the Free Angle Mode option in the [Interactive Router Settings dialog](#).

There are four main routing functions: Route Single Track, Route Differential Pair, Tune length of a single track, and Tune skew of a differential pair. All of these are present in both the Route menu dropdown (individually) on the top toolbar and the drawing toolbar in two overloaded icons on the drawing toolbar on the right. The use of the overloaded icons is described above. One is for the two Route functions and one is for the two Tune functions. In addition, the Route menu allows the selection of Set Layer Pair and Interactive Router Settings.

To route tracks, click the Route Tracks  icon (from the drawing toolbar or from the top toolbar under **Route**) or use the hotkey **X**. Click on a starting location to select which net to route and begin routing. The net being routed will automatically be highlighted and the allowable clearance for the net will be indicated with a gray outline around the tracks being routed. The clearance outline can be disabled by changing the Clearance Outlines setting in the Display Options section of the Preferences dialog.

**NOTE**

The clearance outline shows the maximum clearance from the routed net to any other copper on the current layer. It is possible to use custom design rules to specify different clearances for a net to different objects. These clearances will be respected by the router, but only the largest clearance value will be shown visually.

When the router is active, new track segments will be drawn from the routing start point to the editor cursor. These tracks are *unfixed* temporary objects that show what tracks will be created when you use a left-click or the **Enter** key to *fix* the route. The unfixed track segments are shown in a brighter color than the fixed track segments. When you exit the router using the **Esc** key or by selecting another tool, only the fixed track segments will be saved. The Finish Route action (hotkey **End**) will fix all tracks and exit the router.

While you are routing, you can use the Undo Last Segment command (hotkey **Backspace**) to unfix the tracks you most recently fixed. You can use this command repeatedly to step back through the route that you have already fixed.

In previous versions of KiCad, using the left mouse button or **Enter** to fix the routed segments would fix all segments up to but *not including* the segment ending at the mouse cursor location. In KiCad 6 and later, this behavior is optional, and by default, all segments *including* the one ending at the mouse cursor location will be fixed. The old behavior can be restored by disabling the "Fix all segments on click" option in the Interactive Router Settings dialog.

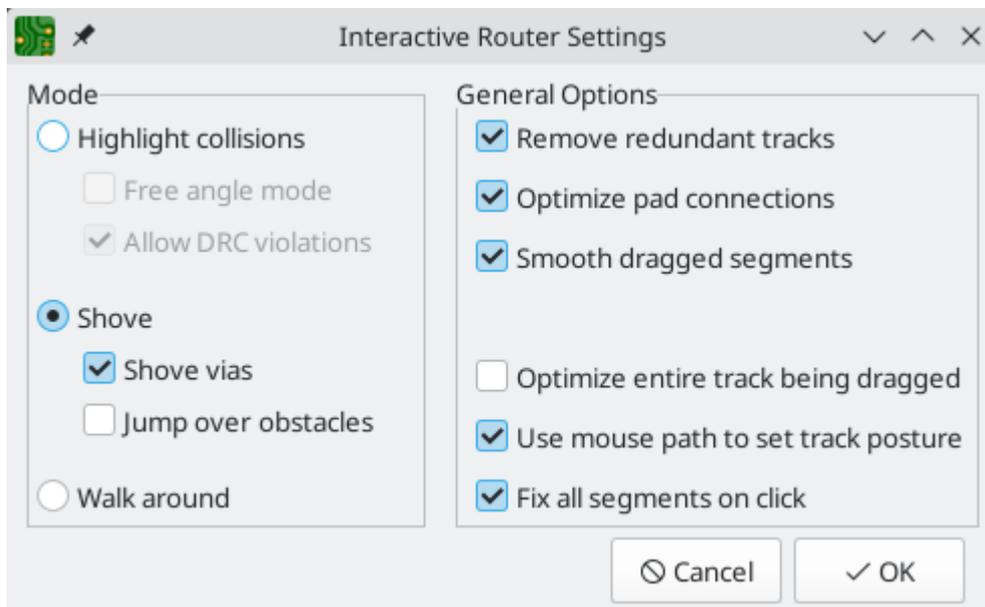
While routing, you can hold the **Ctrl** key to disable grid snapping, and hold the **Shift** key to disable snapping to objects such as pads and vias.

**NOTE**

Snapping to objects can also be disabled by changing the Magnetic Points preferences in the Editing Options section of the Preferences dialog. We recommend that you leave object snapping enabled in general, so that you do not accidentally end tracks slightly off-center on a pad or via.

## Interactive router settings

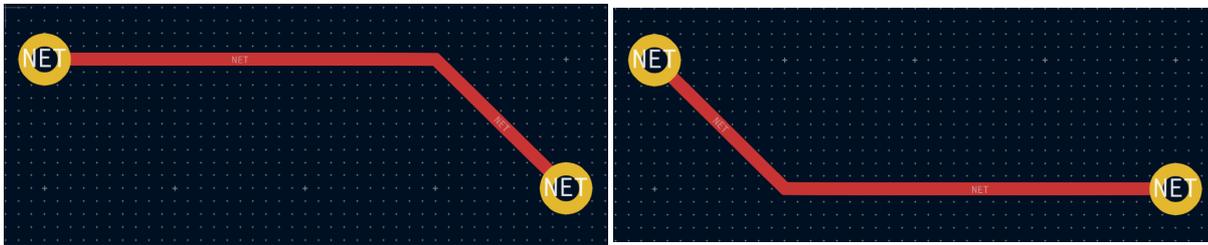
The interactive router settings can be accessed through the **Route** menu, or by right-clicking on the  button in the toolbar. These settings control the router behavior when routing tracks as well as when dragging existing tracks.



Setting	Description
Mode	Sets the operating mode of the router for creating new tracks and dragging existing tracks. <a href="#">See the routing overview</a> for more information.
Free angle mode	Allows routing tracks at any angle, instead of just at 45-degree increments. This option is only available if the router mode is set to Highlight collisions.
Allow DRC violations	Allow placing tracks and vias that violate DRC rules. This option is only available if the router mode is set to Highlight collisions.
Shove vias	Allow the router to shove vias along with tracks. When this is disabled, vias cannot be shoved. This option is only available if the router mode is set to Shove.
Jump over obstacles	Allow the router to attempt to move colliding tracks behind solid obstacles (such as pads). This option is only available if the router mode is set to Shove.
Remove redundant tracks	Automatically removes loops created in the currently-routed track, keeping only the most recently routed section of the loop.
Optimize pad connections	When this setting is enabled, the router attempts to avoid acute angles and other undesirable routing when exiting pads and vias.
Smooth dragged segments	When dragging tracks, attempts to combine track segments together to minimize direction changes.
Optimize entire track being dragged	When enabled, dragging a track segment will result in KiCad optimizing the rest of the track that is visible on the screen. The optimization process removes unnecessary corners, avoids acute angles, and generally tries to find the shortest path for the track. When disabled, no optimizations are performed to the track outside of the immediate section being dragged.
Use mouse path to set track posture	Attempts to pick the track posture based on the mouse path from the routing start location.
Fix all segments on click	When enabled, clicking while routing will fix the position of all the track segments that have been routed, including the segment that ends at the mouse cursor. A new segment will be started from the mouse cursor location. When disabled, the last segment (the one that ends at the mouse cursor) will not be fixed in place and can be adjusted by further mouse movement.

## Track posture

When routing in H/V/45 mode, the *posture* refers to how a set of two track segments connect two points that cannot be reached by a single H/V/45-degree segment. In such a case, the points will be connected by one horizontal or vertical segment and one diagonal (45-degree) segment. The posture refers to the order of these segments: whether the horizontal/vertical segment or the diagonal segment comes first.



KiCad's router attempts to pick the best posture automatically based on a number of factors. In general, the router will attempt to minimize the number of corners in a route, and will avoid "bad" corners such as acute angles whenever possible. When routing from or to a pad, KiCad will choose the posture that lines up the route with the longest edge of the pad.

In some cases, KiCad cannot guess the posture you intend correctly. To switch the posture of the track while routing, use the Switch Track Posture command (hotkey `/`).

In situations where there is no obvious "best" posture (for example, when starting a route from a via), KiCad will use the movement of your mouse cursor to select the posture. If you would like the route to begin with a straight (horizontal or vertical) segment, move the mouse away from the starting location in a mostly horizontal or vertical direction. If you would like the route to begin diagonally, move in a diagonal direction. Once the cursor is a sufficient distance away from the routing start location, the posture is set and will no longer change unless the cursor is brought back to the starting location. Detection of posture from the movement of the mouse cursor can be disabled in the Interactive Router Settings dialog as described below.

#### NOTE

If you use the Switch Track Posture command to override the posture chosen by KiCad, the automatic detection of posture from mouse movement will be disabled for the remainder of the current routing operation.

## Track corner mode

KiCad's router can place tracks using four different corner modes:

- 45 degree (default)
- 45 degree rounded
- 90 degree
- 90 degree rounded

Use the Track Corner Mode command (`Ctrl` + `/`) to cycle between these modes. These corner modes do not apply when the router is in [free angle mode](#).

In the 45 degree modes, tracks can be placed horizontally, vertically, or at 45 degree diagonals, and track segments are joined at 45 or 135 degree angles. This is the most common corner mode and it is selected by default.



*45 degree track corner mode*

In the 90 degree modes, diagonal tracks cannot be placed and track segments are joined at 90 degree angles.



*90 degree track corner mode*

When routing with rounded corners, each routing step will place either a straight segment, a single arc, or both a straight segment and an arc. The track posture determines whether the arc or the straight segment will be placed first.



*45 degree rounded track corner mode*



*90 degree rounded track corner mode*

Track corners can also be rounded after routing by using the Fillet Tracks command after selecting the tracks on either side of the corner to be filleted. If a contiguous track selection contains multiple corners, they will all be filleted.

**NOTE**

Dragging of tracks with arcs is not supported. Arcs are treated as immovable by the shove router.

## Track width

The width of the track being routed is determined in one of three ways: if the routing start point is the end of an existing track and the  button on the top toolbar is enabled, the width will be set to the width of the existing track. Otherwise, if the track width dropdown in the top toolbar is set to "use netclass width", the width will be taken from the netclass of the net being routed (or from any custom design rules that specify a different width for the net, such as inside a neckdown area). Finally, if the track width dropdown is set to one of the [pre-defined track sizes](#) configured in the Board Setup dialog, this width will be used.

**NOTE**

The track width can never be lower than the minimum track width configured in the Constraints section of the Board Setup dialog. If a pre-defined width is added that is lower than this minimum constraint, the minimum constraint value will be used instead.

KiCad's router supports a single track width for the active route. In other words, to change widths in the middle of a track, you must end the route and then restart a new route from the end of the previous route. To change the width of the active route, use the hotkeys **W** and **Shift + W** to step through the track widths configured in the Board Setup dialog.

## Placing vias

While routing tracks, switching layers will insert a through via at the end of the current (unfixed) track. Once you place the via, routing will continue on the new layer. There are several ways to select a new layer and insert a via:

- By using the hotkey to select a specific layer, such as **PgUp** to select **F.Cu** or **PgDn** to select **B.Cu**.
- By using the Next Layer or Previous Layer hotkeys (**+** and **-**).
- By using the Place Via hotkey (**V**), which will switch to the next layer in the active layer pair.
- By using the Select Layer and Place Through Via action (hotkey **<**), which will open a dialog to select the target layer.

After using any of the above methods to add a via and change layer, but before clicking to fix the via and commit the current track segment, you can cancel placing the via by pressing **V**. The via will be removed and routing will continue on the original layer.

You can place a via and end the current track, without changing layers, by pressing **V** and then double-clicking to place the via.

The size of the via will be taken from the active Via Size setting, accessible from the drop-down in the top toolbar or the Increase Via Size (**+**) and Decrease Via Size (**-**) hotkeys. Much like track width, when the via size is set to "use netclass sizes", the via sizes configured in the Net Classes section of the Board Setup will be used (unless overridden by a custom design rule).

You can also place microvias and blind/buried vias while routing. Use the hotkey **Ctrl + V** to place a microvia and **Alt + Shift + V** to place a blind/buried via. While regular vias always go through every board layer, microvias and blind/buried vias can start and end on any layer, not just the outer layers.

**NOTE**

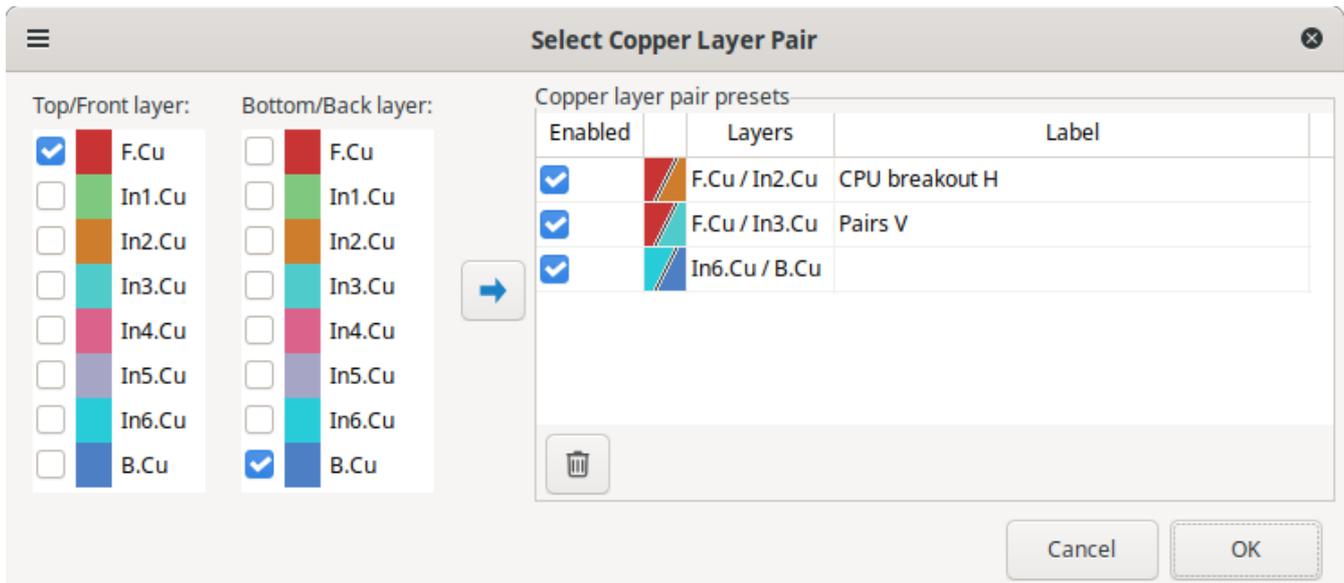
For the purposes of DRC, microvias are not considered drilled holes as they are laser drilled rather than mechanically drilled. See the [DRC documentation](#) for more information.

Vias placed by the router are considered to be part of a routed track. This means that the via net can be updated automatically (just like track nets can), for example when updating the PCB from the schematic changes the net name of the track. In some cases this may not be desired, such as when creating stitching vias. The automatic update of via nets can be disabled for specific vias by turning off the "automatically update via nets" checkbox in the via properties dialog. Vias placed with the Add Free-standing Vias tool are created with this setting disabled.

## Layer Pairs

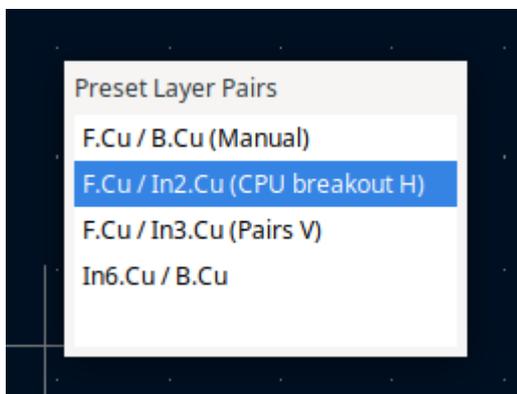
The active layer is swapped with the other one in the current layer pair using the Place Via hotkey (**V**).

You can define the active pair along with a list of "preset" layer pairs in the Set Layer Pair dialog, accessed from the two-color swatch on the toolbar. These pairs are stored in the project file.



Each can be enabled or disabled, and given an optional user-friendly name.

The enabled presets can be cycled using the Cycle Layer Pair Presets hotkey (**Shift** + **V**). If the last-used or current layer pair is not a preset, it is included in the list with the name "Manual".



## Placing free vias

In addition to [placing vias while routing](#), you can also place standalone vias. These vias connect to items that they touch when they are placed. Free vias may be useful for via stitching, via shielding, thermal design, or other reasons.

To place a free via, click the  button or press **Ctrl** + **Shift** + **X**, then click in the desired location in the editing canvas. If you place a via directly over a track, it will connect to that track as if it was placed while routing: it will take the track's net, it will create a joint in the track, and dragging the via will also drag the attached tracks.

The net assigned to a free via depends on where the via was placed. If the via was placed over a track or pad, it will have the same net as the track, and its **Automatically update via nets** setting will be enabled so that

its net changes with the track's net. Otherwise, the via will take the net of any zone under the via, if one exists, and its net will not update automatically. If there are multiple zones under the via, you will be prompted to choose which net to use. If there is no zone, the via will not have a net assigned.

## Modifying tracks

After tracks have been routed, they can be modified by moving or dragging, or deleted and re-routed. When a single track segment is selected, the hotkey **U** can be used to expand the selection to all connected track segments. The first press of **U** will select track segments between the nearest junctions with pads or vias. The second press of **U** will expand the selection again to include all track segments connected to the selected track on all layers. Selecting tracks with this technique can be used to quickly delete an entire routed net.

There are two different drag commands that can be used to reposition a track segment. The Drag (45-degree mode) command, hotkey **D**, is used to drag tracks using the router. If the router mode is set to Shove, dragging with this command will shove nearby tracks. If the router mode is set to Walk Around, dragging with this command will walk around or stop at obstacles. Multiple tracks can be dragged at once using this command. The Drag Free Angle command, hotkey **G**, is used to split a track segment into two and drag the new corner to any location. Drag Free Angle behaves like the Highlight Collisions router mode: obstacles will not be avoided or shoved, only highlighted.

### NOTE

Dragging of tracks containing arcs is not yet possible. Attempting to drag these tracks will result in the arcs being removed in some cases. It is possible to resize a particular arc by selecting it and using the drag command (**D**). When resizing an arc using this command, no DRC checking is performed.

The Move command (hotkey **M**) can also be used on track segments. This command will pick up the selected track segments, ignoring any attached track segments or vias that are not selected. No DRC checking is done when moving tracks using the Move command.

It is also possible to move a footprint while keeping tracks attached to the footprint as it moves. To do so, use the drag command (**D**) with one or more footprints selected. Any tracks that end at one of the footprint's pads will be dragged along with the footprints. This feature has some limitations: it only operates in Highlight Collisions mode, so the tracks attached to footprints will not walk around obstacles or shove nearby tracks out of the way. Any DRC violations caused by the drag operation will be highlighted and will prevent the footprint drag from being committed when you click. To ignore the violations and commit the drag anyway, use **Ctrl**+click. Additionally, only tracks that end at the origin of the footprint's pads will be dragged. Tracks that simply pass through the pad or that end on the pad at a location other than the origin will not be dragged.

To break a single track segment into two, use the Break tool (right click a track → **Break Track**). The track will be broken into two connected track segments at the cursor location. Each track segment can then be selected, moved, and edited individually. To recombine the segments into a single segment, drag the track, or use the **merge co-linear tracks** option in the [Cleanup Tracks and Vias dialog](#).

## Editing track and via properties

You can modify the width of tracks and the size of vias, without re-routing them, in the properties dialog for the track or via. This modifies all selected tracks and vias. The properties dialog shows the relevant properties for the items in the selection: if both tracks and vias are selected, then properties for both types

of objects will be displayed, but if only one type of object is selected then properties for the other type of object will not be shown.

**NOTE**

The properties of selected tracks and vias can also be modified using the [Properties Manager](#).

Track & Via Properties

Common

Net:   Automatically update via nets

Locked

Tracks

Start X:  Y:  mm    Layer:

End X:  Y:  mm    Technical Layers

Pre-defined sizes:  mm     Solder mask    Expansion:  mm

Track width:  mm

Vias

Position X:  Y:  mm    Via type:

Pre-defined sizes:  mm    Start layer:

Padstack mode:  Edit layer:     End layer:

Via diameter:  mm    Annular rings:

Via hole:  mm    Protection features:

Backdrill:     Top backdrill must-cut:     Bottom backdrill must-cut:

Hole Post-machining

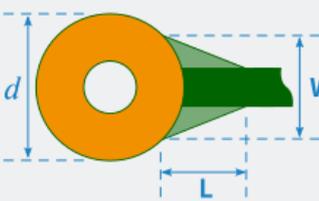
Top:     Bottom:

Size:  mm    Size:  mm

Angle:  mm    Angle:  mm

Add teardrops on via's track connections

Allow teardrops to span two track segments



Maximum track width:  %  
*(as a percentage of pad/via/track size)*

Best length (L):  %(*d*)

Maximum length (L):  mm

Best width (W):  %(*d*)

Maximum width (W):  mm

Curved edges

## Track and via nets

In the Common section, you can change the assigned net of the selected objects using the **Net** dropdown. If the **Automatically update via nets** option is checked, the selected vias cannot have their assigned net manually changed, but instead will be assigned the net of any zone or pad that they touch. You can also [lock](#) the selected objects.

## Track size, position, and layers

In the Tracks section, you can set the start and end position of the tracks and the layer they are on. You can also change the track width, either from a list of [pre-defined sizes](#) or to an arbitrary value.

You can remove the solder mask from on top of tracks on outer layers by enabling the **Solder mask** checkbox. When enabled, solder mask openings will be drawn for each of the selected tracks with the same shape as the source track. The **Expansion** textbox controls the size of the mask opening relative to the original track: the expansion value will be added to each side of the original track to form the mask shape. For example, a 1mm wide track with a 1mm expansion would result in a 3mm wide mask cutout, because the 1mm expansion is added to both sides of the track.

## Via size, position, and layers

In the Vias section you can change the properties of selected vias. You can change the position of a via, the via's type (through, micro, or blind/buried), and which layers it spans. Through vias always start and end on the front and back copper layers, but micro vias and blind/buried vias can start and end on any layers.

You can modify the via annulus and hole diameters, either from a list of [pre-defined sizes](#) or to arbitrary values. A via's diameter and hole size can be defined on a per-layer basis. This is also known as defining the via's *padstack*. The **Padstack mode** controls whether the via shape is the same on all layers or whether individual layers are individually controlled.

- In the **Normal** padstack mode, the via's diameter and hole size are the same on all layers.
- In the **Front/Inner/Back** padstack mode, the via's diameter and hole size can be controlled separately for the front layer, the back layer, and the inner layers (the inner layers will all have the same settings). The **Edit layer** dropdown controls which layer (or group of layers) is currently being displayed and edited.
- In the **Custom** padstack mode, the via's diameter and hole size can be controlled completely independently on each layer. The **Edit layer** dropdown controls which layer is currently being displayed and edited.

The **Annular rings** setting controls which layers will have annular rings for the via.

- When set to **All copper layers**, the via will have annular rings on every layer.
- When set to **Start, end, and connected layers**, the via will have annular rings on its start and end layers as well as any layer with a track or zone connection to the via. Any layer without track or zone connections, other than the start and end layers, will not have an annular ring.
- When set to **Connected layers only**, the via will have annular rings only on layers with a track or zone connection to the via. Any layer without track or zone connections will not have an annular ring.
- When set to **Start and end layers only**, the via will have annular rings only on its start and end layers. Zones will not connect to any layers other than the start and end layers. Vias configured in this way are

also known as *skip vias*.

Annular rings can be removed or added in bulk using the [Edit Track and Via Properties dialog](#) or by running the [Unused Pads tool](#).

The **Front tenting** and **Back Tenting** options control whether the via has front and back solder mask covering it.

- When set to **From design rules**, the tenting settings are taken from the Solder Mask/Paste panel in [Board Setup](#).
- When set to **Tented**, the via will be covered in solder mask, regardless of the settings in Board Setup.
- When set to **Not tented**, the via will not be covered with solder mask, regardless of the settings in Board Setup.

If the  button is pressed, the front and rear tenting settings will be linked. If it is unpressed, they can be modified independently.

### **Via backdrilling and post-machining (counterbores and countersinks)**

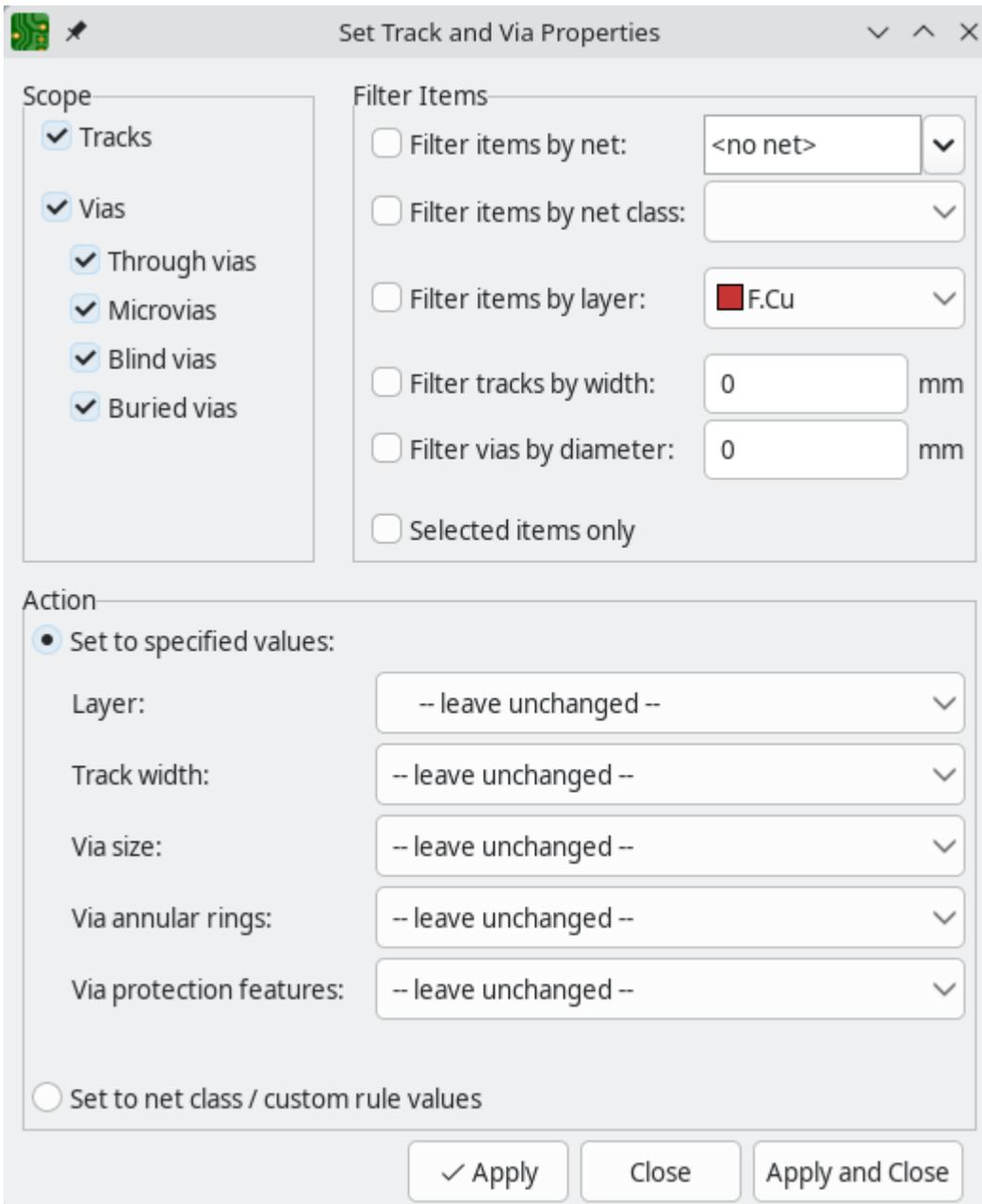
You can configure backdrills and post-machining (counterbores and countersinks) for vias in this dialog. [Backdrills](#) and [Post-machining](#) are explained in their own sections.

### **Via teardrops**

You can configure teardrops for vias in this dialog. Teardrop properties are explained in the [Teardrops](#) section.

### **Bulk editing tracks and vias**

To modify tracks and vias in bulk you can use the Edit Track and Via Properties dialog (**Edit** → **Edit Track & Via Properties...**).



**Scope** settings restrict the tool to editing only tracks, vias, or both. Vias can be additionally filtered by via type. If no scopes are selected, nothing will be edited.

**Filter Items** restricts the tool to editing particular objects in the selected scope. Objects will only be modified if they match all enabled and relevant filters (some filters do not apply to certain types of objects. For example, via diameter filters do not apply to tracks). If no filters are enabled, all objects in the selected scope will be modified. For filters with a text box, wildcards are supported: \* matches any characters, and ? matches any single character.

- **Filter items by net** filters to items assigned the specified net.
- **Filter items by net class** filters to items assigned to the specified net class.
- **Filter items by layer** filters to items on the specified board layer.
- **Filter tracks by width** filters to tracks with the specified track width.
- **Filter vias by diameter** filters to vias with the specified diameter.

**Selected items only** filters to the current selection.

The **Action** section determines what editing actions are performed on the filtered objects.

- When **Set to net class / custom rule values** is selected, the filtered objects are adjusted to match the values specified by the [net class values](#) and [custom design rules](#).
- When **Set to specified values** is selected, you can choose which properties to modify and how to set each property. For each property, you can choose `-- leave unchanged --` to preserve objects' existing values for that property, or select a new value from the dropdown menu.

The editable properties for tracks are:

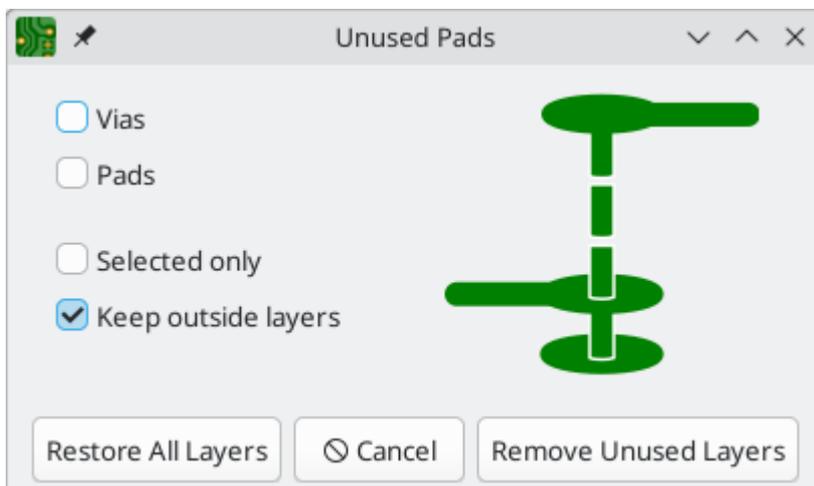
- Layer.
- Track width. The options are defined in [Board Setup's Pre-defined Sizes](#).

The editable properties for vias are:

- Via size. The options are defined in [Board Setup's Pre-defined Sizes](#).
- [Via annular rings](#).
- [Via protection features](#).

## Removing unused pads

You can quickly remove unused annular rings from pads and vias using the Unused Pads tool (**Tools** → **Remove Unused Pads...**). This will leave annular rings in place on layers where they are used and remove them on layers where they are not used. An annular ring is considered unused if there are no track or zone connections to the pad/via on that layer.



The **Remove Unused Layers** button removes all unused annular rings from pads and vias that meet the selected filter settings. The **Restore All Layers** button restores all annular rings to the pads and vias that meet the selected filter settings.

The checkboxes filter which objects will be modified (annular rings removed or restored) and which layers will be removed for those objects.

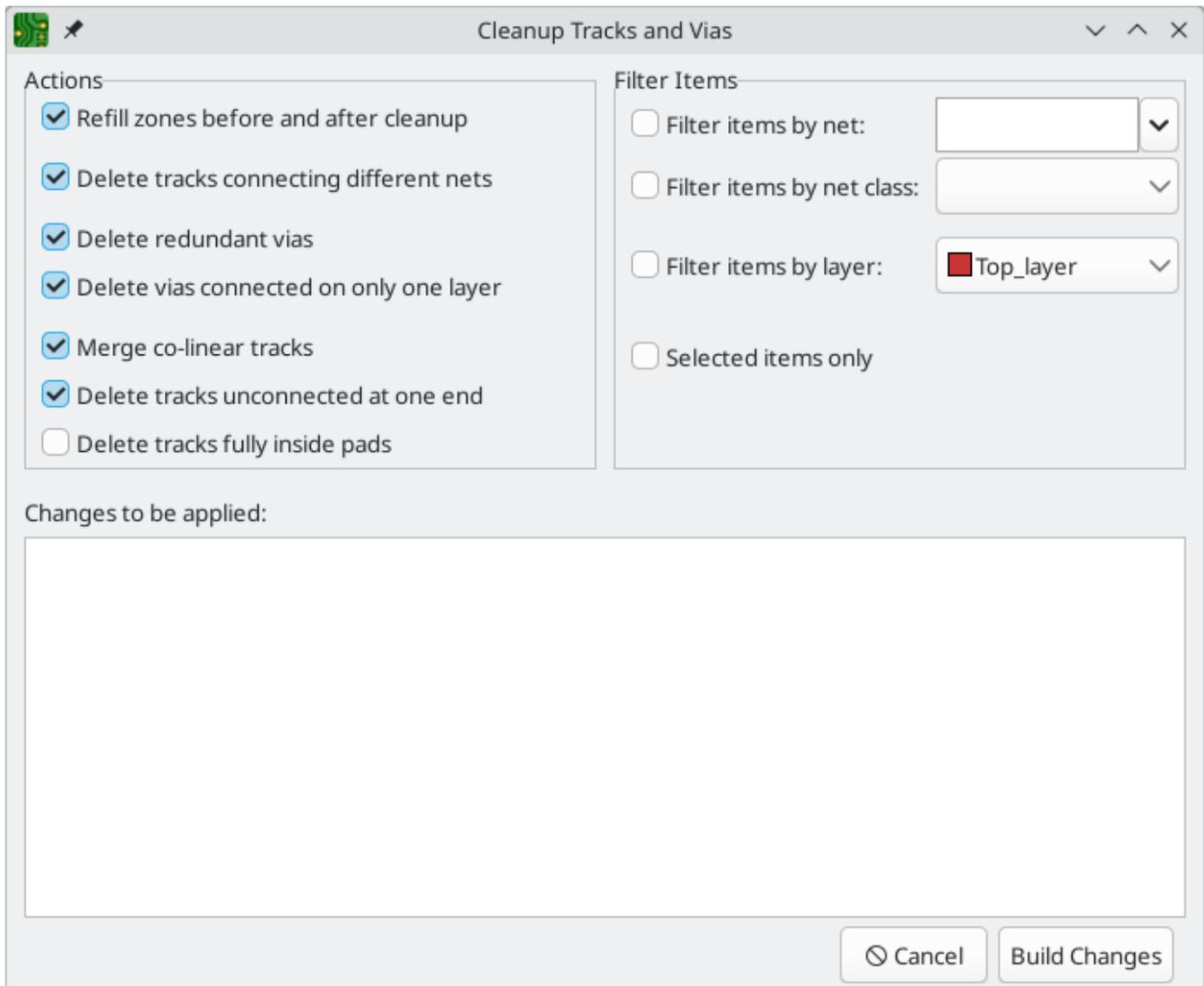
- If the **Vias** checkbox is enabled, annular rings for vias will be modified.
- If the **Pads** checkbox is enabled, annular rings for pads will be modified.

If the **Selected only** checkbox is enabled, only selected vias and pads will have their annular rings modified. If it is disabled, annular rings for all vias and pads will be modified. This setting applies in combination with the **Vias** and **Pads** checkboxes; for example, a selected via will not be modified if the **Via** checkbox is disabled.

- If the **Keep outside layers** checkbox is enabled, the pad or via's start and end layers will remain, even if they are unused.

## Cleaning up tracks and vias

There is a dedicated tool for performing common cleanup operations on tracks and vias, which is run via **Tools → Cleanup Tracks & Vias...**



The following cleanup actions are available and will be performed when selected:

- **Refill zones before and after cleanup:** refills all zones both before and after the cleanup operation. If unchecked, zone fills will not be changed.
- **Delete tracks connecting different nets:** removes any track segments that short multiple nets.
- **Delete redundant vias:** remove vias that are redundant because they are located on top of another via or on top of a through hole pad.
-

**Delete vias connected on only one layer:** removes vias that are only connected to copper on a single layer and are therefore unnecessary.

- **Merge co-linear tracks:** merges any track segments that are connected and co-linear into a single equivalent track segment.
- **Delete tracks unconnected at one end:** removes track segments that have at least one dangling end.
- **Delete tracks fully inside pads:** removes tracks that have both start and end points within a pad and are therefore unnecessary.

You can also filter the objects that will be cleaned up by net, netclass, layer, or selection.

- **Filter items by net:** limits the cleanup to tracks and vias assigned to the specified net.
- **Filter items by netclass:** limits the cleanup to tracks and vias in the specified netclass.
- **Filter items by layer:** limits the cleanup to tracks and vias on the specified layer.
- **Selected items only:** limits the cleanup to just the selected tracks and vias.

Any changes that will be applied to the board are displayed at the bottom of the dialog after clicking the **Build Changes** button. After building the changes, the button changes to say **Update PCB**. The changes are not applied until you press the **Update PCB** button.

## Routing Convenience Functions

KiCad offers several functions to make certain routing operations more convenient.

If you need to route a number of tracks from a set of pads, you can use the Route Selected tool to quickly route from each pad in sequence. Select the pads you want to use as starting points, then right click and choose **Route Selected** (**Shift** + **X**) to route from each pad in sequence. The router will begin a track from the first selected pad, which you can route as you would any other track. You can also select footprints instead of pads; all unrouted pads in the selected footprints will be used as starting points. When you complete the first track, the router will automatically begin a new track from the next pad in the selection, in the same order that you selected the pads. Pads that already have tracks attached are skipped. You can skip routing the current track and move on to the next pad by right clicking and choosing **Cancel Current Item**. Pressing **Esc** (or right clicking and pressing **Cancel**) skips the rest of the operation, leaving any already-completed tracks as they are.

If you want to route a number of tracks *to* a set of pads, instead of *from* the pads, you can use the Route Selected From Other End tool. Select the pads you want to use as ending points, then right click and choose **Route Selected From Other End** (**Shift** + **E**). This tool works the same way as the Route Selected tool, except it uses each selected pad as an end point rather than a starting point. The starting point for each track is the other end of the ratsnest line for each selected pad.

Routing from the other end is also possible while routing individual tracks: press **Ctrl** + **E** while routing a track to commit the current segment and begin routing from the other end of the in-progress track's ratsnest line.

Finally, you can quickly unroute tracks connected to an object (footprint, pad, or track) by selecting the object, right-clicking, and choosing **Unroute Selected**. Any tracks connected to the selected object will be removed, starting at the selected object and continuing until another pad is encountered.

## Automatically completing tracks

KiCad's router can automatically route individual tracks, based on the connections defined in the schematic. This can be thought of as a limited form of auto-routing that considers a single track at a time. The router will only use the current layer; it will not use vias or change layers.

While routing, press the **F** key to have the router attempt to automatically finish the current track. The track will be automatically routed from the end of the last fixed track segment to the closest ratsnest anchor. If the router can't automatically finish the track, it will allow you to complete the track manually. This action can also be performed by clicking **Attempt Finish** in the context menu while routing.

When the router is not the active tool, you can automatically route multiple tracks by selecting footprints, pads, and tracks to route from, right clicking, and choosing **Attempt Finish Selected (Autoroute)** (**Shift** + **F**). You do not need to select both ends of a desired connection; the router will route from the selected item to its nearest ratsnest anchor. If multiple items were selected, each item will be routed in sequence, in the order that they were selected. If a connection cannot be automatically completed, the tool will pause with the router active so that you can complete the track manually. With the automatic completion paused for a manual connection, you can skip the current track and move on to routing the next track by right clicking and choosing **Cancel Current Item**. After manually completing the track or skipping the connection, the tool will continue attempting to route the remaining connections. Pressing **Esc** (or right clicking and pressing **Cancel**) skips the rest of the operation, leaving any already-completed tracks as they are.

## Routing differential pairs

Differential pairs in KiCad are defined as nets with a common *base name* and a positive and negative suffix. KiCad supports using **+** and **-**, or **P** and **N** as the suffix. For example, the nets **USB+** and **USB-** form a differential pair, as do the nets **USB\_P** and **USB\_N**. In the first example, the base name is **USB**, and **USB\_** in the second. The suffix styles cannot be mixed: the nets **USB+** and **USB\_N** do not form a differential pair. Make sure you name your differential pair nets accordingly in the schematic in order to allow use of the differential pair router in the PCB editor.

To route a differential pair, click the Route Differential Pairs  icon (from the drawing toolbar or from the top toolbar under **Route**) or use the hotkey **6**. Click on a pad, via, or the end of an existing differential pair track to start routing. You can start routing from either the positive or negative net of a differential pair.

The differential pair router will attempt to route the pair of tracks with a gap taken from the design rules (differential pair gap can be configured in the Net Classes section of the Board Setup dialog, or by using custom design rules). If the starting or ending location of the route is a different distance apart from the configured gap, the router will create a short "fan out" section to minimize the length of track where the differential pair is not coupled.

When switching layers or using the Place Via (**V**) action, the differential pair router will create two vias next to each other. These vias will be placed as close as possible to each other while respecting the design rules for copper and hole-to-hole clearance.

## Length tuning

The length tuning tools can be used to add serpentine tuning shapes to tracks after routing. Length tuning shapes are persistent objects that can be modified after they are created. To tune the length of a track, first pick the appropriate tool.

- The single-track length tuning tool (icon  or hotkey 7) will add serpentine shapes to bring the length of a single track up to the target value.
- The differential pair length tuning tool (icon  or hotkey 8) will do the same for a differential pair.
- The differential pair skew tuning tool (icon  or hotkey 9) will add length to the shorter member of a differential pair in order to eliminate skew (phase difference) between the positive and negative sides of the pair.

As with the Routing icons, the Tuning icons are found in both the **Route** menu dropdown from the top toolbar and the drawing toolbar on the right.

When a tuning tool is active, you can hover over tracks in the board to show a status window that displays their current length or skew as well as the target values. Click on the desired track to start tuning it. As you move the mouse cursor along the track, meander shapes will be added interactively. If a target length has been set, meanders will stop being added when the target length is reached. You can set a target length with custom DRC rules or in the tuning shape properties; both methods are explained below. The popup window next to the cursor shows a live measure of the length or skew compared to the design targets. You can adjust the spacing (1 to increase and 2 to decrease) and amplitude (3 to increase and 4 to decrease) while you tune. When you are done, click again to commit the tuned shape. The tuned track doesn't need to be perfect because you can adjust the shape after committing it. You can also place multiple tuning shapes on the same track.

**NOTE**

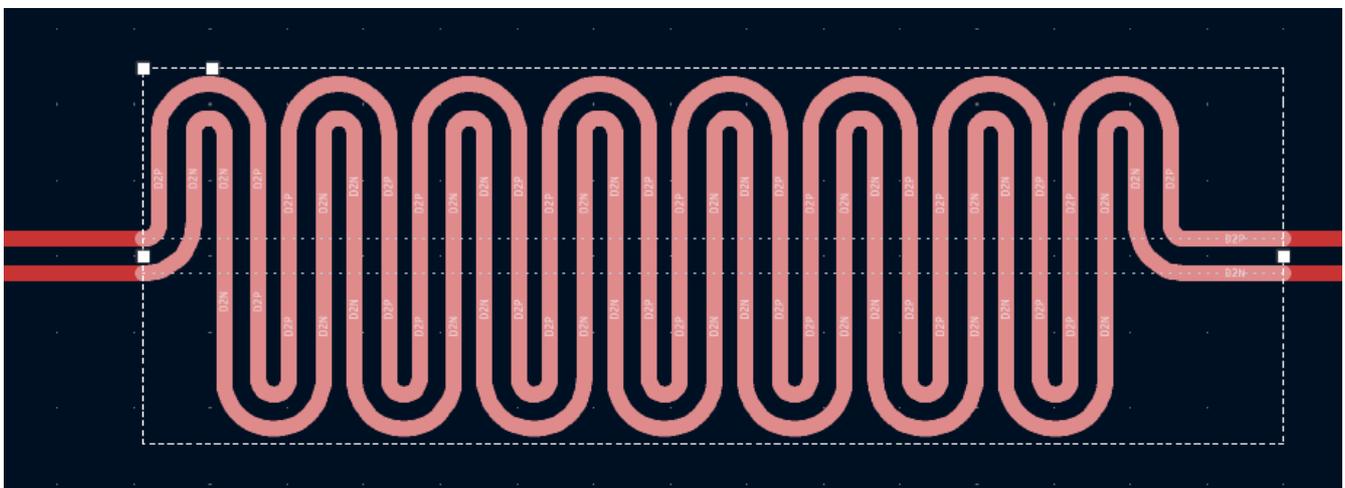
The length tuning tools only support tuning the length of point-to-point nets between two pads. Tuning the length of nets with different topologies is not supported.

**NOTE**

Differential pair length tuning can only be applied to the coupled portions of differential pairs. To apply length tuning to the uncoupled portions of differential pairs, you must use single-track length tuner.

## Editing tuning patterns

After a tuning pattern has been added, it can be selected, modified, and moved. While it is selected, the target length and routed length are shown in the message panel at the bottom left of the window.



When a pattern is selected, editing handles appear, which let you adjust the pattern geometry.

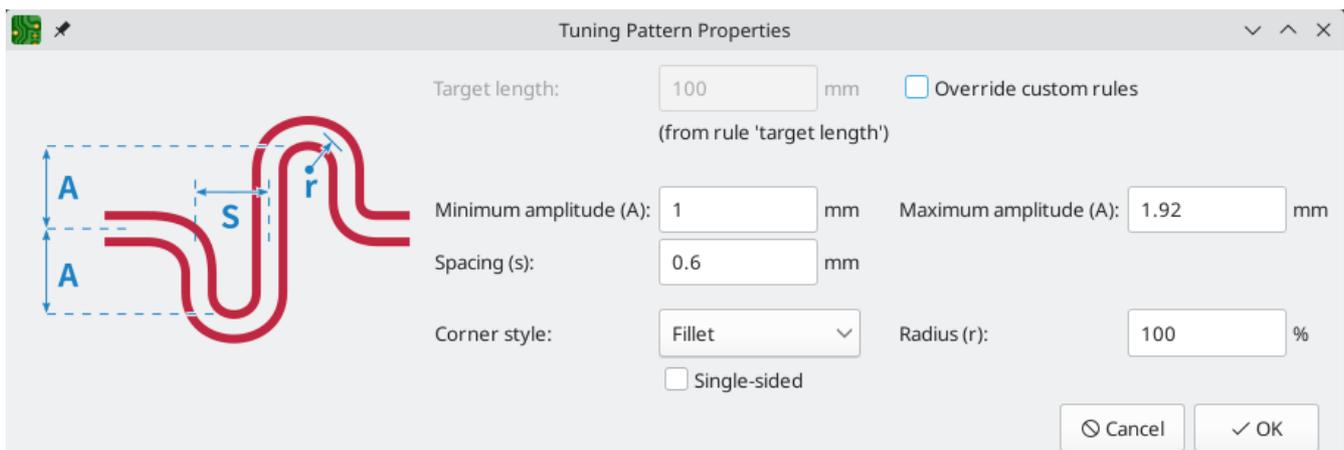
Dragging the handles at the ends of the pattern will expand or contract the pattern along the track.

- Dragging the corner handle towards or away from the track will respectively decrease or increase the maximum meander amplitude.
- The final handle controls the meander spacing; dragging it towards the corner handle will increase the spacing, while dragging it away from the corner handle will increase the spacing.

The selection box and editing handles represent the maximum allowable extents of the tuning pattern. Making the box smaller will reduce the size of the tuning pattern, even if this results in the tuned track being shorter than the target length. When the box is enlarged, the tuning pattern will expand to fill the box until the target length is reached.

You can move a tuning pattern along its track by selecting it and dragging with the mouse, or using the Move tool (**M**). Deleting a tuning pattern (**Del**) removes the tuning pattern and restores the original untuned tracks. You can also [ungroup](#) the tuning pattern, which will decompose it into its component tracks. The basic tracks have the same shape as the tuning pattern but can be edited individually. Once ungrouped into tracks, a tuning pattern cannot be regrouped.

Another way to edit a tuning pattern is through its properties dialog. The properties dialog exposes several additional parameters that can't be modified using the on-canvas interactive editor. These properties can also be edited in the [Properties Manager](#).



As with the interactive editor, you can set a maximum amplitude for the tuning pattern and a spacing between meanders, but here you can set a minimum amplitude and configure the corner style. Corners can be **filleted** (rounded) or **chamfered**. In each case you can set the **radius** as a percentage of the maximum possible radius for the spacing and amplitude. You can also configure the tuning pattern to be **single-sided**, which restricts it to one side of the baseline, as opposed to the default style which positions meanders on both sides of the baseline.

You can set default values for these properties in the **Design Rules** → **Length-tuning Patterns** page of the Board Setup dialog. Each type of tuning pattern (single track length, differential pair length, and differential pair skew) can have its own defaults.

Finally, the tuning pattern properties dialog is one of two ways to set the target length or skew for a tuning pattern. Setting length targets is explained below.

## Setting target length and skew

There are two ways to set a target length or skew for a net:

- In the properties dialog for a tuning pattern that has already been added to a track.
- Using a custom DRC rule with the `length` and/or `skew` constraints.

The first method is to specify a target in the **target length** or **target skew** field of the tuning pattern's properties dialog. This target will only apply to the selected tuning pattern. Therefore, length targets set in this way must be set separately for each tuning pattern in the design. The properties dialog for a tuning pattern is only accessible after the pattern is initially created, so changing a target length or skew in this way may require the pattern to be adjusted to meet the new target value, if the pattern's geometric constraints do not allow sufficient space to meet the new target.

You can also set a target length and/or skew using [custom design rules](#). If custom rules are used, they will override any targets set in tuning pattern properties, unless the **override custom rules** checkbox is enabled in the tuning pattern properties.

Using a custom rule allows you to set a net's target length and/or skew up front, before a pattern is created. With custom rules you can set different length and skew targets based on specific criteria, such as netclass or net name. You will also result in a DRC violation if the net's length or skew is out of bounds.

When target length or skew is adjusted in a custom DRC rule after a pattern is created, the pattern geometry will not be automatically updated to achieve the new target. You can use **Edit** → **Update All Tuning Patterns** to recalculate all tuning patterns to meet the new targets.

The following example custom rule sets a target length and skew for nets in the `high_speed` netclass. The target length is 100mm, and a DRC error will be raised if it is below 95mm or above 105mm. The target skew is at most 0.1mm.

```
(rule "target length and skew"
  (condition "A.hasNetclass('high_speed')")
  (constraint length (min 95mm) (opt 100mm) (max 105mm))
  (constraint skew (max 0.1mm)))
```

See the custom rule documentation for more details of how to create rules that only apply to certain nets.

## Length tuning pitfalls and tips

The length tuner only tunes nets with a point-to-point topology; branching nets are not supported. When the length tuner encounters a branch, it stops at the branch and only considers the length of the net up to that branch.

Sometimes you may end up with leftover stub tracks somewhere in your design. These can turn what appears to be a point-to-point net into a branched topology, which will prevent length tuning from working as expected. It may be easier to find such stub tracks when you switch footprints, vias, and tracks to outline mode (, , and  buttons, respectively). You can also use the [track cleanup tool](#) (**Tools** → **Cleanup Tracks and Vias...**) to remove many of these stubs automatically.

By default, the length tuner includes vias in its length calculations. Only the layer-to-layer length of the via is used, which may be shorter than the full top-to-bottom via height if the tuned path is not exclusively on the board top and bottom. The accuracy of this calculation depends on the board stackup being accurately configured. Via length can be ignored in length tuner calculations by deselecting **include stackup height in track length calculations** in the **Constraints** page of the [Board Setup dialog](#).

The length tuner is optimized for adjusting the effective electrical distance between two points, and therefore it calculates net length in a slightly different way than other tools, such as the Net Inspector. In addition to discounting net branches and unused portions of vias, the length tuner also optimizes paths through pads to use the shortest possible path in its calculations. In comparison, the Net Inspector reports a simple summation of copper segment lengths. Both calculations are accurate, but they are optimized for different purposes. These differences are discussed in more detail in the [Net Inspector documentation](#).

## Time-domain tuning (propagation delay)

Traditional length tuning matches the physical trace length of nets so that all routed paths are the same distance. This works well when all traces in a group are routed on the same layer with the same geometry, because every trace has the same propagation velocity and equal lengths produce equal delays.

In many real-world designs, however, signals travel through multiple board layers with different stackup geometries. A trace on an outer layer (microstrip) has a different propagation velocity from a trace on an inner layer (stripline), because the effective dielectric constant surrounding the conductor differs between the two geometries. When a net group contains traces routed on a mix of layers, matching physical length alone does not guarantee that all signals arrive at the same time.

Time-domain tuning solves this problem by matching **propagation delay** instead of physical length. The length tuner calculates the signal delay through every segment of the path—tracks, vias, and pad-to-die connections—using per-layer propagation parameters that you define in a [tuning profile](#). Meander shapes are then added or adjusted until the total propagation delay meets the target, rather than matching a target length.

## When to use each mode

**Length-domain tuning** is appropriate when:

- All traces in the matched group are on the same layer.
- The board stackup is simple (e.g., two-layer board) and propagation velocity differences between layers are negligible.
- The design requirements specify matched lengths rather than matched delays.

**Time-domain tuning** is appropriate when:

- Matched nets are routed across multiple layers (e.g., signals that transition between outer and inner layers with vias).
- The board stackup causes meaningful differences in propagation velocity between layers (e.g., a high-layer-count design with both microstrip and stripline routing).
- The design requirements specify matched delays or flight times (common in DDR memory, PCIe, USB, and other high-speed interfaces).

**TIP**

Even if your design appears to have all traces on one layer, enabling time-domain tuning accounts for via transitions and pad-to-die delays that pure length matching ignores. For the most accurate timing analysis on high-speed buses, prefer time-domain tuning.

## How time-domain tuning works

When time-domain tuning is active for a net, the length tuner replaces its length-based calculations with delay-based calculations. The overall propagation delay for a routed path is the sum of three components:

### Track delay

For each trace segment, the delay is calculated by multiplying the segment's physical length by the per-layer **unit delay** (propagation delay per unit distance) defined in the tuning profile. Because each layer can have a different unit delay, traces on different layers contribute different amounts of delay per unit of physical length.

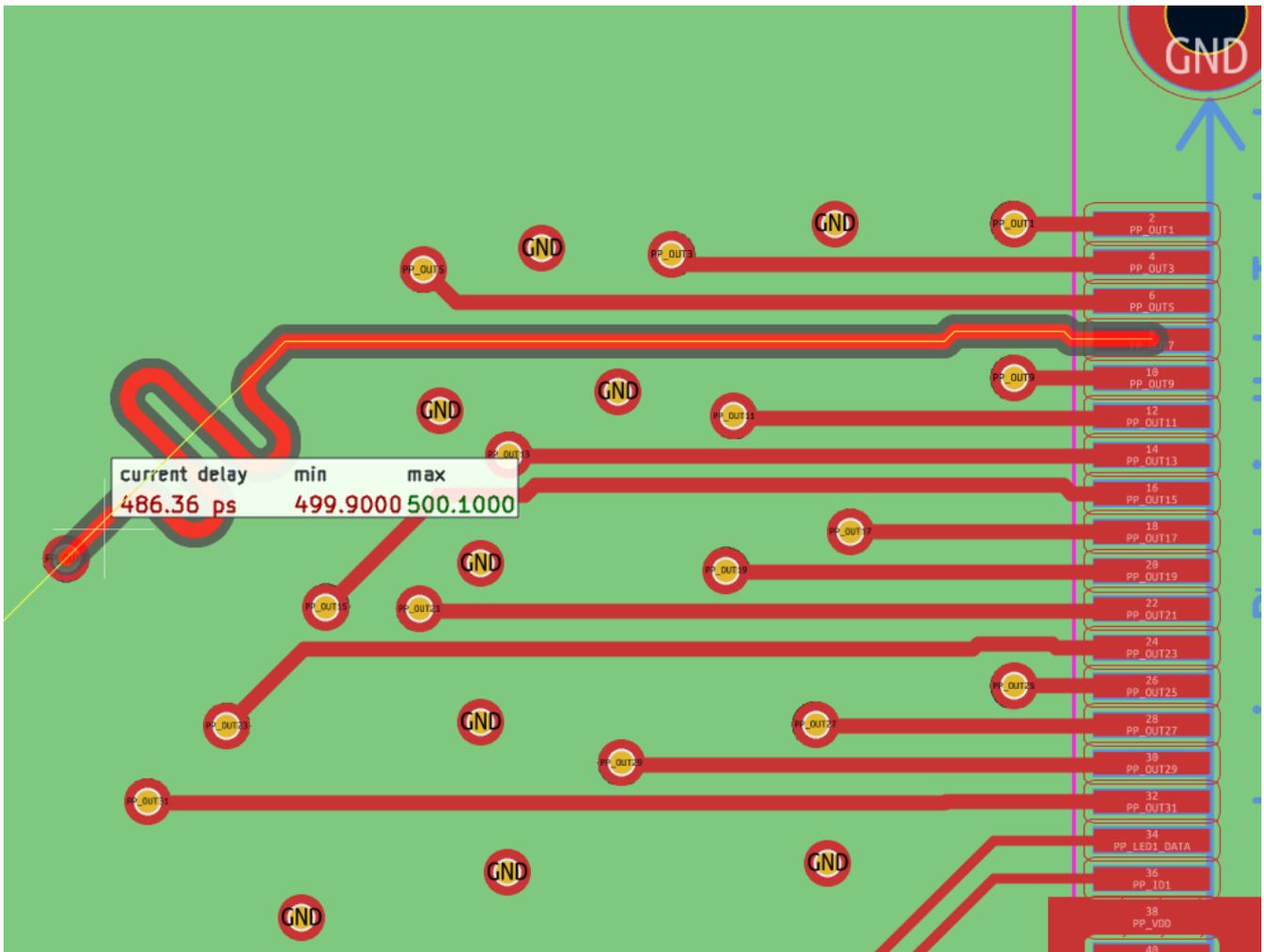
### Via delay

Each via contributes a propagation delay based on the distance the signal travels through the via (determined from the board stackup) and the via's unit delay. The tuning profile defines a **global via unit delay** and allows per-layer-pair overrides for situations where different via transitions have different electrical characteristics.

### Pad-to-die delay

If a pad has a pad-to-die length or pad-to-die delay configured in its padstack properties, this value is included in the total delay calculation.

The length tuner displays the total propagation delay in the status window when you hover over a net with a time-domain tuning profile active. The target is shown in time units (picoseconds) instead of length units.



## Board stackup and delay calculations

Accurate time-domain tuning depends on a correctly configured board stackup. The stackup defines the physical distances between copper layers, which determines via heights, and the dielectric properties of each layer, which affect propagation velocity calculations.

Configure your stackup in **Board Setup** → **Physical Stackup**. Ensure that:

- The number of copper layers is correct.
- Dielectric materials and thicknesses are set to match your fabrication stackup.
- Prepreg and core thicknesses are accurate, as they directly affect the calculated impedance and propagation delay for each signal layer.

The tuning profile's built-in calculator uses the stackup to automatically determine unit delay values for each signal layer. If you recalculate delay values after changing the stackup, the new dielectric properties will be taken into account.

### NOTE

The stackup must be configured before creating tuning profiles. If you change the stackup after setting up tuning profiles, you should recalculate the delay values in your tuning profiles to reflect the updated geometry.

## Propagation delay calculation details

KiCad's propagation delay calculation engine processes the routed path of a net and applies several optimizations to determine the true electrical path:

- The path through a via is optimized so that only the portion between the layers actually used by the routed signal is counted. If a through via connects traces on `F.Cu` and `In1.Cu`, only the stackup distance between those two layers contributes to the via delay, not the full via height from `F.Cu` to `B.Cu`.
- Contiguous track segments on the same layer are merged into single line chains before delay calculation, ensuring that segment boundaries do not introduce calculation artifacts.
- Where a track enters a pad, the electrical path is clipped to the shortest distance through the pad shape rather than using the full track length to the pad center. This gives a more accurate representation of the actual signal path.

These optimizations ensure that the delay calculation closely matches the true electrical path of the signal.

## Per-layer delay tracking

When time-domain tuning is active, the delay engine tracks delay contributions on a per-layer basis. This means that for a net routed on multiple layers, you can see how much delay is contributed by each layer individually. This per-layer detail is available in the **View** → **Panels** → **Net Inspector** panel; enable **Show Time Domain Details** from the Net Inspector's settings menu to switch the per-layer columns from length to delay values. This can help identify which layer transitions are contributing the most to timing skew in a group of matched nets.

## Setting target delay

There are two ways to set a target propagation delay for a net:

- In the properties dialog for a tuning pattern, select the **Delay** radio button and enter a value in the **Target Delay** field. The delay is specified in picoseconds.
- Using a custom DRC rule with the `length` or `skew` constraint and time-based units (e.g., `ps` for picoseconds).

When using custom DRC rules, specifying the constraint value in time units automatically switches the length tuner to time-domain mode for matching nets. For example:

```
(rule "DDR data delay matching"  
  (condition "A.hasNetclass('DDR_DATA')")  
  (constraint length (min 450ps) (opt 500ps) (max 550ps)))
```

This rule sets a target propagation delay of 500 ps for all nets in the `DDR_DATA` netclass, with an allowable range of 450 ps to 550 ps. A DRC violation will be raised for any net outside this range.

You can also set delay-based skew constraints for differential pairs:

```
(rule "DDR differential skew"  
  (condition "A.hasNetclass('DDR_CLK')")  
  (constraint skew (max 5ps)))
```

#### NOTE

When a custom DRC rule uses time units, the constraint operates in the time domain. When it uses length units, it operates in the length (space) domain. You cannot mix time and length units within the `min`, `opt`, and `max` fields of a single constraint.

## Practical example: DDR4 memory bus

A DDR4 data bus typically requires tight delay matching between all data signals in a byte lane. Consider a design with the following characteristics:

- Data signals routed on `F.Cu` (microstrip) and `In1.Cu` (stripline)
- Layer change via from `F.Cu` to `In1.Cu`
- Target delay matching of 500 ps +/- 50 ps

To set up time-domain tuning for this scenario:

1. Configure the board stackup in **Board Setup** → **Physical Stackup** with accurate dielectric thicknesses and materials. Use 6 layers for this example with GND on `In1.Cu` and `In4.Cu`.
2. Create a tuning profile in **Board Setup** → **Design Rules** → **Tuning Profiles**. Name it `DDR4_Data`, select **Single**, and check **Enable time domain tuning**.
3. In the **Track Propagation** table, add rows for `F.Cu` and `In2.Cu`. Select appropriate reference layers and press the  button to auto-calculate the unit delay for each layer.
4. In the **Via Propagation** section, set the global via unit delay or add a specific override for the `F.Cu` to `In2.Cu` transition.
5. Assign the `DDR4_Data` tuning profile to your DDR data net class in the **Net Classes** page.
6. Add a custom DRC rule to set the target delay:

```
(rule "DDR4 data delay"  
  (condition "A.hasNetclass('DDR4_Data')")  
  (constraint length (min 450ps) (opt 500ps) (max 550ps)))
```

7. Use the length tuning tools to tune the data nets. The tuner will display delay in picoseconds and add meanders until the target delay is met.

## DRC integration

DRC validates both length-domain and time-domain tuning constraints. When a custom DRC rule specifies a `length` or `skew` constraint in time units, DRC checks the calculated propagation delay of each matching net against the specified bounds.

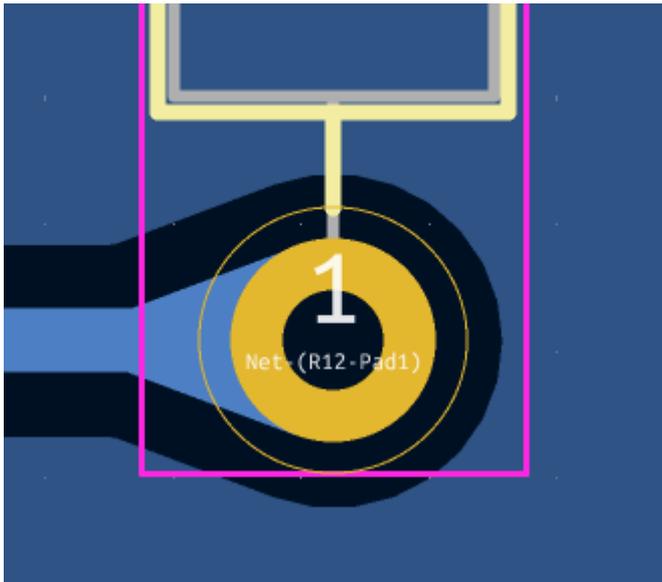
In addition, tuning profiles generate implicit DRC rules that enforce track geometry. When a tuning profile is assigned to a net class, DRC can verify that tracks on each layer match the width (and differential pair gap) defined in the tuning profile. The severity of these geometry checks is controlled by the **Tuning profile**

**track geometries** violation type in the **Violation Severity** page of Board Setup. By default this check is disabled (set to **Ignore**); set it to **Warning** or **Error** to enable it.

If a net class references a tuning profile that does not exist (for example, if a profile was deleted), DRC raises a **Missing tuning profile** violation.

## Teardrops

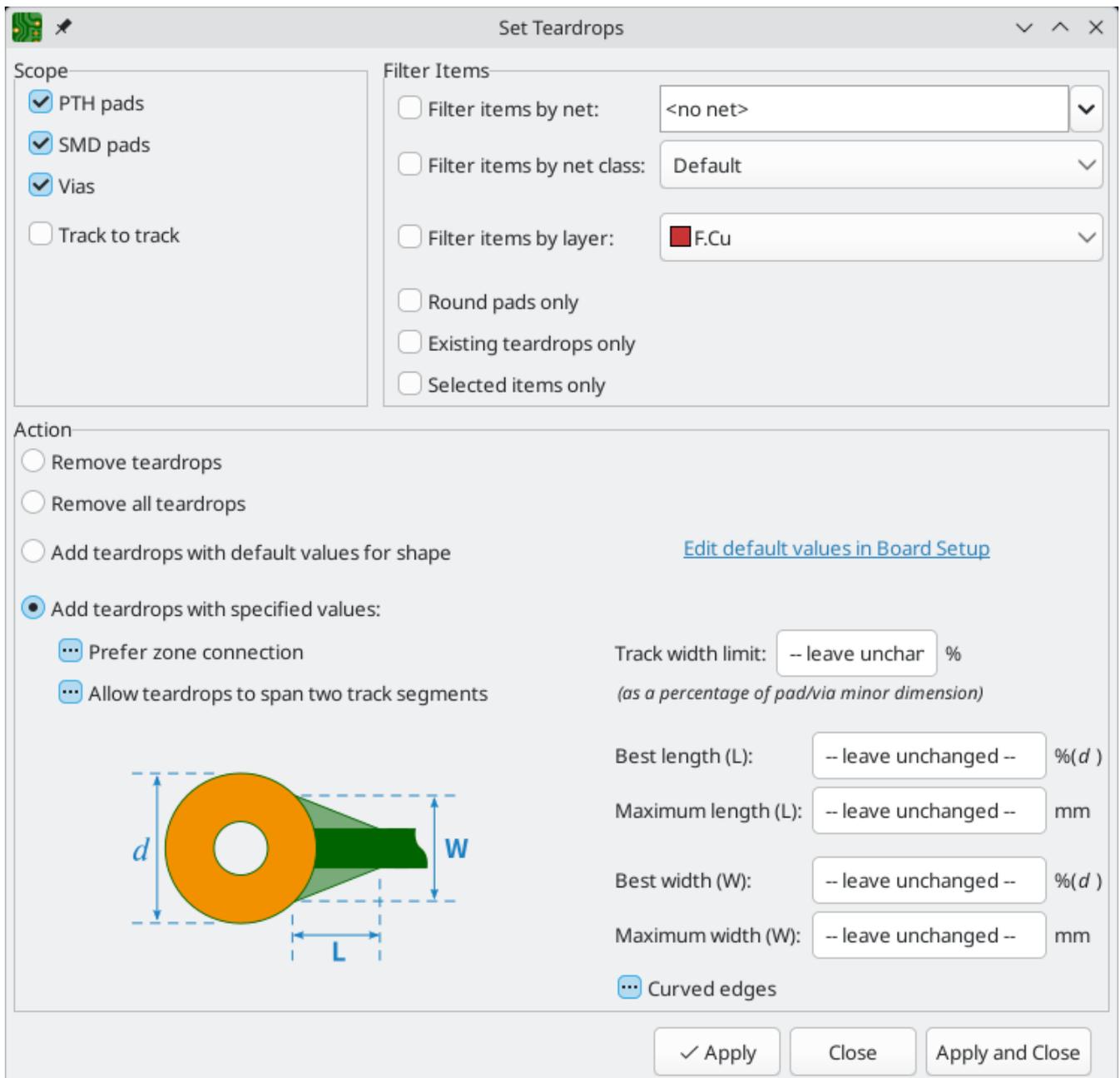
Teardrops are areas of extra copper that smooth the transition between tracks and pads, vias, or other tracks with different width. Teardrops are added to increase the mechanical robustness of a track connection. They also reduce the risk of a misaligned drill hole disconnecting a track from a drilled pad or via.



There are two ways to add teardrops to your design. You can add them in bulk using the Edit Teardrops dialog, or you can add them to individual pads and vias in the respective properties of the pad or via.

### Adding teardrops in bulk

The Edit Teardrops dialog (**Edit** → **Edit Teardrops...**) lets you add teardrops to many board objects at once. The dialog has controls for filtering which objects are affected and settings for configuring the shape of the new teardrops. It also lets you edit or remove existing teardrops.



The **Scope** section controls which types of objects will be affected: PTH pads, SMD pads, vias, and/or track-to-track connections. The **Filter Items** section lets you filter objects by other criteria; you can filter items by net, net class, and layer, or choose to act only on round pads, pre-existing teardrops, or the objects in your selection.

The **Action** section controls whether to add or remove teardrops, as well as the size and shape of the new teardrops.

**Remove Teardrops** will remove teardrops that match the scope and filtering options at the top of the dialog. **Remove All Teardrops** will remove all teardrops on the board, even if they do not match the scope and filters.

**Add teardrops with default values for shape** will add teardrops with the configured default teardrop settings to every board object that matches the scope and filters. To configure the default teardrop settings, click the **Edit default values in Board Setup** link or manually open the **Teardrops** panel in [Board Setup](#). The defaults are configured separately for teardrops connecting to round shapes, rectangular shapes, or between tracks.

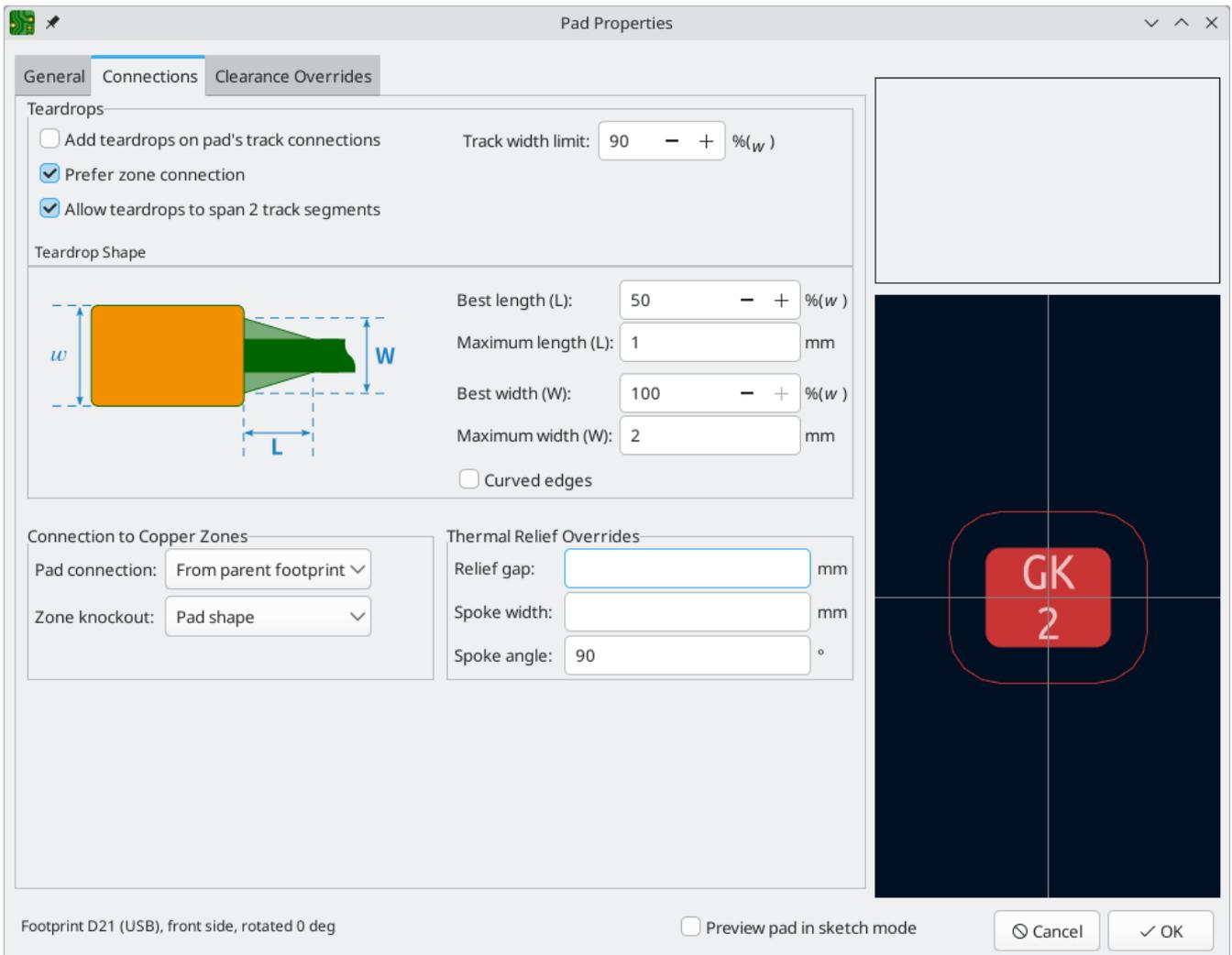
Instead of using the default values, you can provide custom teardrop settings by selecting **Add teardrops with specified values**. The available teardrop settings are:

- **Prefer zone connection:** if selected, a teardrop will not be created if the object is also connected to a zone.
- **Allow teardrops to span 2 track segments:** if selected, the teardrop will be able to spread over a second track segment if the first segment is too short to support a full teardrop.
- **Maximum track width:** a teardrop will not be created for a track connection that is wider than this percentage of the pad width (minimum pad dimension).
- **Best length:** the ideal length of the teardrop, as a percentage of the width (smallest dimension) of the attached object.
- **Maximum length:** the maximum length of the teardrop, as an absolute length.
- **Best width:** the ideal width of the teardrop, as a percentage of the width (smallest dimension) of the attached object.
- **Maximum width:** the maximum width of the teardrop, as an absolute width.
- **Curved edges:** if selected, the teardrop edges will be curved instead of a straight line.

Adding a teardrop to an object that already has a teardrop will update the existing teardrop with the new settings. However, you can leave any existing teardrop setting in an object unchanged by setting the value to `-- leave unchanged --` in a textbox, or by selecting the third, indeterminate state for a checkbox. Any value set this way will not be updated in the targeted objects' teardrop settings.

## Adding teardrops to individual objects

Rather than in bulk, you can add or edit teardrops for individual vias in the properties dialog for that via, or for individual pads in the **Connections** tab of the pad's properties dialog. The settings in the properties dialogs are the same as in the Edit Teardrops dialog. You can also edit teardrops for individual pads and vias with the [Properties Manager](#).



## Other details about teardrops

Teardrops in KiCad are small zones, meaning that when they refill they avoid shorting to copper objects on other nets. They are initially filled when they are added, but they are unfilled and refilled with other zones on the board: when using the Unfill All Zones and Refill All Zones commands, running DRC, generating fabrication outputs, etc. Teardrops can be shown in filled or outline mode using the zone display controls in the left toolbar.

Teardrops can be added to any type of pad, including custom pads. Some custom pad shapes may produce undesirable teardrop shapes. In those cases, it may be preferable to disable teardrop generation for those specific pads.

## Backdrills and hole post-machining (counterbores/countersinks)

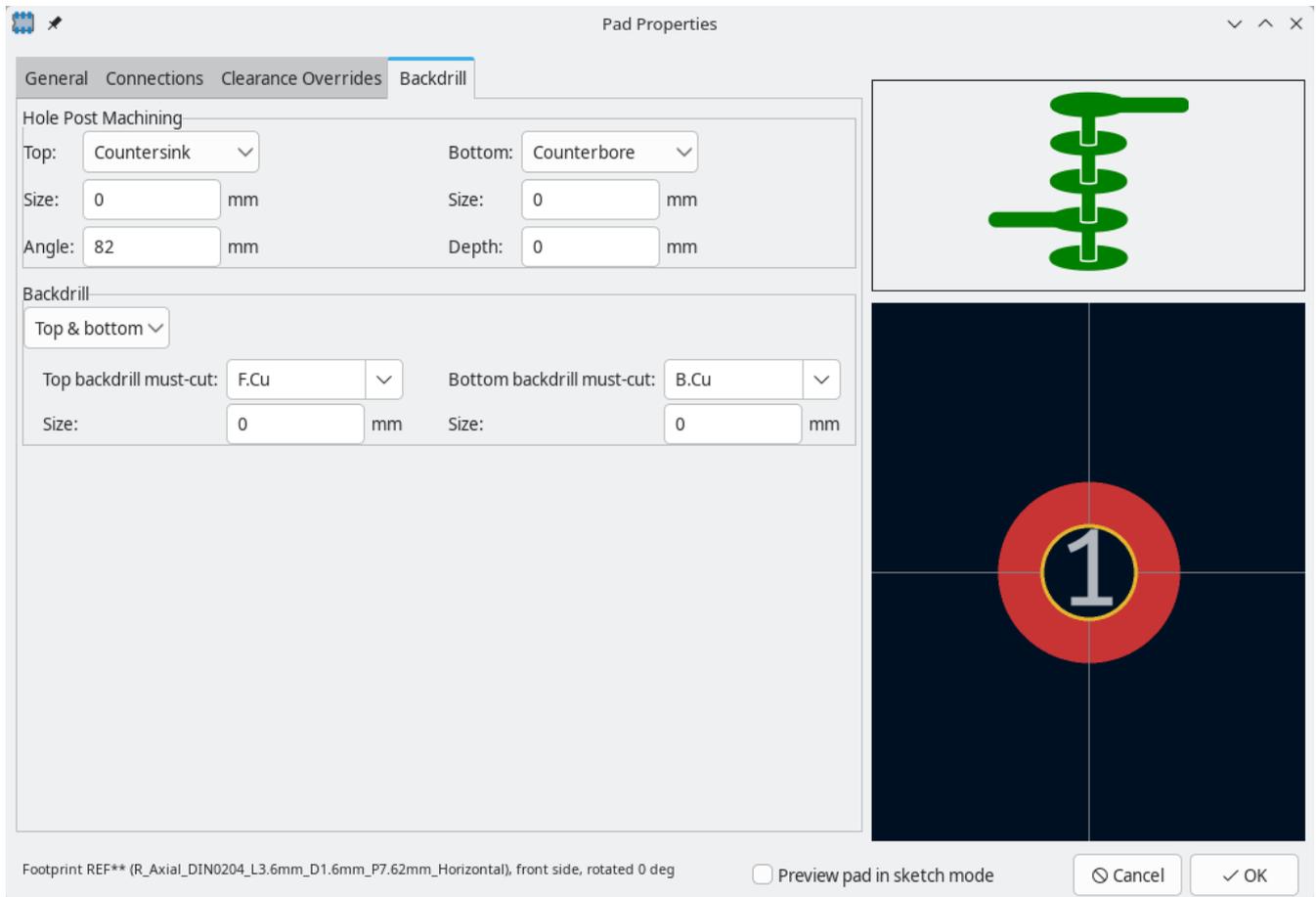
Backdrilling and post-machining are post-processing steps that can be applied to vias and through hole pads that modify the existing via or pad hole.

*Backdrilling* removes the plating from a plated hole up to a certain layer in order to eliminate the unused portion of the hole stack (a *stub*). Stubs cause reflections at high signal speeds, so avoiding stubs can improve signal integrity. Backdrilling is performed by partially re-drilling an existing hole to a larger size. The hole is only re-drilled to the depth necessary to remove the stub, and the enlarged diameter must be big enough to completely remove the plating on the hole wall.

*Post-machining* adds a countersink or counterbore to the front and/or back side of an existing pad or via hole. This creates clearance for the head of a fastener installed in the hole. Countersunk holes are conical, meaning the sides of the cutout are angled. Counterbored holes have straight sides and a flat bottom.

## Backdrills

You can add backdrills to vias using the [Via Properties dialog](#) and to pads using the **Backdrill** tab of the [Pad Properties dialog](#).



You can choose to backdrill the front, back, or both sides of a pad or via hole. For each of the selected sides, you can select the **must-cut layer** and **size**. The must-cut layer is the deepest copper layer that will be removed by the backdrilling operation, beginning at the selected side of the board. For example, for a backdrill beginning on the front side of the board, the backdrill will enlarge the hole beginning at the front of the board and continuing through the selected must-cut layer. The size is the diameter of the enlarged hole.

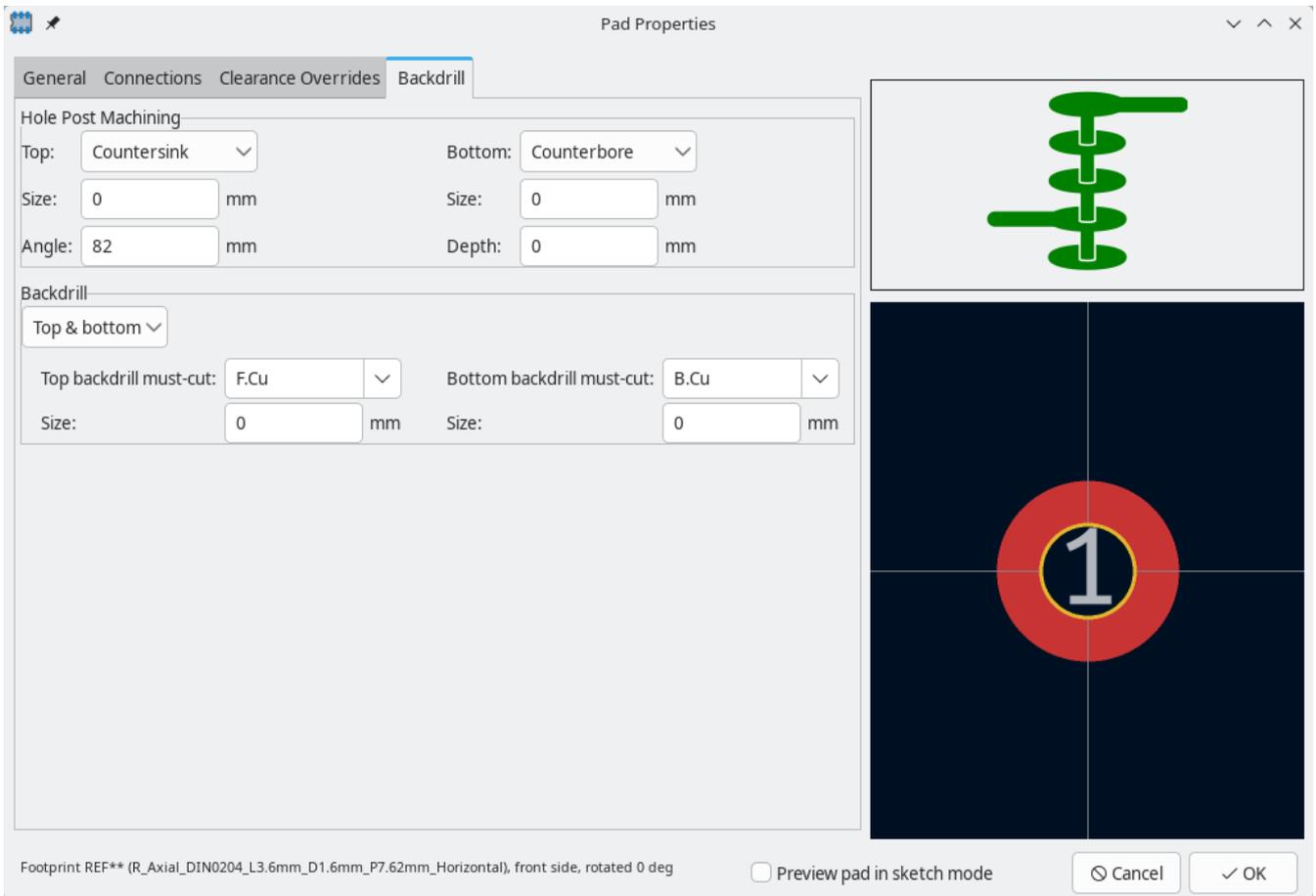
Backdrills are shown in the canvas as a ring drawn with the backdrill's diameter, half in the backdrill's outer layer color and half in the must-cut layer's color. For pads, the pad layer after the must-cut layer is drawn below the backdrill ring.



*A pad with a backdrill from F.Cu (red) to In3.Cu (cyan). In4.Cu (magenta) is shown as the primary pad color.*

## **Post-machining holes (counterbores and countersinks)**

You can add counterbores or countersinks to vias using the [Via Properties dialog](#) and to pads using the **Backdrill** tab of the [Pad Properties dialog](#).

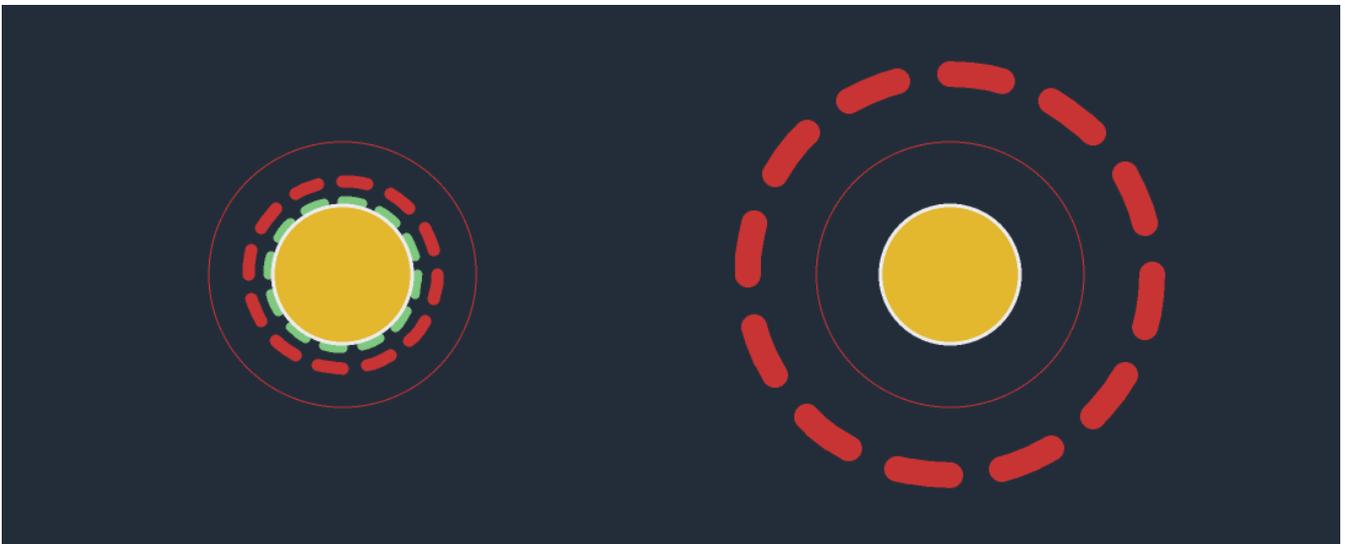


For each side (front and back) of a via or through hole pad, you can choose whether to add a countersink, a counterbore, or neither (no post-machining).

For countersinks, you can configure the **size** (outer diameter) and **angle**. The angle must be between 0 and 180 degrees. 0 degrees is the steepest possible countersink (the edges are vertical); 180 degrees is the shallowest (the edges are horizontal).

For counterbores, you can configure the **size** (outer diameter) and **depth**.

Post-machined pads are shown in the canvas with additional dashed rings drawn around them. Each dashed ring represents the intersection of the post-machined feature with a copper layer. The color of each dashed circle represents the intersecting layer.



Left: via with countersink. Right: via with counterbore.

## Graphics and text

Graphical objects (lines, arcs, rectangles, circles, polygons, text, tables, and dimensions) can exist on any layer. They exist primarily for aesthetics and documentation, although shapes on copper layers can make electrical connections and have nets assigned.

## Graphical shapes

Graphical shapes are geometric objects that can be drawn on any board layer.

When they are drawn on copper layers, graphical objects can be assigned nets and make connections to other copper objects, much like tracks and zones. There are differences between copper shapes and tracks or zones, however:

- The shape of a graphical object is exactly defined by its own properties (size, position, line width, fill, etc.) and is not affected by other nearby objects. In contrast, a zone fills the area within a specified outline, but avoids different-net copper items to automatically maintain a specified clearance.
- Graphic lines and arcs are edited as simple shapes; the [interactive router](#) is not used for drawing or modifying them. Therefore collisions with other items are not detected interactively as they would be when routing tracks (although they will be detected by [DRC](#)).

The buttons on the right toolbar can be used to create:

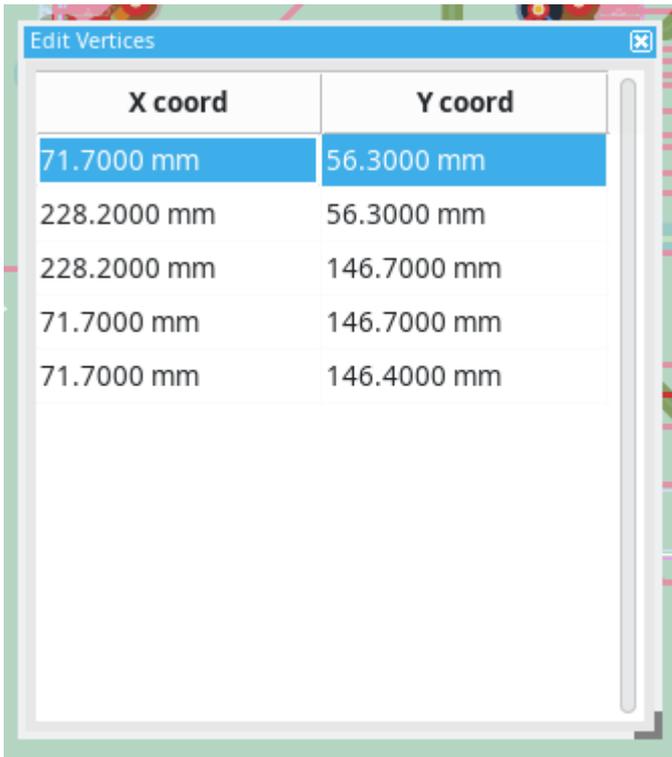
- Lines ( , default hotkey Ctrl + Shift + L )
- Arcs ( , default hotkey Ctrl + Shift + A )
- Bezier curves ( , default hotkey Ctrl + Shift + B )
- Rectangles and rounded rectangles ( )
- Circles ( , default hotkey Ctrl + Shift + C )
- Polygons ( , default hotkey Ctrl + Shift + P )

To place a shape, select the tool, then click in the canvas to place the shape's first point. Click again to place the shape's second point. For rectangles and circles, placing the second point will fully define the shape and finish drawing it. Some shapes require three or more points to be placed, however. Arcs require three points, while lines, polygons and bezier curves can accept an arbitrary number of points, and require a double click to complete.

To modify an existing graphical object, select it, then drag its editing handles to change the shape. Moving a handle at the corner of a shape will move that corner. Moving a handle on the edge of a shape will move that edge in a direction perpendicular to the edge. Normally, dragging an edge maintains the angles of the corners adjacent to the edge while allowing the edge's length to vary. Holding Ctrl instead holds the edge's length constant and allows the adjacent corner angles to vary.

To precisely position a corner, right click the corner's handle and choose **Shape Modification** → **Move Corner To...**, then enter new X and Y coordinates for the corner. For polygons, there is an additional tool that lets you edit the coordinates of each outline point as a table, which you can open by right clicking the

polygon and choosing **Shape Modification** → **Edit Corners....** This opens a floating dialog with a table containing the coordinates of every corner. Editing the coordinates of a corner immediately updates the polygon.



The image shows a floating dialog box titled "Edit Vertices" with a close button in the top right corner. It contains a table with two columns: "X coord" and "Y coord". The table has five rows of data. The first row is highlighted in blue.

X coord	Y coord
71.7000 mm	56.3000 mm
228.2000 mm	56.3000 mm
228.2000 mm	146.7000 mm
71.7000 mm	146.7000 mm
71.7000 mm	146.4000 mm

Arcs have two vertex editing modes, which are selectable in **Preferences** → **PCB Editor** → **Editing Options** or by right clicking the  button on the right toolbar.

- The first mode (**keep arc center, adjust radius**) maintains the position of the arc center as the arc endpoints or midpoint are dragged, changing the radius as necessary.
- The second mode (**keep arc endpoints or direction of starting point**) maintains the position of the arc endpoints and the arc's direction of curvature as the midpoint or center are dragged.

Just like with tracks, you can expand a selection from one graphic line to include all other contiguous graphic lines by pressing .

The properties of a graphic shape can be adjusted in the shape's properties dialog or with the [Properties Manager](#).

●
●
●

## Segment Properties

By Endpoints
By Length and Angle
By Start/Midpoint

Start Point

X:  mm

Y:  mm

End Point

X:  mm

Y:  mm

Locked

Line width:  mm

Line style:  Solid ▼

Net: <no net> ▼

Layer:  F.Cu ▼

Technical Layers

Solder mask      Expansion:  mm

Cancel
OK

- The top section contains controls for editing the object's location and shape. Some types of objects can be edited in multiple ways, with each method in its own tab. For example, a line segment can be edited by its start and end points, by its start point, length, and angle, or by its start and mid points.
- **Locked** controls whether or not the text object is [locked](#). Locked objects may not be manipulated or moved, and cannot be selected unless the **Locked Items** option is enabled in the Selection Filter panel.
- Rectangles can have their corners rounded by checking the **Rounded rectangle** checkbox. The corner radius can be adjusted with the **Corner radius** parameter or by dragging the radius handle in the editing canvas. Rounding the corners of a rectangle is a non-destructive action: the corner radius can be adjusted or removed at any time.
- The **Line width** option controls the width of the outline, even for filled objects. The outline width extends on both sides of the "ideal" shape of the graphic object. For example, a graphic circle that is defined to have 2mm radius and 0.2mm line width will consist of a torus with an outer radius of 2.1mm

and inner radius of 1.9mm. If the shape is filled and the line width is set to 0, the shape will be a filled circle with 2mm radius. Several line styles are available in the **Line style** dropdown: solid, dashed, dotted, dash-dot, and dash-dot-dot.

#### NOTE

You can customize the default style of newly-created graphical shapes in the Text & Graphics Defaults section of the Board Setup dialog.

- Closed shapes (rectangles, circles, and polygons) can be outlines or filled shapes, which is controlled by the **Filled shape** checkbox.
- The **Layer** dropdown controls which layer the shape is placed on. Graphical shapes on copper layers can have a net assigned in their properties dialog. Copper shapes with a net make connections like tracks or zones. Unlike zones, copper graphical objects always maintain their shape and do not keep clearance to other copper objects.
- When shapes are placed on outer copper layers, they can be configured to affect the corresponding solder mask layer in addition to their primary copper layer by enabling the **Solder mask** checkbox. When enabled, a shape on the front copper layer will also be drawn on the front solder mask layer, while a back copper shape will also be drawn on the back solder mask layer. Because solder mask layers are negative, this will result in a solder mask opening with the same shape as the copper shape. The **Expansion** textbox controls the size of the mask opening relative to the original copper shape: the expansion value will be added to each side of the original shape to form the mask shape. For example, a 1mm wide copper segment with a 1mm expansion would result in a 3mm wide mask cutout, because the 1mm expansion is added to both sides of the segment.

## Shape modification tools

KiCad has several tools for modifying combinations of graphic shapes in useful ways, such as chamfering two lines or combining two polygons. These tools are used by selecting the shapes or corners you want to modify, right clicking, and then choosing the relevant tool in the **Shape Modification** submenu. Different tools are available for different combinations of selected shapes or corners.

- **Heal Shapes** fixes a discontinuity between two lines or arcs. A new line segment is added to connect the ends of each shape together, up to a specified tolerance.
- **Simplify Polygons** removes superfluous corners from the selected shape with a configurable tolerance. Corners are removed if they are more than the specified distance from the line between their two neighboring corners.
- **Fillet Lines** adds an arc to round the corner between two connected lines with a specified radius. The two original lines are shortened to meet the endpoints of the arc. Note that for rectangles, you can instead round the corners by turning the rectangle into a [rounded rectangle](#).
- **Chamfer Lines** adds a line segment to create a new edge between two connected lines with a specified setback. The two original lines are shortened to meet the endpoints of the new segment.
- **Dogbone Corners** adds circular reliefs to the corners of the selected shapes. This is similar to filleting, but the modified shape is larger than the original, with the added arcs intersecting the vertices of the original corners. In other words, the added reliefs exactly enclose the original corners. This can be useful for relieving the corners of interior cutouts so that they can be manufactured using a round cutting tool. Because dogbones are intended to allow interior corners to be manufactured, a dogbone is only added to

a corner if the corner points into the body of the board as defined by the [board outline](#). There is an option to **Add slots to acute corners**, which adds extended slots in corners that are too narrow to reach with a cutting tool of the selected radius.

- **Extend Lines to Meet** lengthens two selected lines until they intersect each other. The two lines will share a coincident endpoint.
- **Move Corner To** moves the selected corner to a specific X and Y coordinate.
- **Move Midpoint To** moves the selected midpoint to a specific X and Y coordinate, maintaining the line's slope and moving the adjacent corners as necessary.
- **Create Corner** adds a new corner to the outline while maintaining the original outline shape.
- **Remove Corner** deletes the selected corner from the outline.
- **Chamfer Corner** deletes the selected corner and adds new corners next to the original corner so that the original corner shape becomes chamfered.
- **Edit Corners** opens a dialog to edit the coordinates of the selected shape's corners.
- **Merge Polygons** combines two or more selected polygons into one new polygon that is the union of the original shapes.
- **Subtract Polygons** subtracts one or more polygons from another polygon, resulting in a new polygon that is the difference of the original shapes. The first-selected polygon(s) are subtracted from the last-selected polygon.
- **Intersect Polygons** results in a new polygon that is the shape of the overlapping area between two or more selected polygons.

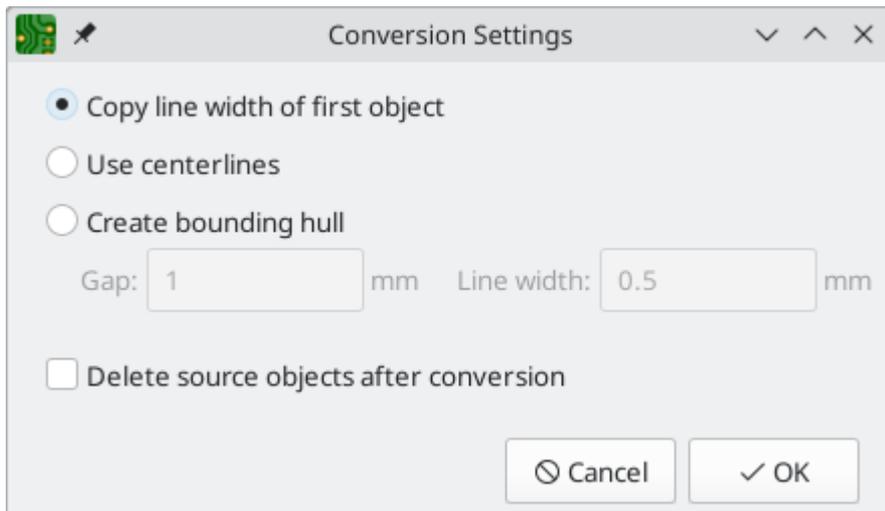
## Converting objects to and from graphic shapes

KiCad provides tools to convert graphic objects to other types of objects, other types of objects to graphic objects, and graphic objects to other kinds of graphic objects. These tools are used by selecting the shapes you want to convert, right clicking, and then choosing the desired result object from the **Create From Selection** submenu. Most types of object conversions have several conversion options that are presented in a settings dialog. The exact options differ based on the target object type.

When converting to a graphic polygon, rule area, or zone, there are several options for how to convert the source objects into a polygonal outline.

- If **copy line width of first object** is selected, an unfilled polygon will be created that has its line width taken from the line width of the first selected source object. This option is only available when converting to a graphic polygon, and the source object must be a closed shape.
- If **use centerlines** is selected, an object with zero line width will be created, with its outline placed at the centerlines of the source objects. The source object must be a closed shape. If the target object is a graphic polygon, it will be filled.
- If **create bounding hull** is selected, an object will be created with the specified **line width**. The object's outline will be offset from the outermost extents of the source object by the specified **gap**. The source object does not need to be a closed shape when a bounding hull is created.

Most conversions provide a **delete source objects after conversion** option, which will result in the original object being deleted during the conversion, only leaving the new object in place. If this option is not selected, the conversion will leave the original object in place in addition to the new object. The original object will be selected following the conversion so that it can be manually deleted by pressing .



The following conversion types are available:

- **Create Polygon From Selection** converts a graphic shape, text, zone, rule area, or track into a polygon. This can be used to convert separate graphic shapes, such as lines and arcs, into a unified shape. It can also be used to convert a text object into a shape that can have its outline manipulated graphically.
- **Create Zone From Selection** converts a graphic shape, text, zone, rule area, or track into a zone. In addition to the conversion settings, the conversion dialog also shows options for [configuring the resulting zone](#). This can be used to create zone outlines with complex shapes, such as curves, that would otherwise be difficult to create using the zone tool.
- **Create Rule Area From Selection** converts a graphic shape, text, zone, rule area, or track into a rule area. In addition to the conversion settings, the conversion dialog also shows options for [configuring the resulting rule area](#). This can be used to create rule area outlines with complex shapes, such as curves, that would otherwise be difficult to create using the rule area tool.
- **Create Lines From Selection** converts a graphic polygon or rectangle into graphic lines that follow the source shape's outline. This can be used to convert a unified shape into its constituent outline segments.
- **Create Outsets From Selection** converts the selected object (graphic shapes, pads, etc.) into a rectangle that surrounds the original shape with some spacing.

**TIP**

The outset tool can be used to quickly create outlines, courtyards, etc., especially in combination with other shape modification tools.

- The **Outset** distance specifies the minimum distance between the outset and the original shape. There will always be at least this much space between the two shapes. If the distance is positive, the outset will be larger than the original shape. If the distance is negative, the outset will be smaller than the original shape.
- If **Round corners (when possible)** is enabled, the outset will have rounded corners rather than being a simple rectangle.
-

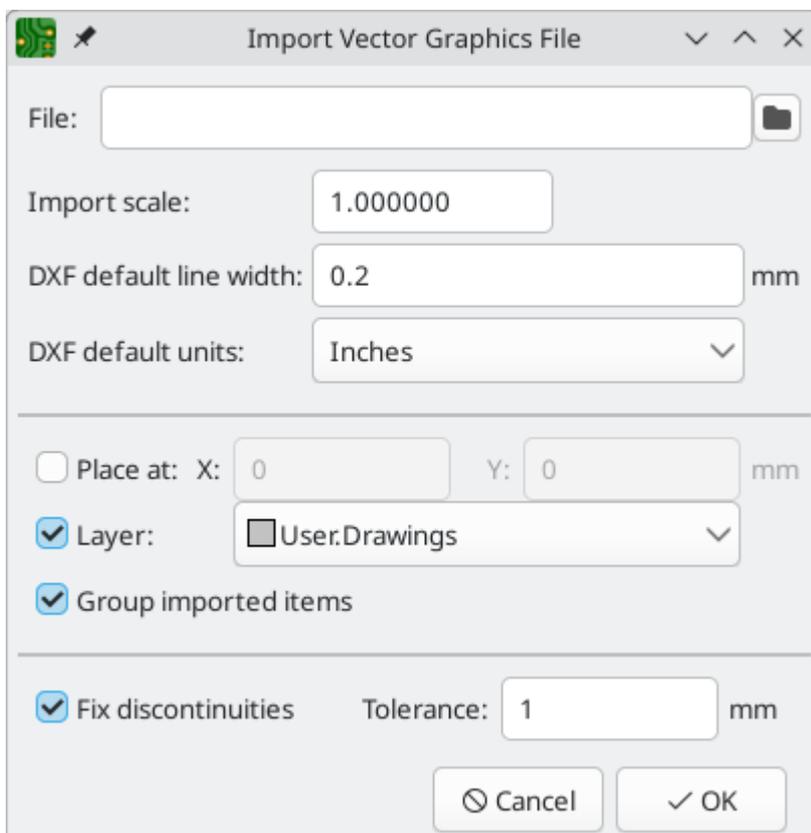
If **Round outwards to grid multiples (when possible)** is enabled, the outset will be placed on the specified grid, rounding outwards when necessary so that the specified outset distance is maintained.

- If **Copy item layers** is enabled, the outset will be drawn on the same layer as the original shape. If disabled, the outset will be drawn on the layer selected in the dropdown menu.
- If **Copy item thickness (when possible)** is enabled, the outset will be drawn with the original item's thickness if that item has a thickness, or the specified thickness otherwise. Some source items, like pads, do not have a thickness property. The specified thickness will always be used when this option is disabled.
- **Create Tracks From Selection** converts a graphic shape, zone, or rule area into tracks that follow the source shape's outline. If the source object is not on a copper layer, a dialog will be presented to specify the target copper layer. The source object is not removed following conversion, but remains selected so that it can be easily deleted if desired.
- **Create Arc From Selection** converts a graphic line segment or track segment into a graphic arc. The arc's endpoints are placed at the endpoints of the source segment and its thickness is taken from the source object's line thickness. The source segment is not removed following conversion, but remains selected so that it can be easily deleted if desired.

## Importing vector graphics

You can add graphic shapes from an external vector graphics file by importing the file into KiCad. DXF and SVG files are supported. To import the file, use **File** → **Import** → **Graphics...** (**Ctrl** + **Shift** + **F**).

Imported vector graphics are part of the design like any other graphic shape. In other words, they have an assigned board layer, they are included in fabrication outputs, and shapes on copper layers can make electrical connections.



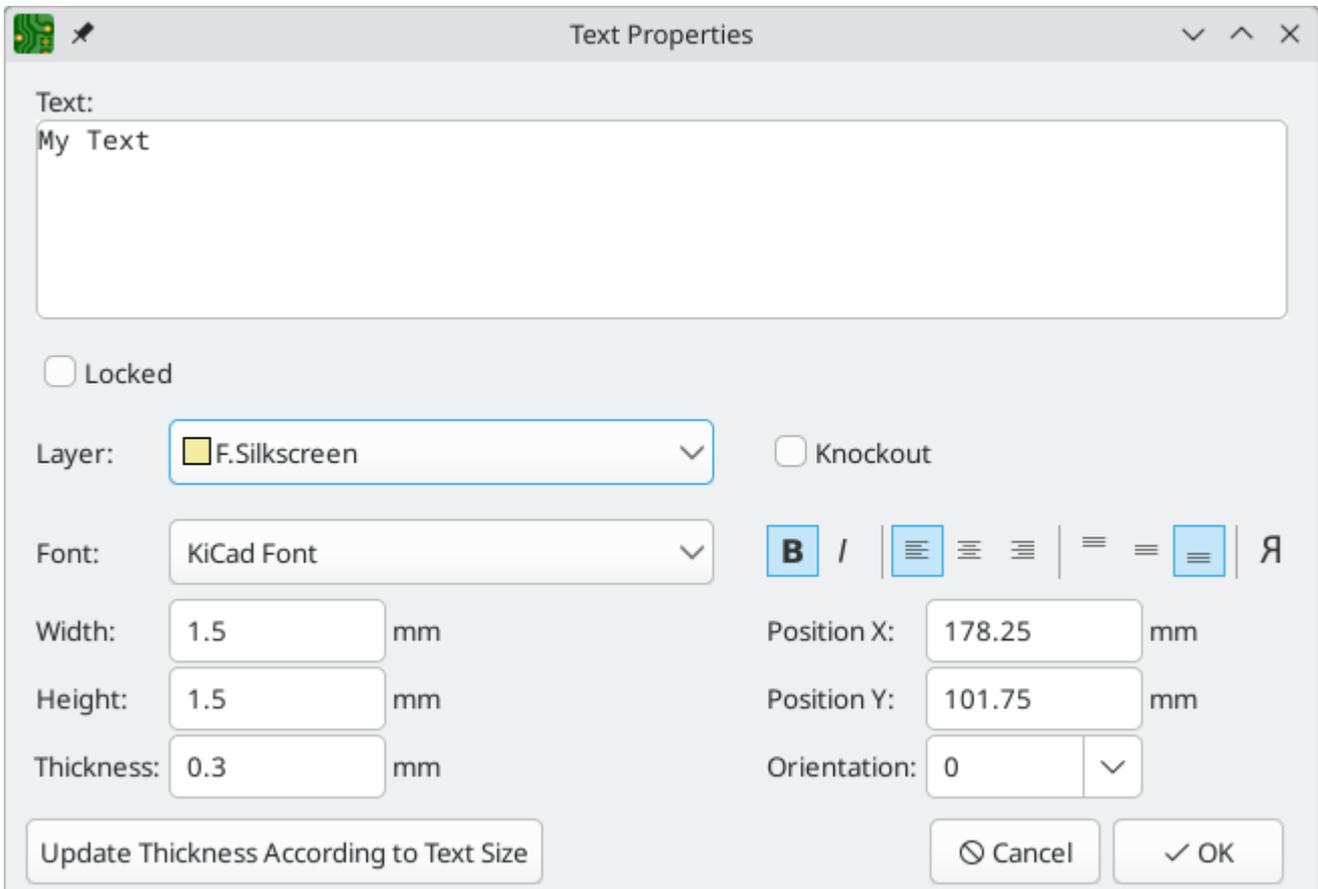
The Import Vector Graphics File dialog has several options:

- **File** specifies the vector graphics file to import.
- **Import Scale** sets the scale factor for the import.
- **DXF default line width** sets the line width for any items in a DXF file that do not specify a line width. It has no effect when not importing DXF files.
- **DXF default units** sets the default unit for DXF files with unspecified units. It has no effect when not importing DXF files.
- If **Place At** is enabled, the imported shapes are placed at the specified location, relative to the PCB Editor's [page origin](#). If it is disabled, the imported shapes are placed interactively.
- If **Layer** is enabled, the imported shapes are placed onto the selected layer. If it is disabled, the shapes are placed onto the active layer.
- If **Group imported items** is enabled, all shapes imported from the vector graphics file are added to a [group](#).
- If **Fix discontinuities** is enabled, any shape discontinuities smaller than the specified **tolerance** are filled by extending each segment until they intersect or adding an additional segment.

## Text objects

Graphical text may be placed by using the **T** button in the right toolbar or by keyboard shortcut **Ctrl** + **Shift** + **T**. Activating the tool brings up a text properties dialog. After configuring the text and its properties and accepting the dialog, you can click in the canvas to place the text.

You can also add text boxes, which are similar to regular text except that they have an optional border and they automatically reflow text within that border. Text boxes are placed with the  button, and require clicking twice to specify the top left and bottom right corners of the box.



- **Locked** controls whether or not the text object is [locked](#). Locked objects may not be manipulated or moved, and cannot be selected unless the **Locked Items** option is enabled in the Selection Filter panel.
- **Layer** controls the text's layer. Text may be placed on any layer, but note that text on copper layers cannot be associated with a net and cannot form connections to tracks or pads. Copper zones will fill around the rectangular bounding box of text objects.
- There are several formatting options: text can be bolded, italicized, left/right/center aligned, top/bottom/center aligned, and reversed.
- The **knockout** option adds a solid rectangle surrounding the text and makes the text itself a negative cutout. This feature is only available for regular text objects, not text boxes.
- The **Font** dropdown lets you select a font for the text. You can use any TTF font available on your system, or the built-in KiCad stroke font.

#### NOTE

User fonts are not embedded by default in the project. If the project is opened on another computer that does not have the selected font installed, a different font will be substituted. You can optionally embed into the board file any fonts used by the design in the [Embedded Files](#) section of [Board Setup](#). For maximum compatibility without embedding, use the KiCad font. Also consider converting text objects to polygons before sharing a project (right click a text object → **Create from Selection** → **Create Polygon from Selection...**). Text converted to polygons is not editable as text, but will render identically on any computer.

- You can adjust the text size with the **Text width** and **Text height** controls. When you are using the KiCad font, you can also adjust the stroke width with the **Thickness** control. When the  button is pressed the

text thickness is automatically adjusted according to the text size: the thickness for normal text is set to the size divided by 8, and the thickness for bold text is set to the size divided by 5.

- **Position X** and **position Y** control the text object's location. These properties are not available for text boxes.
- **Orientation** is the rotation angle of the text object. You can select an angle in 90 degree increments from the dropdown, or type in an arbitrary angle.

Text boxes additionally have options controlling their border.

- The **border** checkbox makes the border visible or invisible. For visible borders, you can adjust the border's thickness with the **border width** control and the line style with the **border style** control (solid, dashed, dotted, dash-dot, or dash-dot-dot).
- The **margins** between the border and the text on each side of a text box can be set using the [Properties Manager](#). Margins cannot be set in the Text Box Properties dialog.

**NOTE**

You can customize the default style of newly-created text objects in the Text & Graphics Defaults section of the Board Setup dialog.

Finally, text supports markup for superscripts, subscripts, overbars, evaluating project variables, and accessing symbol field values.

Feature	Markup Syntax	Result
Superscript	<code>text^{superscript}</code>	<code>text<sup>superscript</sup></code>
Subscript	<code>text_{subscript}</code>	<code>text<sub>subscript</sub></code>
Overbar	<code>~{text}</code>	<code>text</code>
<a href="#">Project text variables</a>	<code>\${variable}</code>	<code>variable_value</code>
<a href="#">Built-in text variables</a>	<code>\${refdes:field}</code>	<code>field_value</code> of symbol <code>refdes</code>

**NOTE**

Project text variables must be defined in [Board Setup](#) before they can be used. There are also a number of [built-in text variables](#).

## Tables

You can use a table to organize text in a tabular format. Tables have customizable borders, cell sizes, and headers, and can be placed on any layer.

Heading 1	Heading 2
Column 1 Row 1	Column 2 Row 1
Column 1 Row 2	Column 2 Row 2

To place a table, use the  button in the right toolbar. Click in the canvas to place the top left corner of the table, then click again to place the bottom right corner of the table and finish drawing the table. The bigger you draw the table, the more rows and columns will be added by default, but [rows and columns can be added or deleted](#) after the table is created.

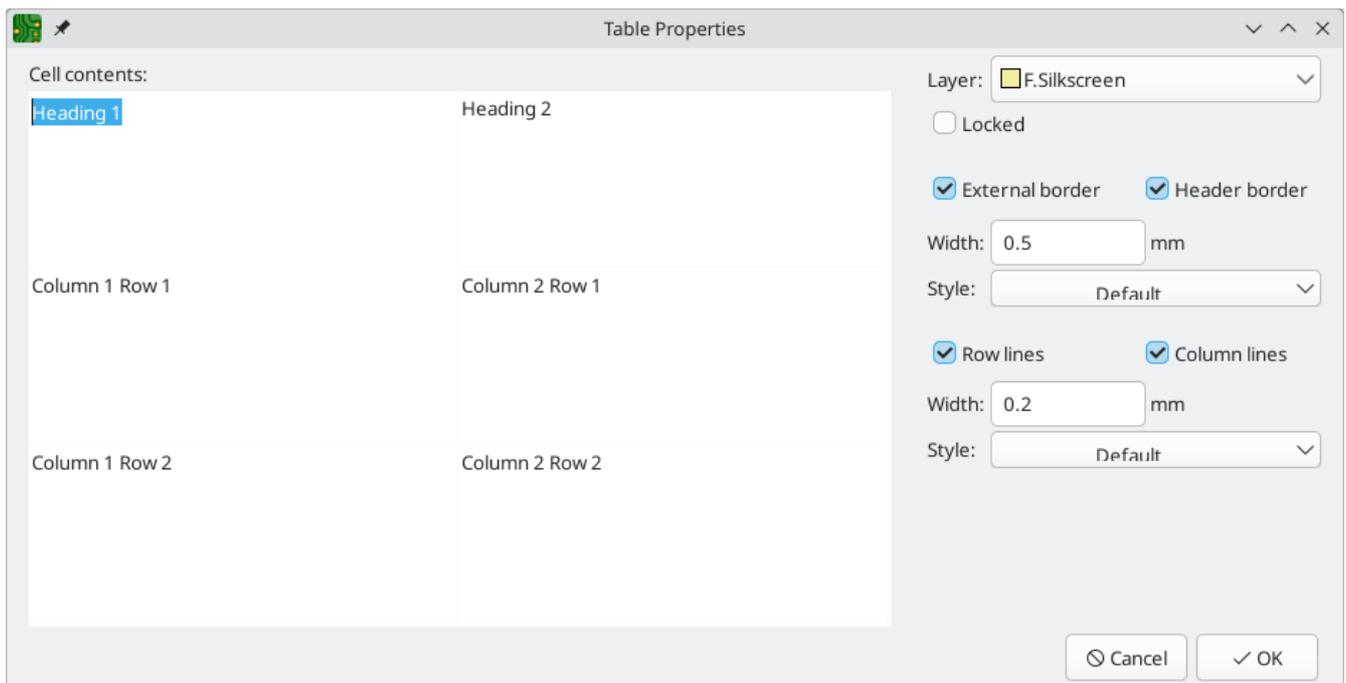
Once you have created a table, you can [edit the table as a whole](#) or [edit cells individually](#). Creating a new table automatically opens the Table Properties dialog to edit the entire table.

You can export a table from KiCad into a CSV file by right clicking a table or table cell and clicking **Export Table to CSV...**

## Editing a whole table

You can edit an entire table with the Table Properties dialog. There are several ways to open the Table Properties dialog:

- Create a new table. The Table Properties dialog opens automatically when the table is created.
- Select any cell in the table, right click, and select **Edit Table** (**Ctrl** + **E**).
- Select the entire table, right click, and select **Properties...** (**E**). You can select the entire table with a drag selection or by selecting a single cell, then right clicking and selecting **Select Table**.
- Click the **Edit Table...** button in the Table Cell Properties dialog.



This dialog lets you edit the properties of the entire table, including the text in each cell and the separators between cells. To change the formatting of text in a cell, edit the properties of individual cells, instead of the properties for the entire table.

### NOTE

The properties for a table can also be edited in the [Properties Manager](#) when the entire table is selected.

The left side of the dialog displays an editable grid of the entire table. You can edit the contents of any cell by clicking on the cell in the grid. You can also edit the text in a cell by selecting the cell and using the Properties

Manager. If you have tabular content in a spreadsheet or other table, you can copy and paste that content into the grid here.

**NOTE**

Text in table cells supports the markup described in the [text objects section](#) (superscripts, subscripts, strikethroughs, etc.).

The right side of the dialog contains formatting options for the table.

- The **Layer** dropdown controls which board layer the table is on.
- The **Locked** checkbox controls whether or not the table is [locked](#). Locked objects may not be manipulated or moved, and cannot be selected unless the **Locked Items** option is enabled in the Selection Filter panel.
- The **External border** and **Header border** checkboxes control whether there is a border drawn around the entire table and the cells in the top row, respectively. When **Header border** is enabled, the border below the cells in the top row is styled using these external border settings rather than the row/column line settings. The line width of the header borders is controlled by the **Width** field. The line style can be set to solid, dashed, dotted, dash-dot, or dash-dot-dot using the **Style** dropdown menu.
- The **Row Lines** and **Column lines** checkboxes enable horizontal lines between rows and vertical lines between columns, respectively. These have the same formatting options as the external and header borders.

## Editing individual table cells

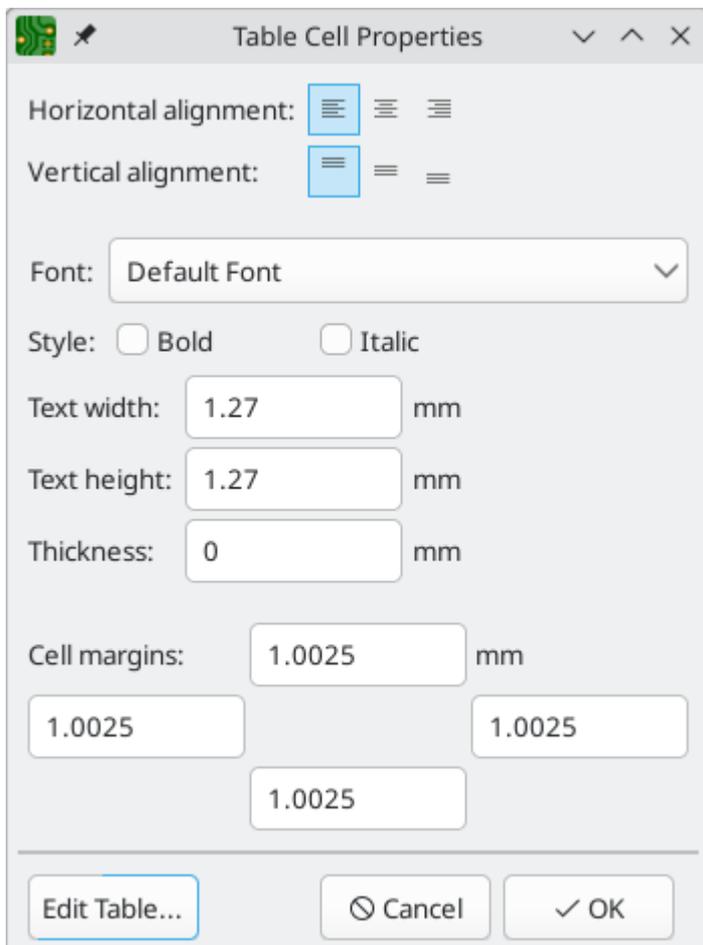
Instead of editing the properties of an entire table, you can also edit the properties of individual cells. This modifies selected cells, but does not affect other cells. To open the Table Cell Properties dialog, double click on a cell, or select a cell, right click, and choose **Properties...** (  ). If you select multiple cells, the properties dialog will act on all of them at once.

**NOTE**

You can select multiple cells by clicking and dragging.

**NOTE**

To select all cells in a row or column, select a cell in that row or column, right click, and choose **Select Row(s)** or **Select Column(s)**. You can select multiple rows or columns in this way by starting with multiple cells selected.



The left side of the dialog lets you edit the contents of the selected cell. The right side of the dialog contains formatting options for the selected cell.

- The **Font** dropdown lets you select a font for the text. You can use any TTF font available on your system, or the built-in KiCad stroke font.
- There are several formatting options: text can be bolded, italicized, left/right/center aligned, and top/bottom/center aligned.
- You can adjust the text size with the **Text width** and **Text height** controls. When you are using the KiCad font, you can also adjust the stroke width with the **Thickness** control. When the  button is pressed the text thickness is automatically adjusted according to the text size: the thickness for normal text is set to the size divided by 8, and the thickness for bold text is set to the size divided by 5.
- The **Cell margins** textboxes control the amount of spacing around the top, bottom, left, and right of the text in the cell.

You can click the **Edit Table...** button to open the properties dialog for the entire table.

#### NOTE

The properties for a table cell can also be edited in the [Properties Manager](#) when one or more table cells is selected.

## Editing table layout

The layout of a table (size and number of columns and rows) is initially set when you create a table, but you can also edit the layout after creation.

To resize a row or column, select a cell in that row or column, then drag the handle on the right (to change the column width) or the bottom (to change the row height) to the desired size.

To add rows or columns, select a cell next to where the new row or column should go, right click, then choose **Add Row Above**, **Add Row Below**, **Add Column Before**, or **Add Column After**, as desired.

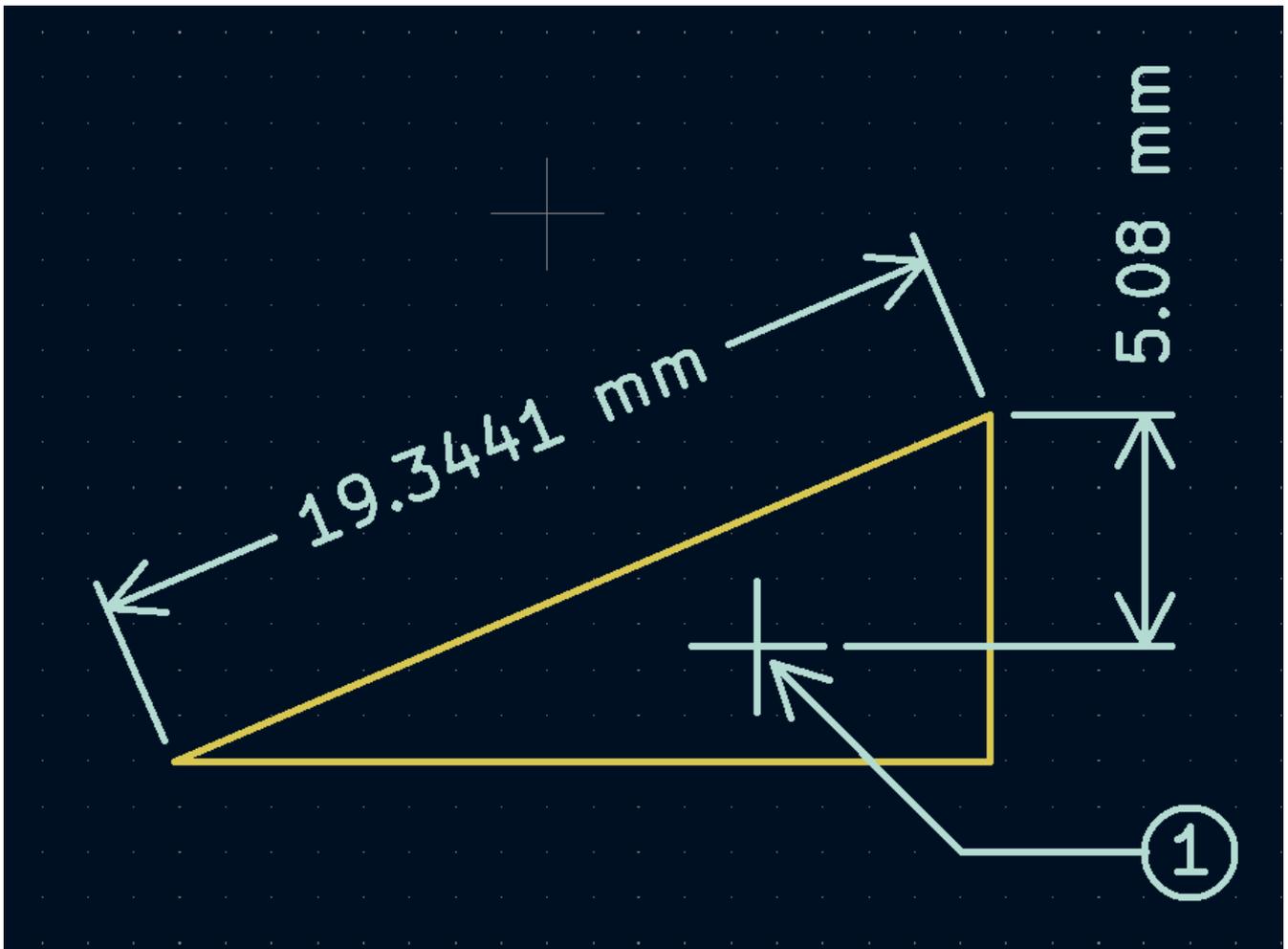
To delete rows or columns, select a cell in the row or column you want to delete, then right click and choose **Delete Row(s)** or **Delete Column(s)**. To delete multiple rows or columns, start with a selection that spans all the rows or columns you want to delete.

You can merge multiple cells into a single cell by selecting all the cells you want to merge, right clicking, and choosing **Merge Cells**. To unmerge them, select the merged cell, right click, and choose **Unmerge Cells**.

## Dimensions

Dimensions are graphical objects used to show a measurement or other marking on a board design. They may be added on any drawing layer, but are normally added to one of the User layers. KiCad currently supports five different types of dimension: aligned, orthogonal, center, radial, and leader.

- **Aligned** dimensions () show a measurement of distance between two points. The measurement axis is the line that connects those two points, and the dimension graphics are kept parallel to that axis.
- **Orthogonal** dimensions () also measure the distance between two points, but the measurement axis is either the X or Y axis. In other words, these dimensions show the horizontal or vertical component of the distance between two points. When creating orthogonal dimensions, you can select which axis to use as the measurement axis based on where you place the dimension after selecting the two points to measure.
- **Center** dimensions () create a cross mark to indicate a point or the center of a circle or arc.
- **Radial** dimensions () show a measurement between a center point and the outside of a circle or arc. The center point is indicated by a cross.
- **Leader** dimensions () create an arrow with a leader line connected to a text field. This text field can contain any text, and an optional circular or rectangular frame around the text. This type of dimension is often used to call attention to parts of the design for reference in fabrication notes.



After creating a dimension, its properties may be edited (hotkey **E**) to change the format of the displayed number and the style of the text and graphic lines.

**NOTE**

You can customize the default style of newly-created dimension objects in the Text & Graphics Defaults section of the Board Setup dialog.

**Dimension Properties**

**Dimension Format**

Value:   Override value      Units:

Prefix:

Suffix:

Layer:        Suppress trailing zeroes

Units format:

Precision:

Preview: 1.3 in

**Dimension Text**

Font:       **B** / | ≡ ≡ ≡ | Я

Width:  mm      Position X:  mm

Height:  mm      Position Y:  mm

Thickness:  mm      Orientation:

Position mode:        Keep aligned with dimension

**Dimension Line**

Line thickness:  mm      Extension line offset:  mm

Arrow length:  mm      Extension line overshoot:  mm

Arrow direction:

## Dimension format options

- **Override value:** When enabled, you may enter a measurement value directly into the **Value** field that will be used instead of the actual measured value.
- **Prefix:** Any text entered here will be shown before the measurement value.
- **Suffix:** Any text entered here will be shown after the measurement value.
- **Layer:** Selects which layer the dimension object exists on.
- **Units:** Selects which units to display the measured value in. **Automatic** units will result in the dimension units changing when the display units of the board editor are changed.
- **Units format:** Select from several built-in styles of unit display.
- **Precision:** Select how many digits of precision to display.
- **Suppress trailing zeroes:** Select whether to hide trailing zeroes in the value text.

## Dimension text options

Most of the dimension text options are identical to those options available for other graphical text objects (see the Graphical Objects section above). Some specific options for dimension text are also available:

**Position mode:** Choose whether to position the dimension text manually, or to automatically keep it aligned with the dimension measurement lines.

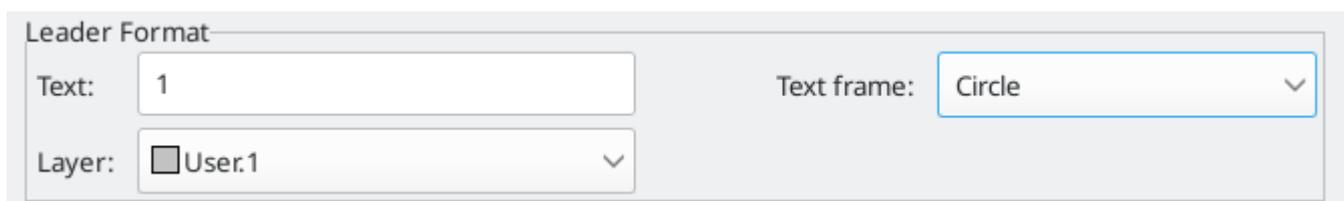
- **Keep aligned with dimension:** When enabled, the orientation of the dimension text will be adjusted automatically to keep the text parallel with the measurement axis.

## Dimension line options

- **Line thickness:** Sets the thickness of the graphical lines that make up a dimension's shape.
- **Arrow length:** Sets the length of the arrow segments of the dimension's shape. A negative arrow length reverses the arrow direction.
- **Arrow direction:** Select whether the dimension object's arrow(s) point inwards towards the value text or outwards away from the text. The arrow direction can also be set while drawing a dimension by right clicking and selecting **Switch Dimension Arrows**.
- **Extension line offset:** Sets the distance from the measurement point to the start of the extension lines.
- **Extension line overshoot:** Sets the distance from the dimension's line to the end of the extension lines.

## Leader options

Leader dimensions have unique options:



Leader Format

Text: 1

Text frame: Circle

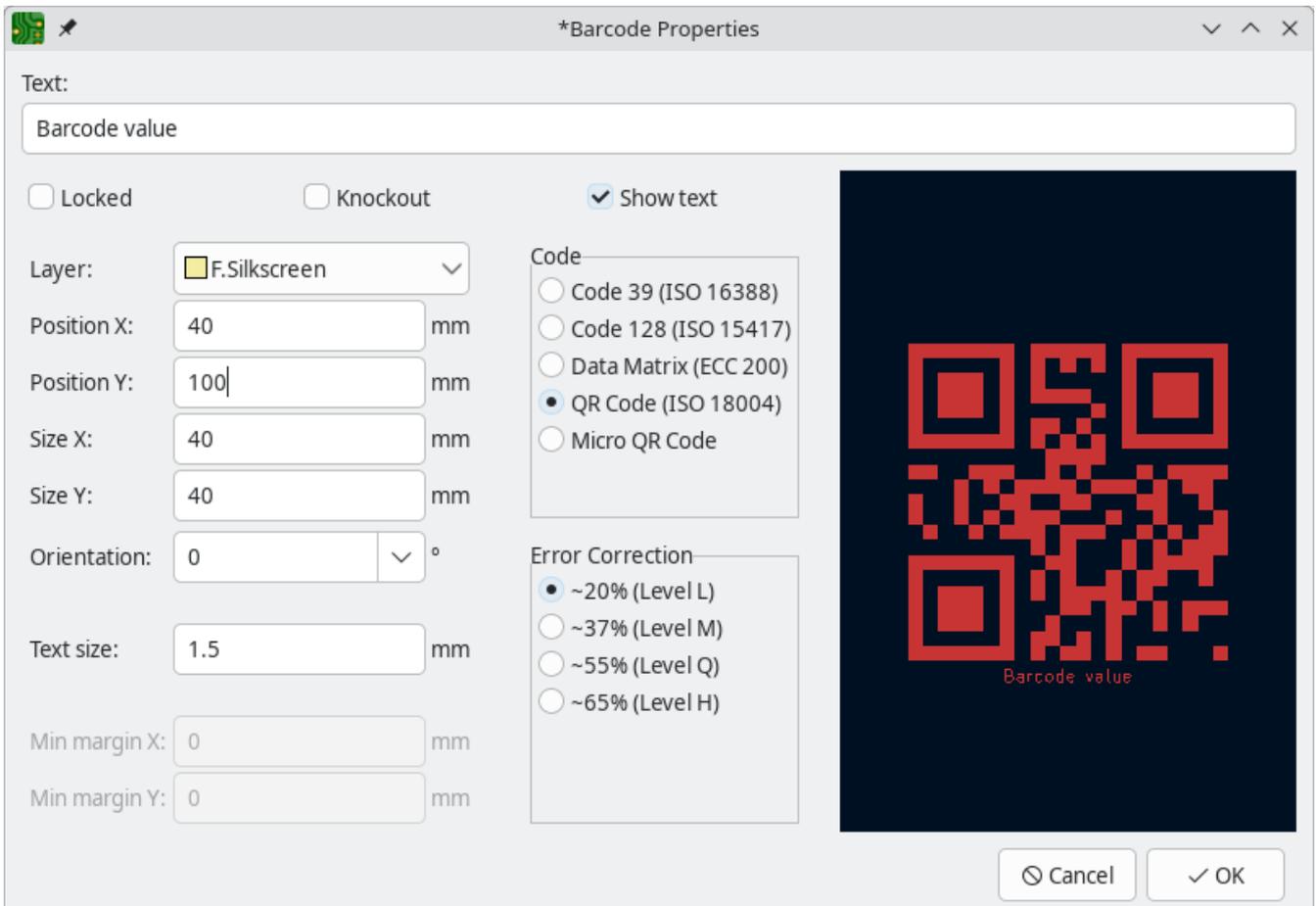
Layer:  User.1

- **Value:** Enter the text to show at the end of the leader line.
- **Text frame:** Select the desired border around the text (circle, rectangle, or none).

## Barcodes

KiCad has a tool for generating barcodes and adding them to the board. Barcodes can be placed on any layer. Five types of barcodes are supported, including QR codes.

To add a barcode, click the  button on the right toolbar, then click in the canvas in the desired location. The Barcode Properties dialog appears, where you can enter the details for the barcode. The right side of the dialog displays a preview of the barcode based on the selected options.

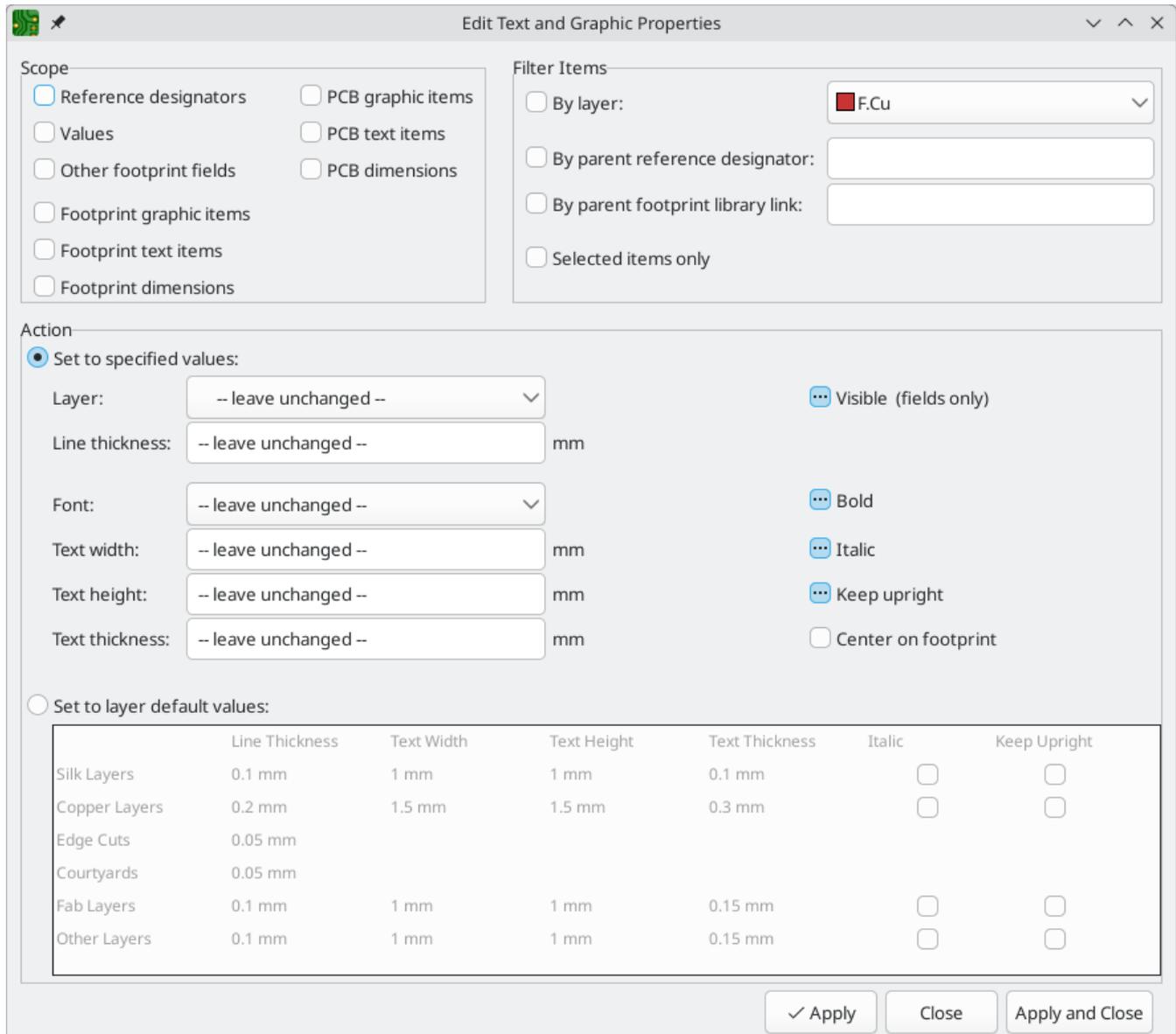


- **Text** is the data to be encoded in the barcode. If the **Show text** checkbox is checked, the text will also be printed verbatim below the barcode.
- If **Locked** is checked, the barcode is created as a [locked object](#).
- If **Knockout** is checked, a solid margin is added around the barcode, and the barcode is created as a negative cutout from the solid background. **Min margin X** and **Min margin Y** control the horizontal and vertical margin around the barcode when knockout mode is selected.
- **Layer** selects the board layer on which to place the barcode.
- **Position X** and **Position Y** determine the location of the barcode.
- **Size X** and **Size Y** determine the width and height of the barcode.
- **Orientation** sets the rotation angle of the barcode in degrees.
- **Text size** controls the size of the text printed below the barcode.
- The **Code** options control the type of barcode that is generated. The following barcode types are available:
  - Code 39 (ISO 16388)
  - Code 128 (ISO 15417)
  - Data Matrix (ECC 200)
  - QR Code (ISO 18004)
  - Micro QR Code

The **Error Correction** options control the error correction included in the generated barcodes. The error correction options only apply to QR codes and Micro QR codes.

## Bulk editing text and graphics

Properties of text and graphics, including footprint fields and dimensions, can be edited in bulk using the **Edit Text and Graphics Properties** dialog (**Edit** → **Edit Text & Graphic Properties...**).



## Scope and Filters

**Scope** settings restrict the tool to editing only certain types of objects. If no scopes are selected, nothing will be edited.

**Filters** restrict the tool to editing particular objects in the selected scope. Objects will only be modified if they match all enabled and relevant filters (some filters do not apply to certain types of objects. For example, parent footprint filters do not apply to graphic items and are ignored for the purpose of changing graphic properties). If no filters are enabled, all objects in the selected scope will be modified. For filters with a text box, wildcards are supported: **\*** matches any characters, and **?** matches any single character.

- **By layer** filters to items on the specified board layer.

By **parent reference designator** filters to fields in the footprint with the specified reference designator.

- **By parent footprint library link** filters to fields in footprint with the specified library link (library and footprint name).
- **Selected items only** filters to the current selection.

## Action

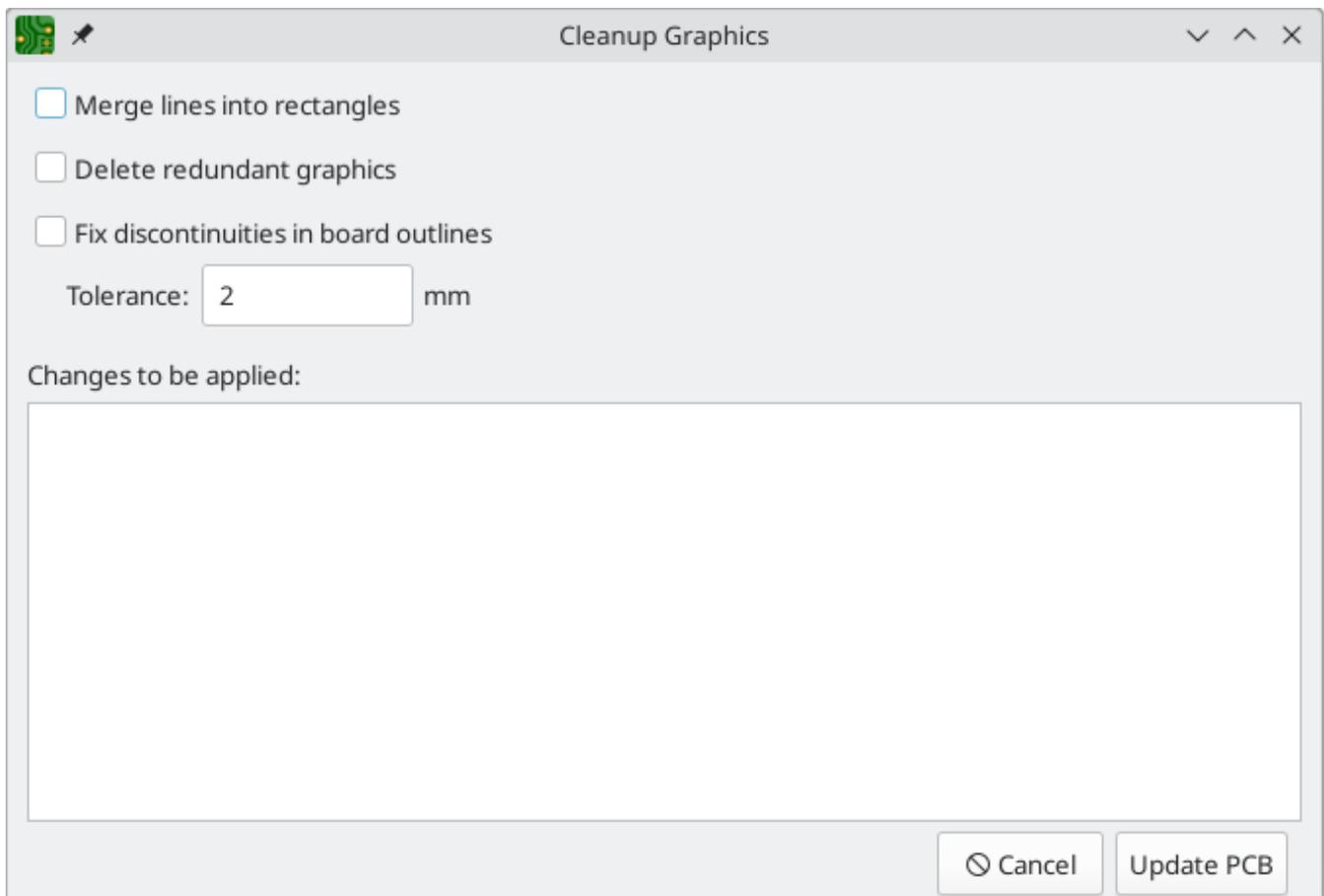
Properties for filtered objects can be set to new values in the bottom part of the dialog. Properties can be set to arbitrary values by selecting **set to specified values** or reset to their layer's default value by selecting **set to layer default values**.

Drop-down lists and text boxes can be set to `-- leave unchanged --` to preserve existing values. Checkboxes can be checked or unchecked to enable or disable a change, but can also be toggled to a third "leave unchanged" state.

- All items can have their **layer** set.
- Graphic items can have their **line thickness** modified.
- Text properties that can be modified are **font**, **text width**, **text height**, **text thickness** (KiCad font only), emphasis (**bold** and **italic**), orientation (**keep upright**), and alignment (**center on footprint**). Footprint fields can also have their **visibility** set.

## Cleaning up graphics

There is a dedicated tool for performing common cleanup operations on graphics, which is run via **Tools** → **Cleanup Graphics...**



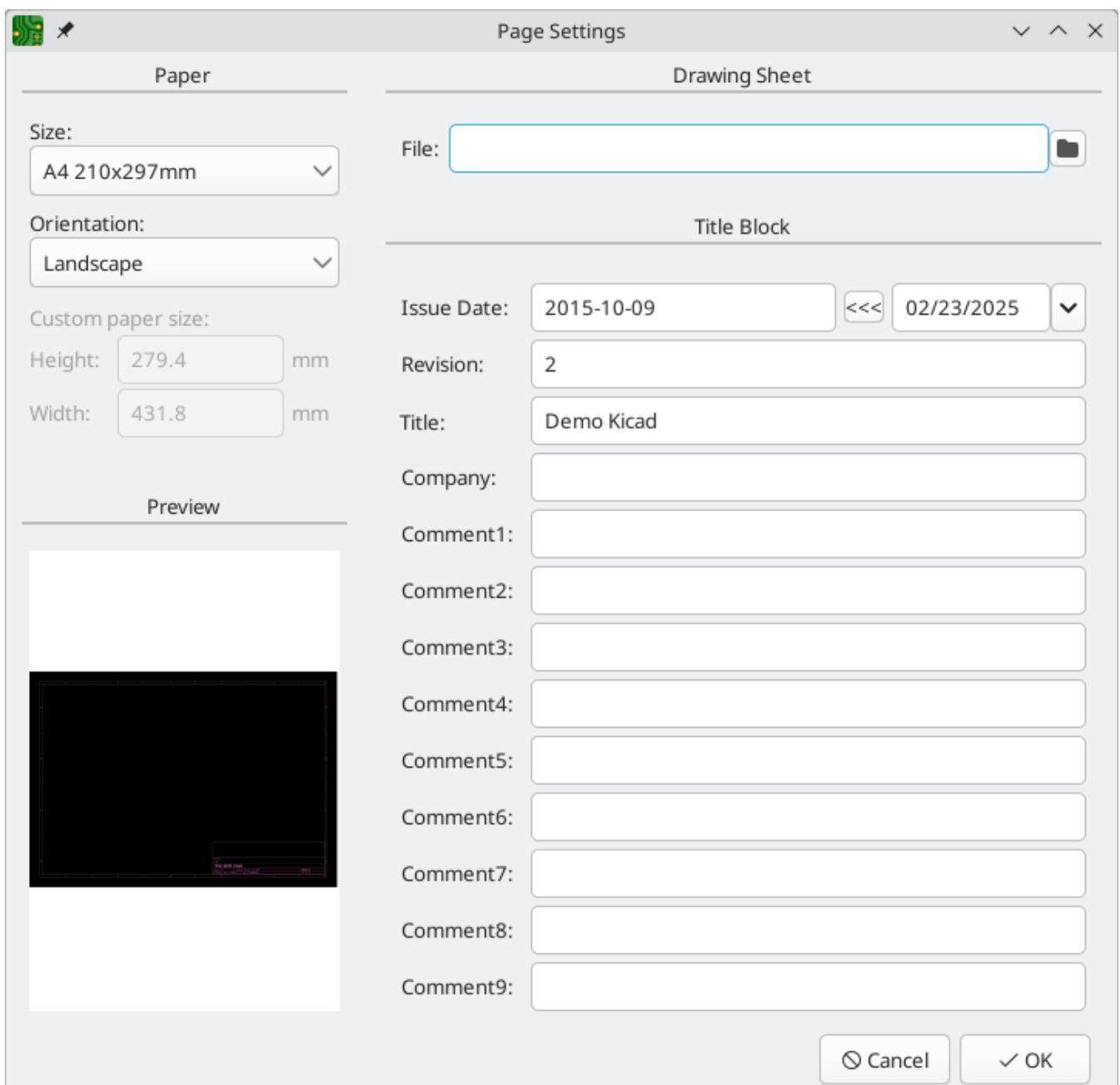
The following cleanup actions are available and will be performed when selected:

- **Merge lines into rectangles:** combines individual graphic lines that together form a rectangle into a single rectangle shape object.
- **Delete redundant graphics:** deletes graphics objects that are duplicated or degenerate.
- **Fix discontinuities in board outlines:** modifies the existing board outline to fix any discontinuities that are within the specified tolerance.

Any changes that will be applied to the board are displayed at the bottom of the dialog. They are not applied until you press the **Update PCB** button.

## Sheet title block

The drawing sheet's title block is edited with the Page Settings tool () . You can also open this tool by double clicking on any part of the drawing sheet.



**Page Settings**

**Paper**

Size: A4 210x297mm

Orientation: Landscape

Custom paper size:

Height: 279.4 mm

Width: 431.8 mm

**Preview**

**Drawing Sheet**

File:

**Title Block**

Issue Date: 2015-10-09 <<< 02/23/2025

Revision: 2

Title: Demo Kicad

Company:

Comment1:

Comment2:

Comment3:

Comment4:

Comment5:

Comment6:

Comment7:

Comment8:

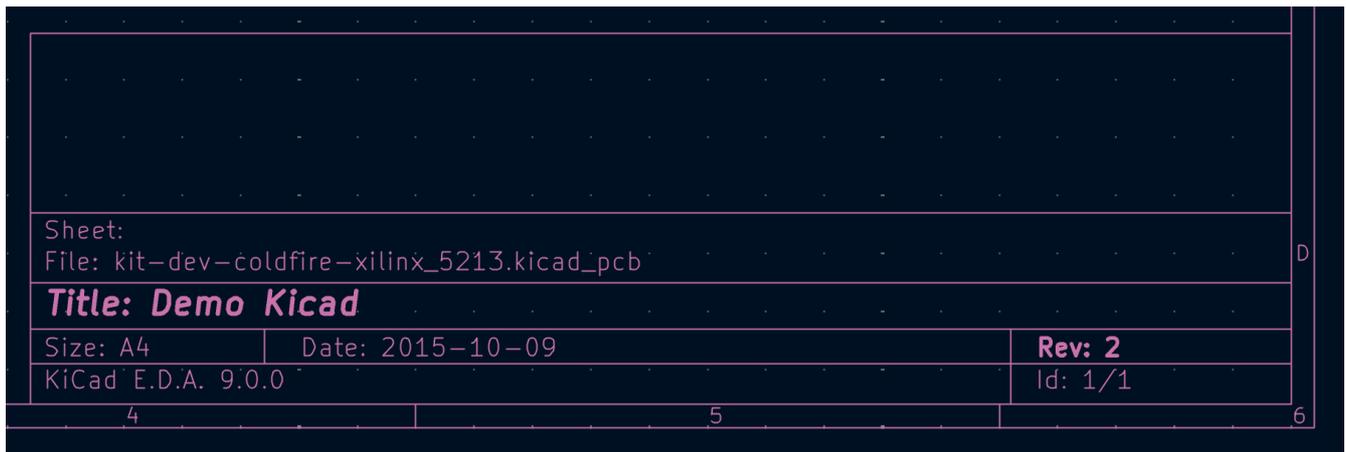
Comment9:

Cancel OK

Each field in the title block can be edited, as well as the paper size and orientation.

You can set the date to today's or any other date by pressing the left arrow button next to **Issue Date**. Note that the date listed in the board title block is not automatically updated. It is only updated when changed in this dialog.

A drawing sheet file can also be selected to replace the default drawing sheet. When choosing a drawing sheet, you can enable the **Embed File** checkbox in the file browser to embed the drawing sheet in the board instead of referencing an external file. This means the board will appear the same when it is opened on another computer that does not have the drawing sheet file available at the same external file path. For more information, see the [embedded files documentation](#).



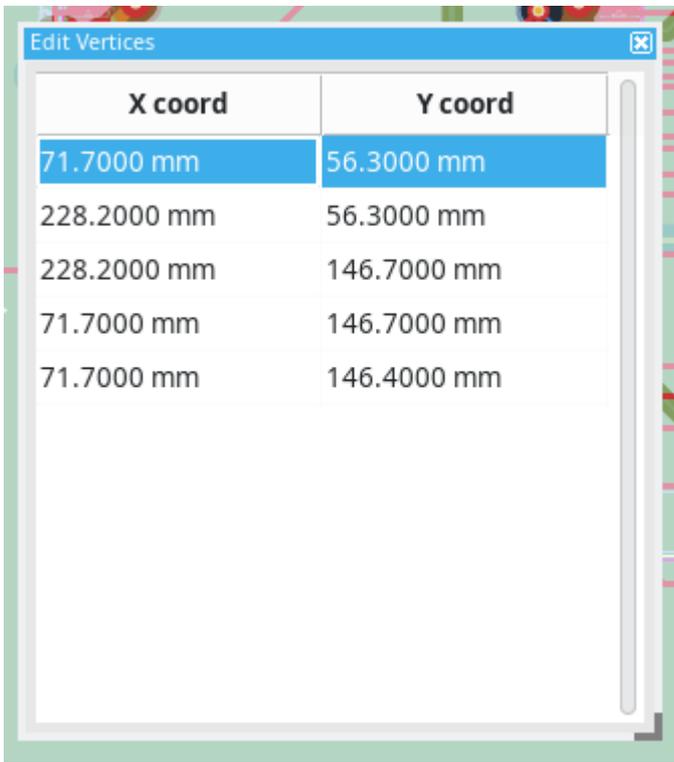
## Rule areas (keepouts)

Rule areas, also known as keepouts, are board regions that can have specific DRC rules defined for them. Some basic rules are available that will raise DRC errors if certain types of objects are within the bounds of the rule area, but rule areas can also be used together with [custom DRC rules](#) to define complex DRC behavior that only applies within the rule area. Rule areas are also used to define channels for [multichannel layout](#).

You can add a rule area by clicking the  button on the right toolbar (**Ctrl** + **Shift** + **K**). Click on the canvas to place the first corner, which will show the Rule Area Properties dialog. After configuring the rule area appropriately, press **OK** to continue placing corners of the rule area. The rule area shape can be an arbitrary polygon; click on the starting corner or double click to finish placing the rule area.

To modify an existing rule area outline, select it, then drag its editing handles to change the shape. Moving a handle at the corner of a rule area will move that corner. Moving a handle on the edge of a rule area will move that edge in a direction perpendicular to the edge. Normally, dragging an edge maintains the angles of the corners adjacent to the edge while allowing the edge's length to vary. Holding **Ctrl** instead holds the edge's length constant and allows the adjacent corner angles to vary.

To precisely position a corner, right click the corner's handle and choose **Shape Modification** → **Move Corner To...**, then enter new X and Y coordinates for the corner. You can also edit the coordinates of every outline corner by right clicking the rule area and choosing **Shape Modification** → **Edit Corners...** This opens a floating dialog with a table containing the coordinates of every corner. Editing the coordinates of a corner immediately updates the rule area.

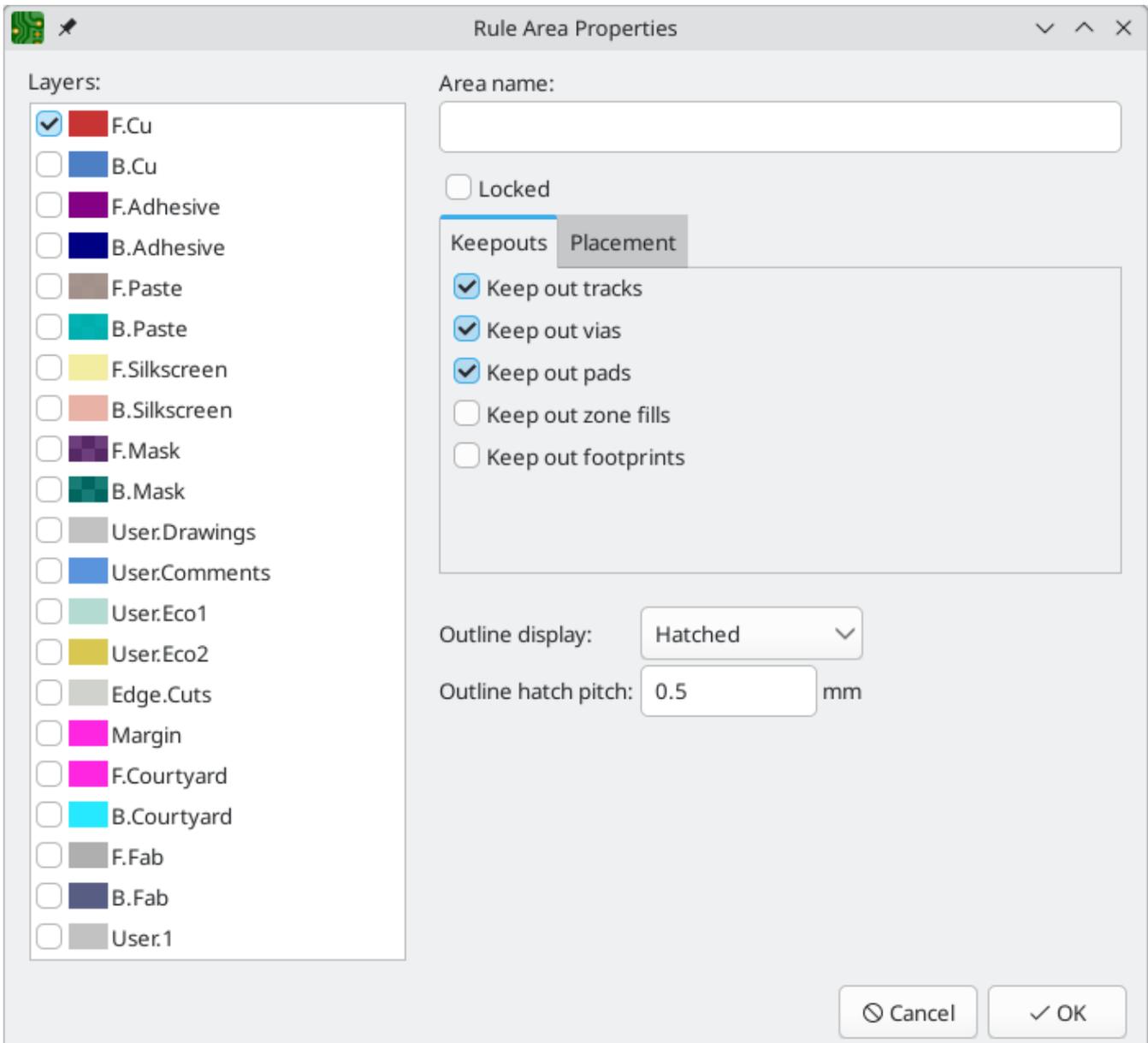


The image shows a software dialog box titled "Edit Vertices" with a close button in the top right corner. It contains a table with two columns: "X coord" and "Y coord". The first row is highlighted in blue. The table lists the following coordinates:

X coord	Y coord
71.7000 mm	56.3000 mm
228.2000 mm	56.3000 mm
228.2000 mm	146.7000 mm
71.7000 mm	146.7000 mm
71.7000 mm	146.4000 mm

**NOTE**

You can also create rule areas by [converting an existing graphic shape to a rule area](#). This can be useful, for example, for creating a rule area with a shape that would otherwise be difficult to draw with the rule area tool, such as a circle. To convert a shape to a rule area, right click the shape, then select **Create from Selection** → **Create Rule Area from Selection....**



The Rule Area Properties dialog has the following options:

- The **layers** list determines which layers the rule area applies to. The area only appears on these layers and the selected keepout rules only apply on these layers. At least one layer must be selected. By default, the active layer in the editing canvas is preselected in the rule area layer list.
- The **area name** field is optional and provides an identifier for the rule area. If it is provided, it is included in DRC violation messages to make them clearer. It can also be used in custom DRC rules to identify a particular rule area.
- The **locked** checkbox determines if the rule area should be [locked](#). As with other objects, rule areas can also be locked or unlocked after they are created.
- The **Keepouts** tab contains several basic rules to prevent various types of objects from being placed within the rule area. The basic rules can be configured to keep out tracks, vias, pads, zone fills, and/or footprints. If an object of one of the selected types is within the rule area, a DRC error will be raised. Additionally, zone fills will automatically avoid a rule area if the rule area is configured to keep out zones.

**NOTE**

Even with no basic rules selected, rule areas can still be used to define specific areas in which to apply [custom DRC rules](#).

- The **Placement** tab contains settings for [multichannel layout](#), which are explained in that section.
- There are a few options for the **outline display** of the rule area. The area can be shown with a hatched outline, fully hatched throughout the area, or with just the outline with no hatching. The **outline hatch pitch** is also adjustable.

## Locking

Locking an item makes it more difficult to select, move, or modify the object, which can prevent unintended modifications. Most objects can be locked through their properties dialogs, by using the right-click context menu, or by using the Toggle Lock hotkey (.

Some actions are restricted for locked objects:

- Locked objects cannot be selected unless the **Locked items** checkbox is enabled in the selection filter.
- Even when selected, locked objects cannot be moved unless the **Override locks** checkbox is enabled in the top toolbar. When it is enabled, locked items can be moved in the same way as unlocked items. When it is disabled, locked items cannot be moved. If you attempt to move a selection that includes locked items, the locked items will be deselected and not moved. Any remaining unlocked items are moved as normal.

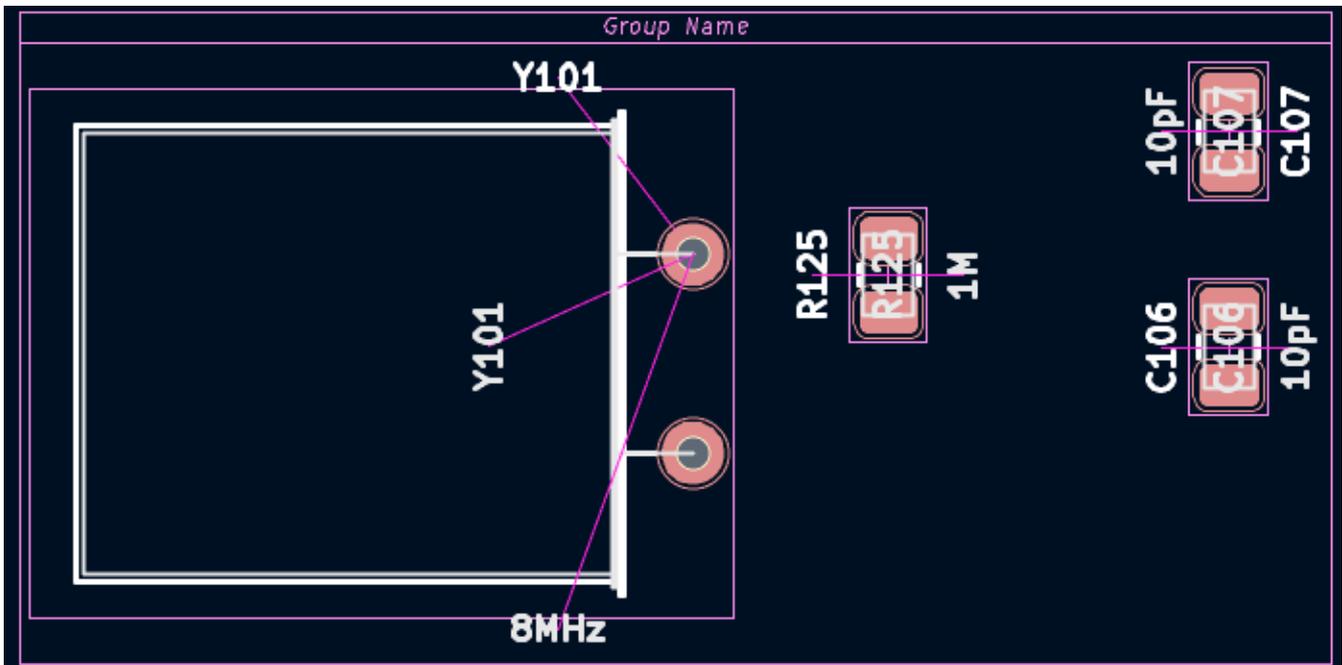
Locked items are displayed with a colored shadow around them. The color can be customized in your color scheme, and the shadow can be hidden in the **Objects** tab of the [Appearance panel](#).

**NOTE**

Locked objects can't be selected unless the **Locked items** checkbox is enabled in the [selection filter](#), or moved unless the **Override locks** checkbox is enabled. By default, these checkboxes are disabled.

## Groups

Groups let you treat multiple objects as a single object for the purposes of moving or rotating them. Each object in the group will maintain its position relative to the other objects in the group. When objects are grouped, it is difficult to accidentally edit them or move them relative to the other members of the group. Groups can have a name, which is displayed in the editing canvas when the group is selected.



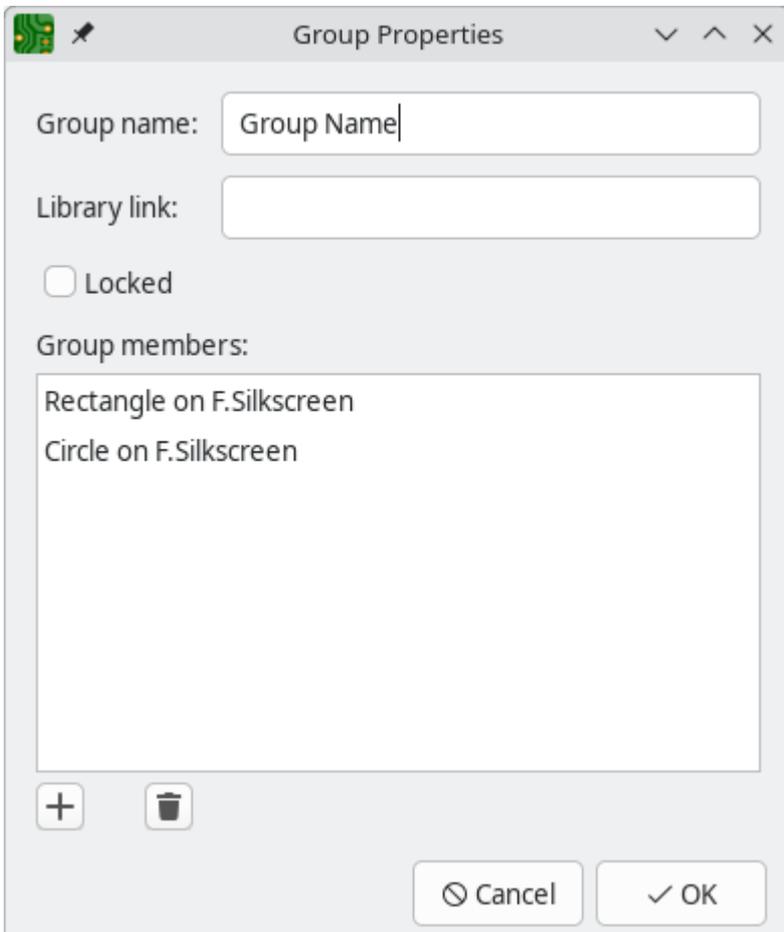
Most types of objects in the Board Editor can be grouped: footprints, tracks, zones, graphic items, and even other groups. Groups can contain multiple different types of objects at once.

To add objects to a group, select them, then right click and choose **Grouping** → **Group Items**, or click the  button in the top toolbar. To remove all items from a group, select the group, right click, and choose **Grouping** → **Ungroup Items**, or click the  button in the top toolbar.

Once objects have been added to a group, selecting any of the objects will select the group as a whole instead of the constituent objects. To edit a specific object within a group, first select the group, the right click and choose **Enter Group**. Double clicking on a group also enters the group. When a group has been entered, objects within the group can be selected and edited individually without affecting the other objects in the group. To leave the group and stop editing its members individually, right click and select **Leave Group**, select an object outside the group, or use .

There are several ways to modify which objects belong to a group. To remove objects from an existing group, enter the group, then select the objects you want to remove, right click, and choose **Grouping** → **Remove Items**. To add items to a group, first ungroup all the items from the group. This will leave the group's former members selected. Then add the new item to the selection and group the selection. Note that without first ungrouping, this process would create a nested group: a new group containing the new item and the entire original group, not just the items in the original group.

You can also add or remove objects from a group in the group's properties dialog. To open a group's properties dialog, press  or right click and click **Properties...** The properties dialog lists the objects contained in the group. To add an additional object to the group, click the  button, then click on the desired object in the editing canvas. The object you click on will be added to the group. To remove an object, select it in the list, then click the  button.



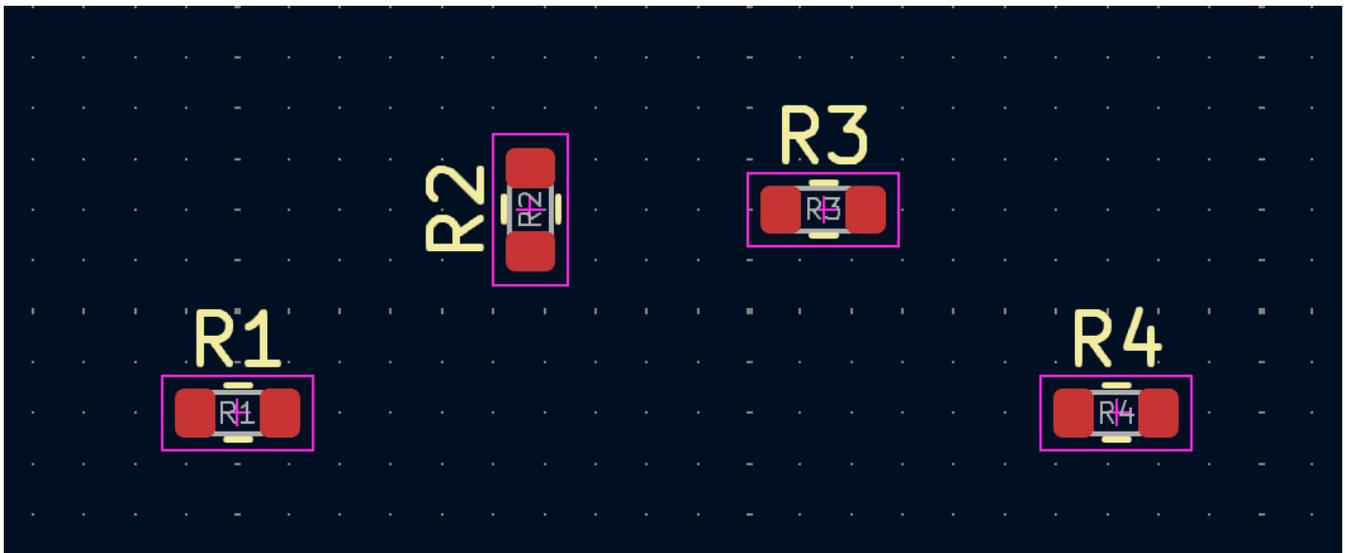
The group properties dialog also lets you specify a name for the group or [lock](#) the group. Groups can also be named or locked using the [Properties Manager](#).

The **Library link** field is used for [design blocks](#). It specifies the group's linked design block name and library in the format `<library>:<block>`. This field must be filled out correctly to link design block layout groups to the corresponding block in the schematic and in the design block library.

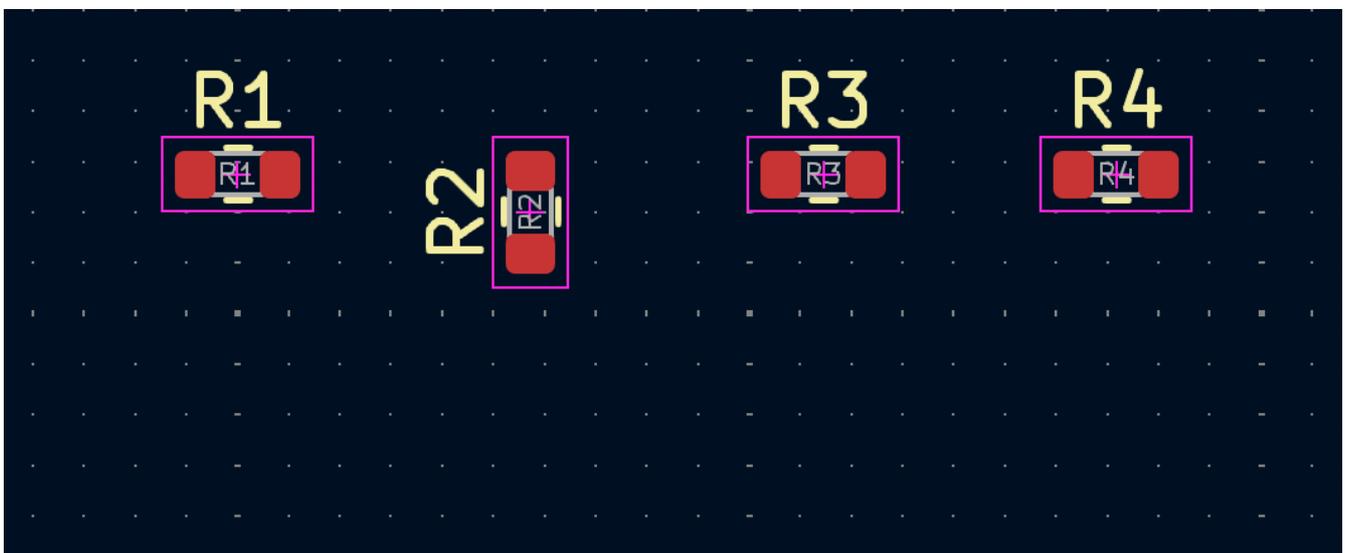
## Aligning objects

The align tool moves a selection of objects so that they are all aligned with a reference object. There are six different alignments to choose from, depending on which part of the objects you wish to align. Objects can be horizontally aligned by their left, center, or right edges, or they can be vertically aligned by their top, center, or bottom edges. Objects are only moved in one dimension, so objects stay in the same horizontal position when aligned vertically, and vice versa. To align objects by a given edge, select the objects, then right click and choose **Align/Distribute** → **Align to Left** (or another alignment as desired).

If the cursor is over an object in the selection, that object is used as the reference object. Otherwise, the reference object is the object in the selection which is located furthest in the alignment direction, for example the leftmost object when aligning by left edge, or the topmost object when aligning by top edge. The topmost object is used when aligning by vertical center, and the leftmost when aligning by horizontal center.



*Before alignment*



*After alignment*

In the example above, R1-R4 are vertically aligned by their top edges, with R2 as the reference object. The first image shows them before alignment and the second image shows them after alignment. In this case, R2 is the topmost object before alignment, so it is chosen as the reference object if the cursor is not over another resistor. After alignment, the top edges of the resistors are at the same position, but the horizontal positions of the resistors are unchanged.

## Distributing objects

You can use the distribute tool to move objects so they are evenly spaced from each other (right click a selection → **Align/Distribute** → **Distribute Horizontally** or **Distribute Vertically**). The two outermost objects in the selection are not moved. This means the top and bottom objects when distributing vertically, and the leftmost and rightmost objects when distributing horizontally. The remaining objects in the selection are evenly distributed between the outermost objects and maintain their relative ordering. Objects are only moved in one dimension, so objects stay in the same horizontal position when distributed vertically, and vice versa.



*Before distribution*



*After distribution*

In the example above, R1-R4 are horizontally distributed. The first image shows them before distribution and the second image shows them after distribution. R1 and R4 are the leftmost and rightmost objects, so they are not moved. R2 and R3 are moved so the horizontal spacing between resistors is equal, but the vertical positions remain unchanged. From left to right, R1-R4 are in the same order that they were in before distribution.

## Arrays

KiCad has an array tool to create rectangular or circular arrays of objects (footprints, vias, graphical objects, etc.). Two types of array are possible: **Grid** and **Circular**.

☰
Create Array
✕

**Grid Array**    Circular Array

---

**Grid Array Size**

Horizontal count:

Vertical count:

**Items Spacing**

Horizontal spacing:  mm

Vertical spacing:  mm

Horizontal offset:  mm

Vertical offset:  mm

**Stagger Settings**

Stagger:

Rows

Columns

**Grid Position**

Source items remain in place

Centre on source items

**Footprint Annotation**

Keep existing reference designators

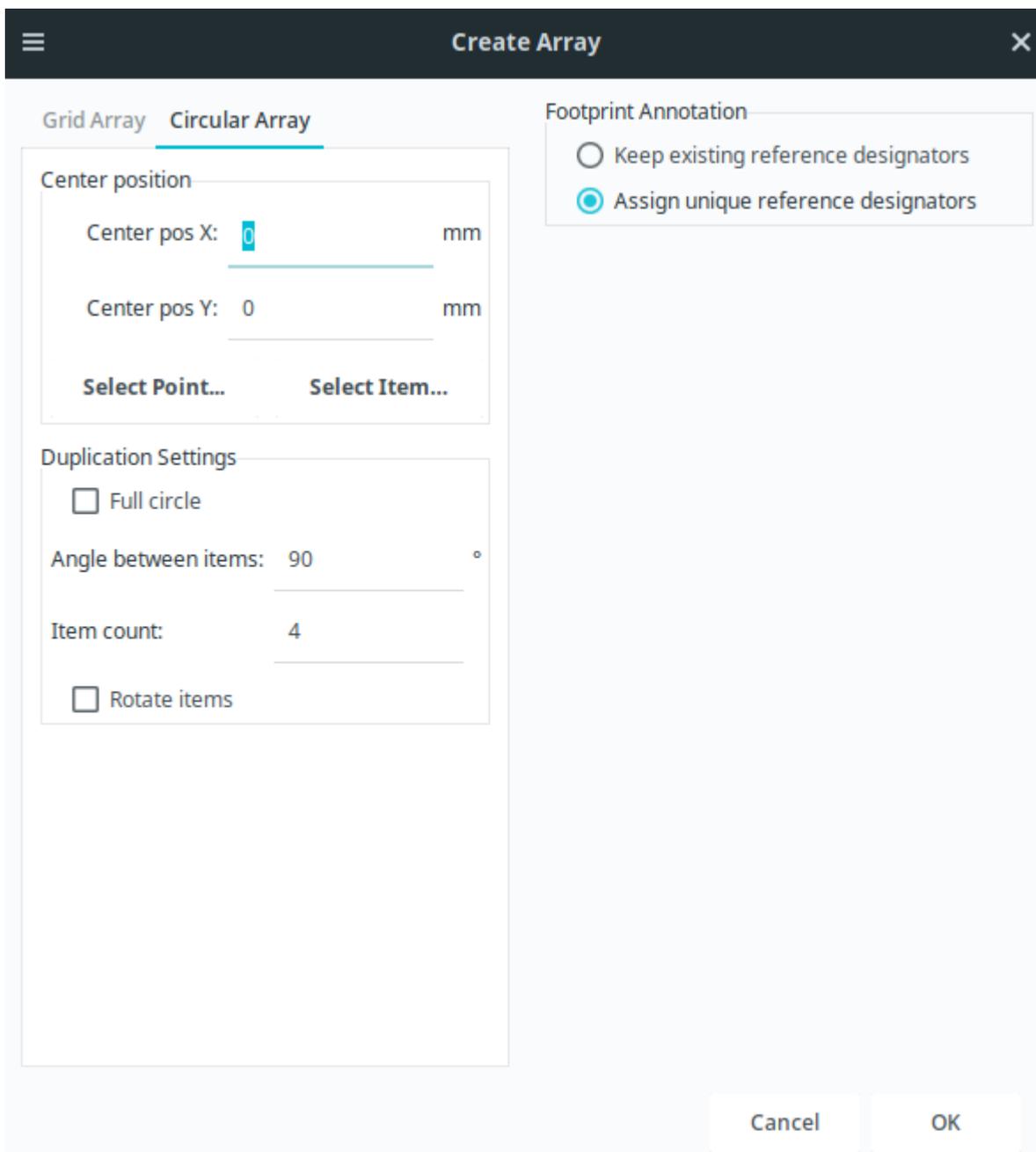
Assign unique reference designators

Cancel
OK

**Grid Arrays** are rectangular and are described by a **horizontal count** and a **vertical count**, which set the number of columns and rows in the array, respectively. The **horizontal** and **vertical spacing** settings describe the distance between columns and rows, while the **horizontal** and **vertical offset** settings describe a shift applied to each row/column compared to the previous row/column.

You can create a repeating staggered pattern by choosing a **stagger** setting, which controls the number of rows or columns that are offset before the pattern repeats. You can stagger by **row** or by **column**. For example, if two staggered rows are selected, each row will be horizontally offset from the previous row by half of the array's horizontal spacing setting. Every other row will be placed at the original spacing and offset. If three staggered columns are selected, each column will be vertically offset by a third of the array's vertical spacing setting. Every third column will be placed at the original spacing and offset. Offsets from the stagger settings are added to the previous horizontal and vertical offset settings.

If the **grid position** option is set to **source items remain in place**, the original items will not be moved, and the grid extends with those items at one corner. If **center on source items** is chosen, the grid is offset so that the resulting grid is centered where the items used to be.



**Circular Arrays** are described by a center point, an angular spacing, and, optionally, the number of arrayed items. If **set center by position** is selected, the center point of the array will be defined by the absolute X/Y position you enter in **center pos X** and **center pos Y**. You can also interactively select a point from the board using **Select Point...**, or select the origin point of another item using **Select Item...**

The **item count** field determines the number of objects in the array, including the source object. The **angle** field determines the angular spacing between items, with the center point at the center of the array. Positive angles result in a counter-clockwise rotation relative to the center point and the source item, while negative angles result in a clockwise rotation. Select **full circle** to evenly space the entered number of items around the circle.

When **rotate items** is selected, objects will be rotated around their origins as array sweeps around the center point. Otherwise, objects will maintain the same orientation as the source item.

When creating an array of footprints, whether rectangular or circular, the **Footprint Annotation** settings control how the reference designators will be set in the new footprints. This affects the linkage of the new

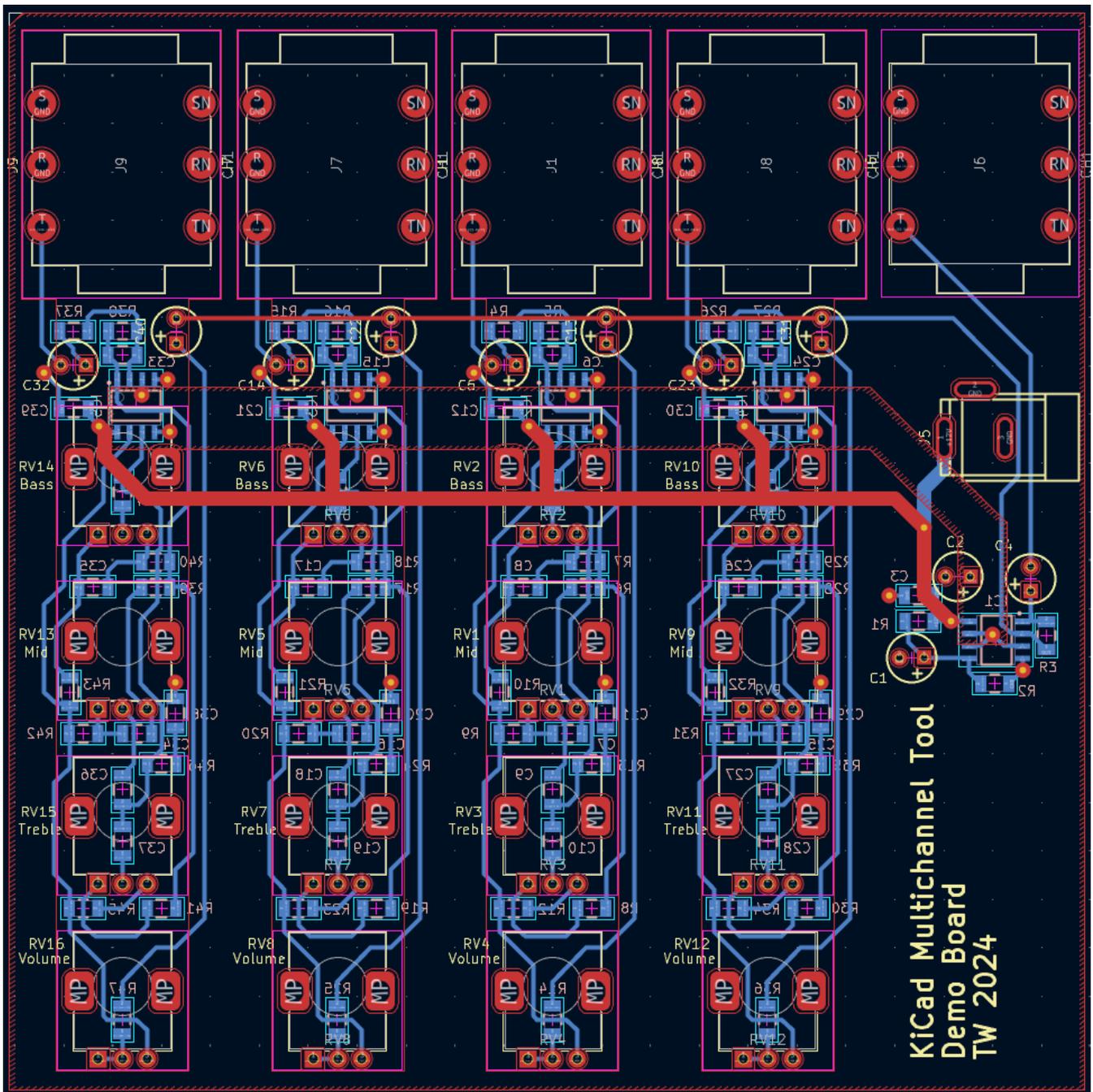
footprints to the schematic. If **keep existing reference designators** is selected, the new footprints in the array will have the same reference designators as the source footprints, resulting in duplicated reference designators in the board. If **assign unique reference designators** is selected, each new footprint created in the array will have a unique reference designator automatically assigned.

**NOTE**

Creating an array of footprints will result in multiple copies of the source footprint(s). If you are using a schematic-based workflow, this will result in footprints that are not represented in the schematic, so careful syncing between the board and the schematic will be needed.

## Multichannel layout

KiCad has a multichannel layout feature for replicating the layout of repetitive blocks of circuitry in a PCB design. This is useful for designs that contain multiple identical subcircuits, such as an audio mixer with many identical channels, a multi-output power supply with repeated regulation stages, or an LED driver board with identical per-channel circuitry. Instead of manually placing and routing each repeated block, you perform the layout for one block and then automatically replicate that placement and routing to all other matching blocks.



The multichannel layout workflow has three main stages:

1. **Schematic setup:** Designate which portions of the schematic represent the repeated parts of the circuit. Each repeated block is called a *channel*.
2. **Rule area generation:** Create placement rule areas on the PCB board to define the physical boundary of each channel.
3. **Layout replication:** Manually place and route one *reference* channel, then use the **Repeat Layout** tool to copy its placement and routing to all *target* channels.

Using the multichannel layout feature first requires you to designate which portions of the schematic represent the repeated parts of the circuit. You can use [hierarchical sheets](#) (with a repeated hierarchical sheet for each channel), [component classes](#) (with a unique component class assigned to the symbols in each channel), or named [groups](#). Each channel will exactly correspond to the symbols (and their associated connections) in a single sheet, component class, or group.

For the layout, specially configured **rule areas** are drawn on the board and used to describe the physical location of each repeated channel. The automatic placement of footprints, routing, and other items is restricted to these placement rule areas. Each "channel" of the design corresponds to a single rule area. One rule area will be the *reference* rule area, which will be manually placed and routed. The other rule areas are the *target* rule areas, which will reuse the placement and routing from the reference rule area.

After setting up the placement rule areas for each channel and manually routing the reference channel, the Repeat Layout tool is used to copy the placement and layout from the reference rule area to the target rule areas.

**NOTE**

KiCad includes a demo project, called `multichannel`, that demonstrates the use of the multichannel layout feature.

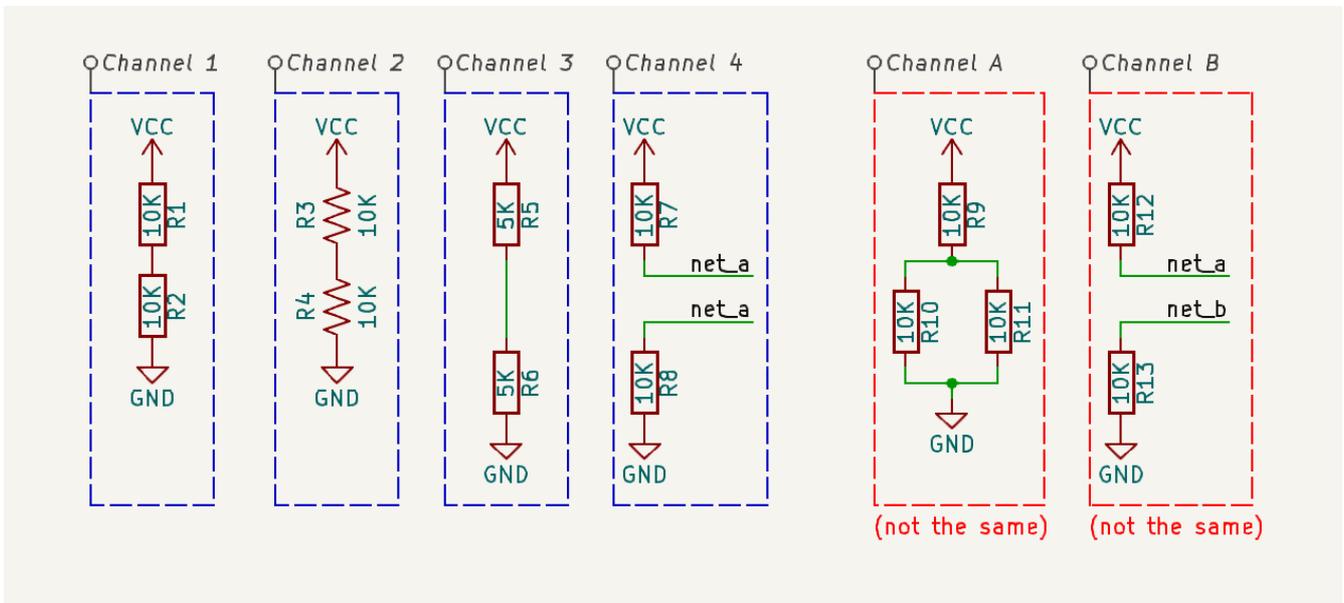
## Multichannel design procedure: schematic

Designing a multichannel layout begins in the schematic. You need to specify which components (symbols) belong to each channel. Each channel in the schematic must be equivalent to the other channels. This means channels must match each other in the following ways:

- each matched channel needs to have the same number of symbols
- corresponding symbols in each channel need to have the same reference designator prefix (e.g. R or U), although the full reference designators need to be unique as usual
- corresponding symbols in each channel need to have the same footprint
- connections between symbols need to be equivalent in each channel

In the example schematic below, Channels 1, 2, 3, and 4 are equivalent and therefore can be used to share routing in a multichannel design. Even though the net connections are drawn differently in some channels, the underlying net connections are the same. The different symbols in Channel 2 and the different values in Channel 3 also do not break the equivalency. Footprint assignments are not shown in the image, but the symbols that correspond between channels must use the same footprints. In this example, this means that R1, R3, R5, and R7 each must use the same footprint, as must R2, R4, R6, and R8.

In contrast, Channels A and B are not equivalent to Channels 1-4, nor to each other. Channel A contains an extra parallel resistor not present in the other channels, and Channel B is missing a connection between the two resistors.



You can assign symbols to channels in three ways:

- **Hierarchical sheets:** using [hierarchical sheets](#), each channel is represented by a different sheet. Normally you will instantiate the same hierarchical sheet file multiple times, with one instantiation per channel. Each sheet instantiation needs to include all of the symbols for the corresponding channel, with no extra symbols.
- **Component classes:** using [component classes](#), each channel is represented by a different component class. Each component class needs to include all of the symbols for the corresponding channel, with no extra symbols. Component classes are shown in the previous schematic example.
- **Groups:** using named [groups](#), each channel is represented by a group of footprints on the board. This provides a more explicit way to define channels when hierarchical sheets or component classes are not suitable.

#### TIP

Hierarchical sheets are the most common and convenient approach for multichannel designs. By instantiating the same sheet file multiple times, you guarantee that each channel has identical circuitry. The schematic editor automatically assigns unique reference designators to each instantiation.

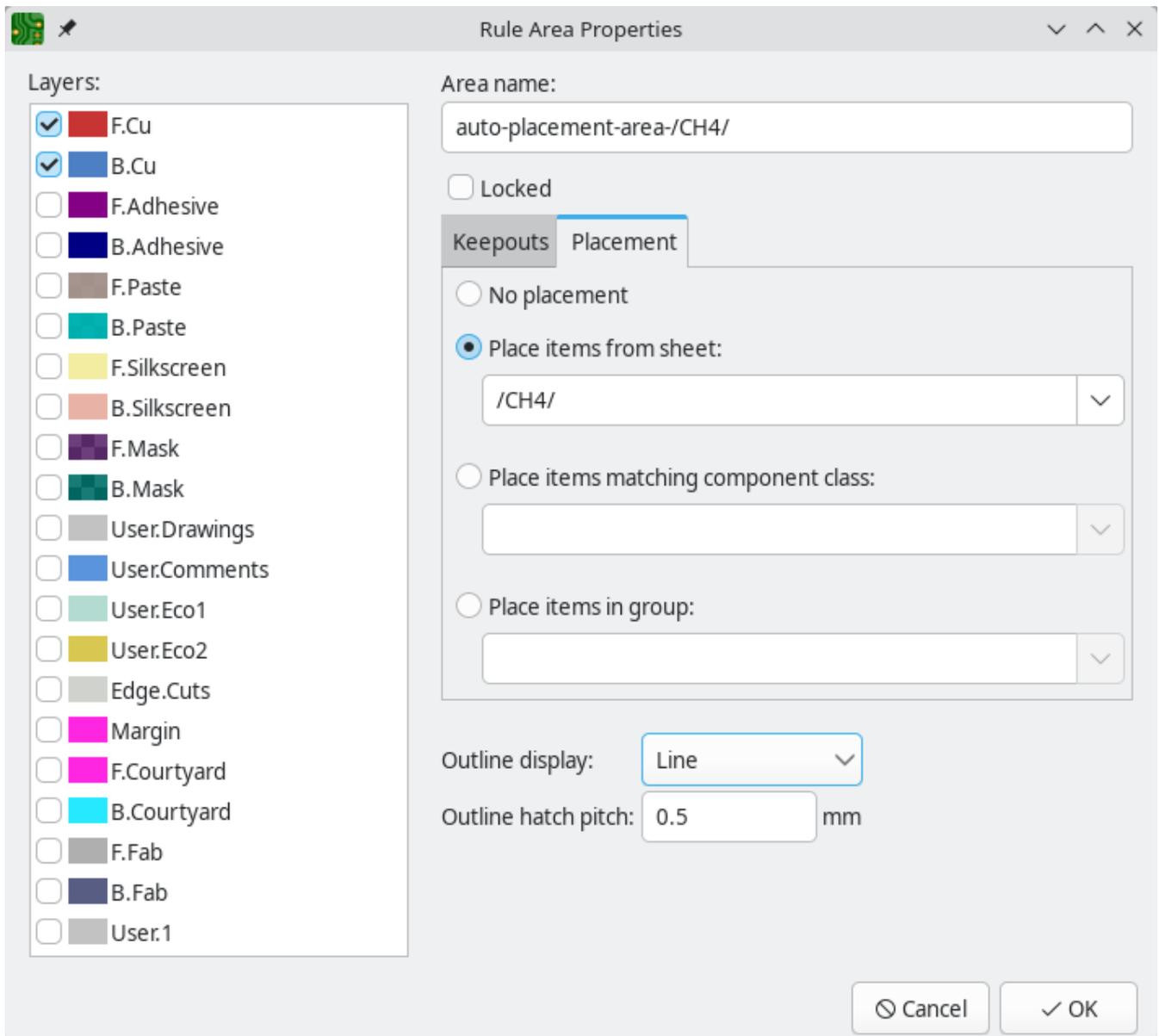
## Multichannel design procedure: board

In the board, each channel is represented with a [rule area](#) with its rules configured for placement. You need to add a rule area for each channel. One rule area is the *reference*, containing manually placed and routed footprints. The other rule areas are *targets* and have the reference placement and routing copied to them. Placement rule areas, whether for reference or target channels, need the following settings configured in their properties:

- The options in the **Placement** tab should be configured to select the hierarchical sheet, component class, or named [group](#) that contains the channel's components.
- The options in the **Keepouts** tab should typically be unselected, unless there is a specific type of item that needs to be excluded from the rule area.
- The rule area's **layers** should be set based on which layers are considered part of the channel. An item will be copied from the reference channel to a target channel only if the item's layer is enabled in both

the reference and target rule areas.

- The **area name** is used to label the rule area when it is listed in the Repeat Layout dialog. While not strictly required, it will be hard for you to distinguish rule areas in the GUI unless each rule area has a unique name.



## Generating placement rule areas

Placement rule areas can be manually created like any [rule area](#), or they can be automatically generated by selecting **Tools** → **Multi-Channel** → **Generate Placement Rule Areas...** This tool scans the design for hierarchical sheets, component classes, and named groups, then creates a placement rule area for each one you select. This is the fastest way to set up placement rule areas for a multichannel design.

### NOTE

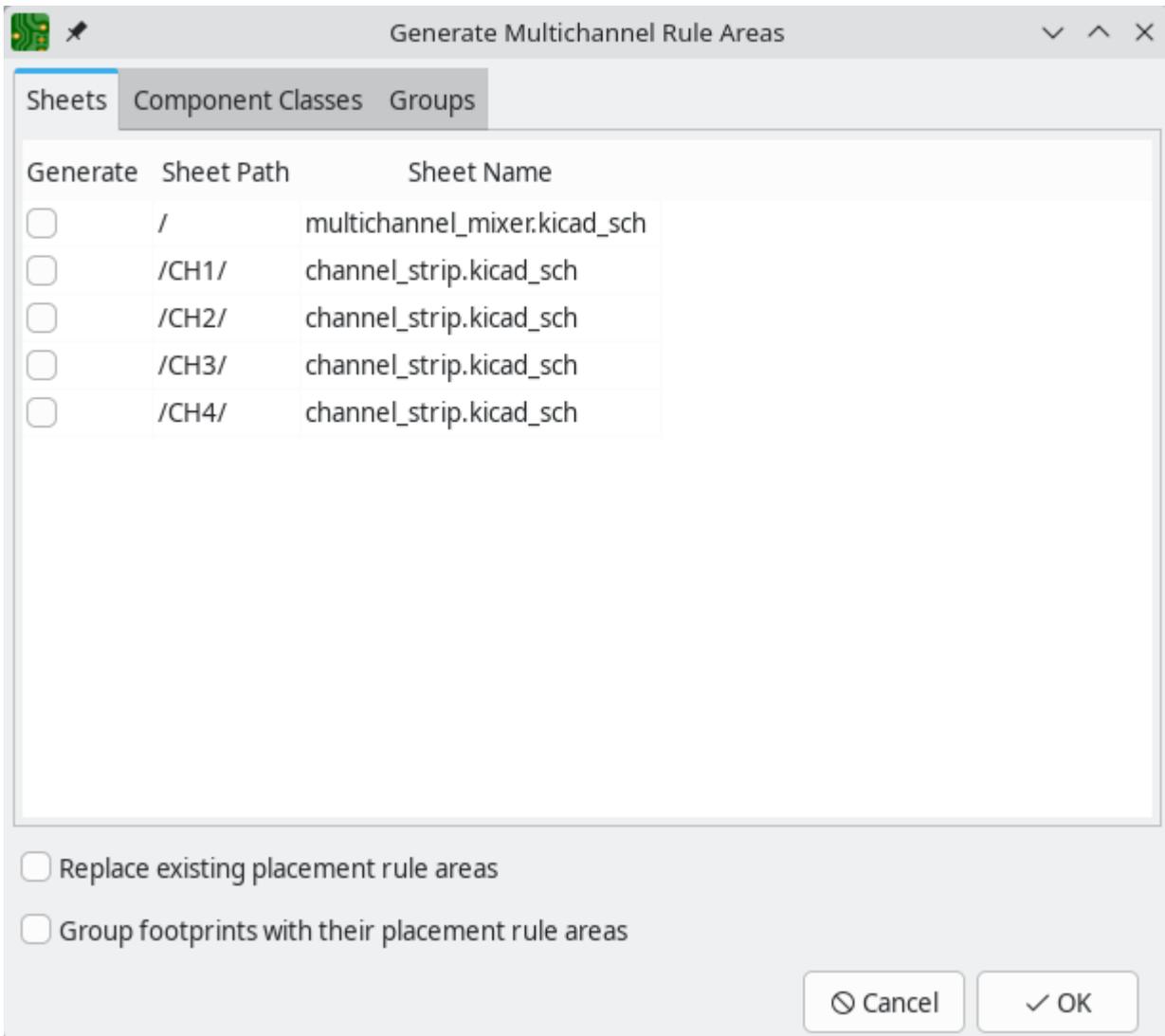
The tool allows you to generate a placement rule area for **any** sheet, component class, or named group, even if it is not intended to represent a channel in a multichannel design. It is your responsibility to select only the desired sheets, component classes, or groups.

The Generate Multichannel Rule Areas dialog contains a tabbed notebook with three tabs:

-

**Sheets:** lists every hierarchical sheet path found on the board. Each row shows the sheet path and the sheet file name. Check the **Generate** checkbox for each sheet that represents a channel.

- **Component Classes:** lists every component class assigned to footprints on the board. Check the **Generate** checkbox for each component class that represents a channel.
- **Groups:** lists every named group on the board. Check the **Generate** checkbox for each group that represents a channel.



Below the tabbed source selection, the dialog has additional options:

Option	Description
<b>Replace existing placement rule areas</b>	If enabled, the newly generated rule area for each channel will replace any rule areas that already exist for that channel. If disabled, channels that already have a placement rule area will be skipped.
<b>Group footprints with their placement rule areas</b>	If enabled, the newly generated rule area for each channel will be added to a <b>group</b> with all footprints associated with that rule area. This allows the target rule area and its associated items to be manipulated as a single entity. This option is disabled for group-based sources, because the footprints already belong to their respective groups.

When you click **OK**, KiCad performs the following steps for each selected channel:

1. Queries the board for all footprints belonging to the sheet, component class, or group.
2. Computes a convex hull around those footprints with a small margin.
3. Rectifies the hull into an axis-aligned polygon using only 90-degree segments.
4. Creates a new rule area configured as a placement area, with the appropriate source type (sheet path, component class, or group name) recorded.
5. Sets the rule area layers to all copper layers.
6. If **Replace existing** is checked and a matching rule area already exists, the old rule area is removed.
7. If **Group footprints** is checked, a new group is created containing the rule area and all its associated footprints.
8. Names the generated rule area automatically using the pattern `auto-placement-area-<source>`, where `<source>` is the sheet path, component class name, or group name.

After generating a rule area, you may want to change the configured layers to ensure all desired items in the reference channel get copied to the target channels.

**NOTE** | If generated rule areas are grouped with their footprints, you will need to enter the group (or ungroup the items) in order to edit the rule area.

**TIP** | After generating rule areas, enlarge the reference rule area to enclose all routing for the reference channel. Target rule areas will be resized automatically to match the reference when you run **Repeat Layout**.

## Repeating the layout

Once you have created rule areas for each channel and completed placing footprints and routing for the reference channel, you can use the **Repeat Layout** tool to copy the reference channel's layout to the other channels. The tool is accessed from **Tools** → **Multi-Channel** → **Repeat Layout...**

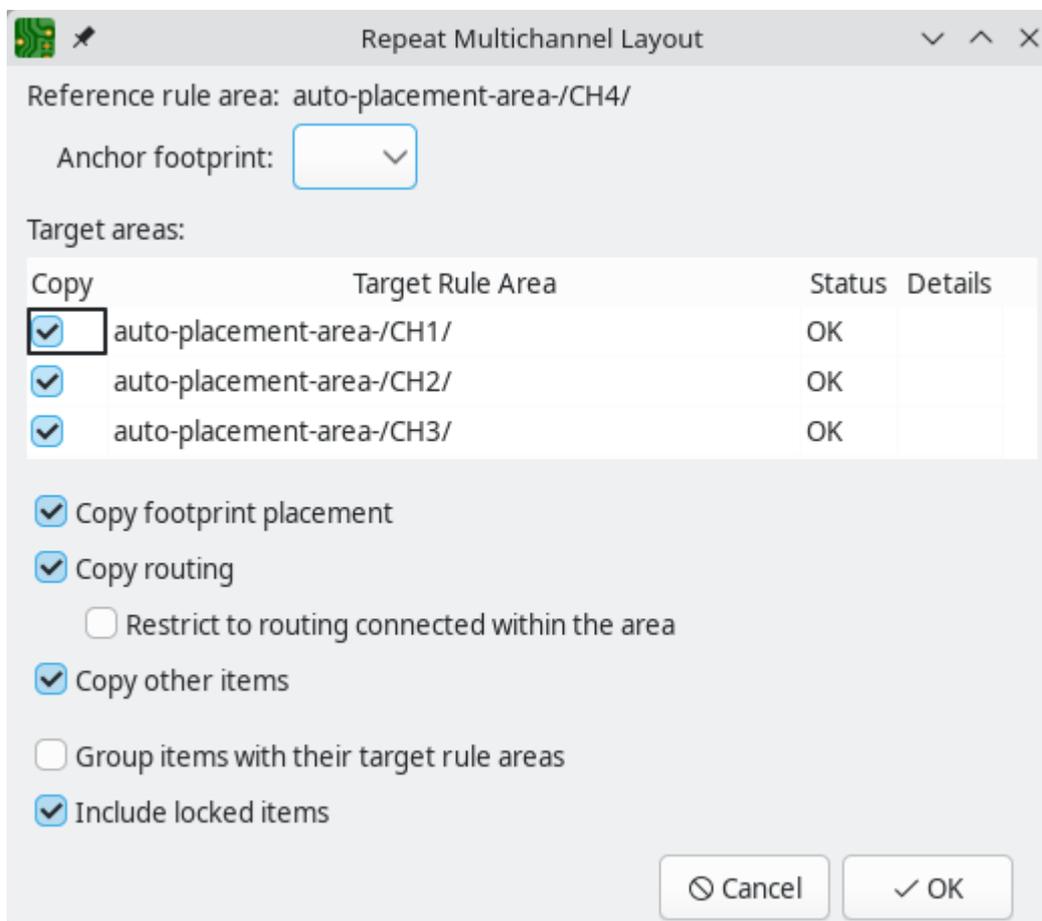
The complete workflow is:

- 1.

**Select the reference rule area.** Either select a placement rule area on the board before running the tool, or run the tool without a selection and you will be prompted to click on the reference rule area. The reference rule area is the one whose placement and routing you want to replicate.

- 2. Topology check.** KiCad automatically finds all other placement rule areas on the board and checks whether their circuit topology matches the reference. This process attempts to find a one-to-one mapping of components and connections (an *isomorphism*) between the reference and each target. A progress indicator is shown for large designs.
- 3. Configure the Repeat Layout dialog.** The dialog displays the results of the topology check and lets you select which targets to update and which items to copy.
- 4. Execute.** Click **OK** to apply the reference layout to all selected target channels.

If a rule area is selected when the tool is run, that rule area will be used as the reference rule area. If no rule area is selected, you will be prompted to select a rule area. You can also select a group that contains a rule area, and the rule area within the group will be used. When the reference rule area is determined, a dialog will appear.



### Target rule area table

The table at the top of the dialog controls which target rule areas will receive the layout from the reference channel. Each row represents one target rule area and shows the following columns:

Column	Description
<b>Copy</b>	A checkbox controlling whether this target will receive the replicated layout. Targets whose topology matches the reference are checked by default. Targets with mismatches are unchecked and cannot be selected.
<b>Rule Area</b>	The name of the target rule area, as configured in its zone properties.
<b>Status</b>	Shows "OK" if the target channel's topology matches the reference, or an error message describing the mismatch. Click the details icon (if present) to see a detailed breakdown of the mismatch reasons.

The tool will only copy items to a target rule area if the target's status is listed as "OK". If the status is not "OK", the target channel's circuit topology cannot be matched to the reference channel; see the requirements for [how channels need to match in the schematic](#) for more information.

### Repeat layout options

There are several options to control which items from the reference channel are copied to the selected target areas, and how those copied items are handled:

Option	Description
<b>Anchor footprint</b>	If a footprint is selected in the dropdown, that footprint (and its equivalent in each channel) acts as an anchor for the channel. Each channel will be positioned and oriented relative to the channel's anchor footprint. If no footprint is selected, each channel will be positioned relative to the center of the reference rule area.
<b>Copy footprint placement</b>	If enabled, the placement of footprints in the reference rule area will be replicated for footprints associated with the target rule area(s). Footprints are copied if they are enclosed by or intersect the reference rule area; they are not copied if they are fully outside. Footprints will only be copied if they are on a layer that is enabled in both the reference and target rule areas.
<b>Copy routing</b>	If enabled, any tracks and vias in the reference rule area will be copied to the target rule area(s). Routing is copied if it is fully enclosed by the reference rule area; it is not copied if it is partially or fully outside. Routing will only be copied if it is on a layer that is enabled in both the reference and target rule areas. Existing routing in the target rule areas is removed before the new routing is placed.
<b>Restrict to routing connected within the area</b>	A sub-option of <b>Copy routing</b> . When checked, tracks and vias will not be copied unless they share a net with a pad inside the reference rule area. This can be useful to prevent copying unrelated tracks that happen to pass through the rule area.
<b>Copy other items</b>	If enabled, any other items (zones, graphic objects) fully enclosed by the reference rule area will be copied to the target rule area(s). Items are copied if they are fully enclosed by the reference rule area; they are not copied if they are partially or fully outside. This means, for example, that a large copper zone that intersects a reference channel will not be copied to the target channels. Items will only be replicated if they are on a layer that is enabled in both the reference and target rule areas.
<b>Group items with their target rule areas</b>	If enabled, the items copied to a target rule area will be added to a <a href="#">group</a> with that rule area. This allows the target rule area and its associated items to be manipulated as a single entity.
<b>Include locked items</b>	If enabled, items in the reference rule area will be copied even if they are <a href="#">locked</a> , and items associated with target rule areas will be updated even if they are locked. If not enabled, locked items in the reference rule area will not be copied and locked items in the target rule area will not be updated or deleted to match the reference rule area.

After clicking **OK**, the layout from the reference channel will be applied to the target channels. An info bar message confirms how many rule areas were updated. When the repeat layout is completed, each channel can be individually edited like any other part of the board design.

**TIP**

You can run **Repeat Layout** multiple times as you refine your reference channel. Each run replaces the target channel contents with the current state of the reference channel.

## Topology matching

The core of the multichannel layout feature is the topology matching algorithm, which determines how components in the reference channel correspond to components in each target channel. Understanding how this matching works is important for diagnosing issues when channels fail to match.

### How components are matched

KiCad analyzes the connectivity of both the reference and target channels. For each channel it catalogs:

- **Components:** one entry per footprint, identified by its reference designator prefix (the letters before the numeric suffix, e.g. R from R1, U from U3).
- **Pins:** one entry per pad on each footprint.
- **Connections:** which pins share the same net.

Two components are considered the "same kind" if they share the same reference designator prefix. For hierarchical designs where references may include a channel identifier (such as TRIM\_1.1 and TRIM\_2.1), the algorithm detects a common prefix and treats the trailing channel-specific suffix as a channel identifier, so the components will still be matched.

The algorithm then searches for a one-to-one mapping (an *isomorphism*) of components in the reference to components in the target such that all pin connections are preserved. This means:

- For every pair of connected pads in the reference channel, the corresponding pads in the target channel must also be connected.
- The matching is based on actual net connectivity, not on reference designator numbers. R1 in the reference does not necessarily map to R1 in the target — it maps to whichever resistor has the same connectivity pattern.

### Tie-breaking for identical components

When multiple components in a target channel have identical connectivity (for example, a row of identical decoupling capacitors all connected between the same power and ground nets), the algorithm uses tie-breaking strategies to select the best match:

1. **Symbol/sheet instance matching:** if the reference and target components originated from the same hierarchical sheet instance, they are preferentially matched. This is the most reliable tie-breaker for hierarchical sheet designs.
2. **Position-based proximity:** components closer in physical position to their reference counterparts may be preferred.

### Mismatch error details

When a target channel fails the topology check, the **Status** column in the Repeat Layout dialog shows an error summary. You can click the details icon in the rightmost column to view a detailed error dialog listing:

- Which specific components could not be matched and why.

The total component count in the reference vs. the target area, if they differ.

- A list of all components in each area for comparison.

See [Troubleshooting multichannel layouts](#) for common reasons that topology matching fails and how to resolve them.

## Routing duplication details

When **Copy routing** is enabled, the Repeat Layout tool duplicates all tracks and vias from the reference rule area to each target rule area. Understanding how this process works will help you achieve clean results.

### What is copied

The tool identifies routing in the reference rule area by testing whether each track or via on the board is *enclosed by* the reference rule area polygon. Items that are fully enclosed are candidates for copying. Items that only partially overlap or are fully outside the rule area are not copied.

For each copied item, the tool:

1. Duplicates the item (track segment or via).
2. Translates the duplicated item from the reference position to the target position, based on the offset between the rule area centers (or anchor footprints, if an anchor is set).
3. Rotates the duplicated item to match any orientation difference between the anchor footprints.
4. Reassigns the net of the duplicated item to the corresponding net in the target channel, using the component matching to look up which target pad connects to the same net.

### Net reassignment

When a track or via is duplicated, it must be assigned to the correct net in the target channel. The tool does this by:

1. Finding which pads in the reference channel are connected to the same net as the original routing item.
2. Looking up those pads' parent footprints.
3. Finding the corresponding pads on the matched target footprints.
4. Assigning the target pad's net to the duplicated routing item.

This means routing is automatically reassigned to the correct target-channel nets without any manual intervention.

### Existing target routing

Before copying routing to a target area, the tool removes any existing routing within the target rule area that is on layers enabled in the target rule area. This ensures a clean copy without leftover routing from a previous iteration. If **Restrict to routing connected within the area** is enabled, only routing connected to pads within the area is removed.

### Layer considerations

A routing item is only copied if its layer is enabled in *both* the reference and target rule areas. This means:

If the reference rule area includes layers `F.Cu` and `B.Cu`, but a target rule area only includes `F.Cu`, then only front-copper routing will be copied to that target.

- Inner copper layers must be explicitly included in both rule areas for inner-layer routing to be copied.

**TIP**

Make sure all copper layers used by your routing are enabled in both the reference and target rule areas. A common issue is forgetting to add inner copper layers to the rule area layer set, which causes inner-layer routing to be silently skipped.

## When routing duplication fails

Routing duplication can produce unexpected results in several situations:

- **Routing extends outside the rule area:** tracks that exit the reference rule area will not be copied. Make sure the reference rule area is large enough to enclose all routing for the channel, including any via fan-outs near the edge.
- **Net reassignment failure:** if a track's net cannot be mapped to a target channel net (for example, because it belongs to a global net that is not part of any channel), the duplicated track may retain the reference channel's net. Enabling **Restrict to routing connected within the area** can help avoid copying such unrelated tracks.
- **Layer mismatch:** if the reference routing uses a layer not enabled in the target rule area, those segments will be silently omitted.

## Other item duplication

When **Copy other items** is enabled, zones and graphic objects (such as drawings, text, and dimensions) within the reference rule area are also duplicated to each target rule area. The same displacement and rotation transforms are applied.

Zones are duplicated with their net reassigned using the same component-match-based lookup used for routing. A zone is only copied if all of its layers are present in both the reference and target rule areas.

Graphic objects that are not connected (such as silkscreen text, board outlines, or fab layer drawings) are copied without net assignment. They must be on a layer enabled in both rule areas.

**NOTE**

Footprint text items (reference designators, values, and other footprint fields) are handled as part of footprint placement, not as standalone items. Their position, visibility, and attributes are updated to match the reference footprint's fields when placement is copied.

## Step-by-step multichannel workflow

The following procedure summarizes the complete multichannel layout workflow from start to finish:

1. **Design the schematic.** Create the repeated circuit as a hierarchical sheet (or assign component classes / groups). Instantiate the sheet once per channel. Ensure all channels have identical circuit topology.
2. **Import the netlist / Update PCB.** Transfer the schematic to the PCB editor. All footprints for all channels will appear on the board.
3. **Generate placement rule areas.** Select **Tools** → **Multi-Channel** → **Generate Placement Rule Areas...** In the dialog, check the sheets (or component classes, or groups) that correspond to your channels. Click

**OK.**

4. **Arrange the rule areas.** Move each generated rule area to its desired position on the board. The footprints associated with each rule area do not need to be inside the rule area yet — that will happen during the repeat layout step.
5. **Design the reference channel.** Choose one rule area as your reference. Place all of its footprints inside the rule area and route the channel completely. Ensure the rule area boundary encloses all footprints, routing, and other items you want to replicate.
6. **Enlarge the reference rule area if needed.** The auto-generated rule area may be too small to enclose routing. Edit the rule area outline to include all routing, including via fan-outs and any local copper zones.
7. **Run Repeat Layout.** Select the reference rule area, then select **Tools** → **Multi-Channel** → **Repeat Layout...** In the dialog:
  - Verify all target channels show **OK** status.
  - Select an anchor footprint if you want channels to be oriented relative to a specific component.
  - Enable **Copy footprint placement**, **Copy routing**, and **Copy other items** as desired.
  - Click **OK**.
8. **Review the results.** Inspect each target channel to verify that placement and routing were copied correctly. Run DRC to check for errors.
9. **Iterate as needed.** If you need to make changes to the reference channel's layout, edit it and run **Repeat Layout** again. Target channels will be updated to match.

## Troubleshooting multichannel layouts

### Channels fail topology matching

If a target channel's status shows an error instead of "OK" in the Repeat Layout dialog, the channel's circuit topology does not match the reference. Click the details icon to see specifics. Common causes:

- **Missing or extra components:** verify that each channel's schematic sheet (or component class) contains exactly the same set of symbols. A common mistake is adding a decoupling capacitor to one channel but not the others.
- **Different reference designator prefixes:** a component in the target channel has a different prefix than any component in the reference (e.g. R vs. C), so no pairing is possible.
- **Different footprint assignments:** corresponding symbols in different channels must use the same footprint. If R1 uses a 0402 footprint but R3 uses a 0603, and they should correspond, the pad counts will differ and matching will fail.
- **Different net connections:** if a net is connected differently in one channel (e.g. a wire was accidentally deleted or connected to the wrong pin), the connectivity will differ.
- **Nested hierarchical sheets:** child sheets within the channel sheet are included when querying components. Ensure that nested sheets are identical across channels.

### Routing does not appear in target channels

**Rule area too small:** the most common cause. Routing outside the reference rule area boundary is not copied. Enlarge the reference rule area to enclose all routing.

- **Layer mismatch:** ensure that all copper layers used by the routing are enabled in both the reference and target rule areas. Check the rule area properties on the **Layers** tab.
- **Unrelated routing filtered out:** if **Restrict to routing connected within the area** is checked, tracks and vias whose net does not appear on any pad inside the rule area will be skipped. This is a net-based check, so a via that shares a net with a pad inside the area will be copied even if it is not physically routed to that pad. Uncheck this option if you need to copy routing on nets that have no pads inside the rule area.
- **Net reassignment to wrong net:** if a track belongs to a global or shared net that is not unique to the channel (e.g. a power net), the duplicated track may retain the reference channel's net instead of being reassigned. Enabling **Restrict to routing connected within the area** can help avoid copying such unrelated tracks.

## Footprint placement is not copied

- **Footprint outside reference area:** footprints must be enclosed by or intersect the reference rule area. Footprints fully outside the area are not copied.
- **Layer not enabled:** a footprint is only copied if its layer is enabled in both the reference and target rule areas.
- **Locked footprints:** if **Include locked items** is not checked, locked footprints in the reference area will not be copied, and locked footprints in the target area will not be moved.

## Other items do not appear in target channels

- **Item not fully enclosed:** zones and graphic objects must be fully enclosed by the reference rule area to be copied. A large copper zone that only intersects the rule area boundary will not be copied.
- **Zone layer mismatch:** a zone is only copied if all of its layers are present in both the reference and target rule areas.
- **Locked items:** if **Include locked items** is not checked, locked items in the reference area will not be copied and locked items in the target area will not be updated.

## Target rule area shape is wrong

Target rule area outlines are automatically updated to match the shape of the reference rule area (translated and rotated to the target position). If a target rule area has a shape you did not expect, check the reference rule area's outline — the target is a direct copy of it.

## Tips for clean multichannel designs

- **Use hierarchical sheets** with a single sheet file instantiated multiple times. This guarantees identical circuitry across channels.
- **Name your rule areas** descriptively (e.g. Channel 1, Channel 2) so you can easily identify them in the Repeat Layout dialog.
- **Make the reference rule area generous** in size. It is better to have extra space around routing than to accidentally clip routing at the boundary.
-

**Route the reference channel completely** before running Repeat Layout. Partial routing can be replicated, but it is easier to verify a complete channel.

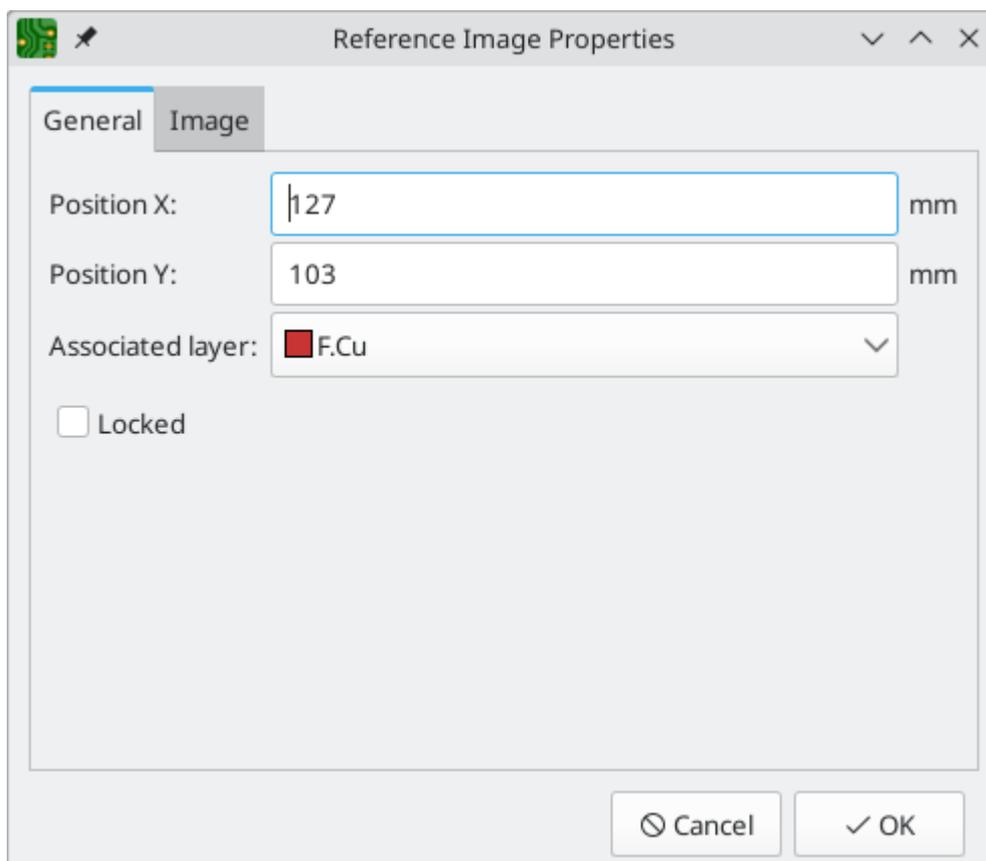
- **Group items with rule areas** for convenient manipulation. After replication, each channel's components, routing, and rule area can be moved as a unit.
- **Run DRC after replication** to catch any issues with the duplicated layout, such as clearance violations caused by different board areas or net assignment problems.

## Using reference images

KiCad supports displaying reference images in the canvas. These are background images that you can use to help you lay out a board; they are purely for reference during the design process and are not included in any fabrication outputs.

To add a reference image, use the  button on the right toolbar and select the desired reference image file. Click in the canvas to place the image.

Once the image has been added to the canvas, you can reposition it using the move tool () or by dragging it in the canvas. You can scale it by dragging the editing handles at the corners of the image. The image is scaled around its reference point; in other words, the reference point is the point in the image that always stays in the same position in the canvas, no matter how the image is scaled. The reference point is shown as a fifth editing handle. Initially it is at the center of the image, but you can reposition the reference point by dragging it in the canvas.



You can also reposition or scale the image in its properties dialog (). You can set the image's exact **Position X** and **Y** in the **General** tab, and set an exact **Scale** factor in the **Image** tab. You can also **Convert to Greyscale** if you wish. Position and scale in this dialog are relative to the center of the image, not its interactive reference point.

Reference images have an associated board layer; they are shown and hidden along with this layer. The layer initially associated with a reference image is the layer that was active when the image was added. You can change the associated layer in the image's properties.

Another way to hide reference images is with the Appearance Manager. You can show or hide all reference images by toggling the visibility of **Image** objects in the **Objects** tab ( button). You can also adjust the opacity of reference images here.

# Forward and back annotation

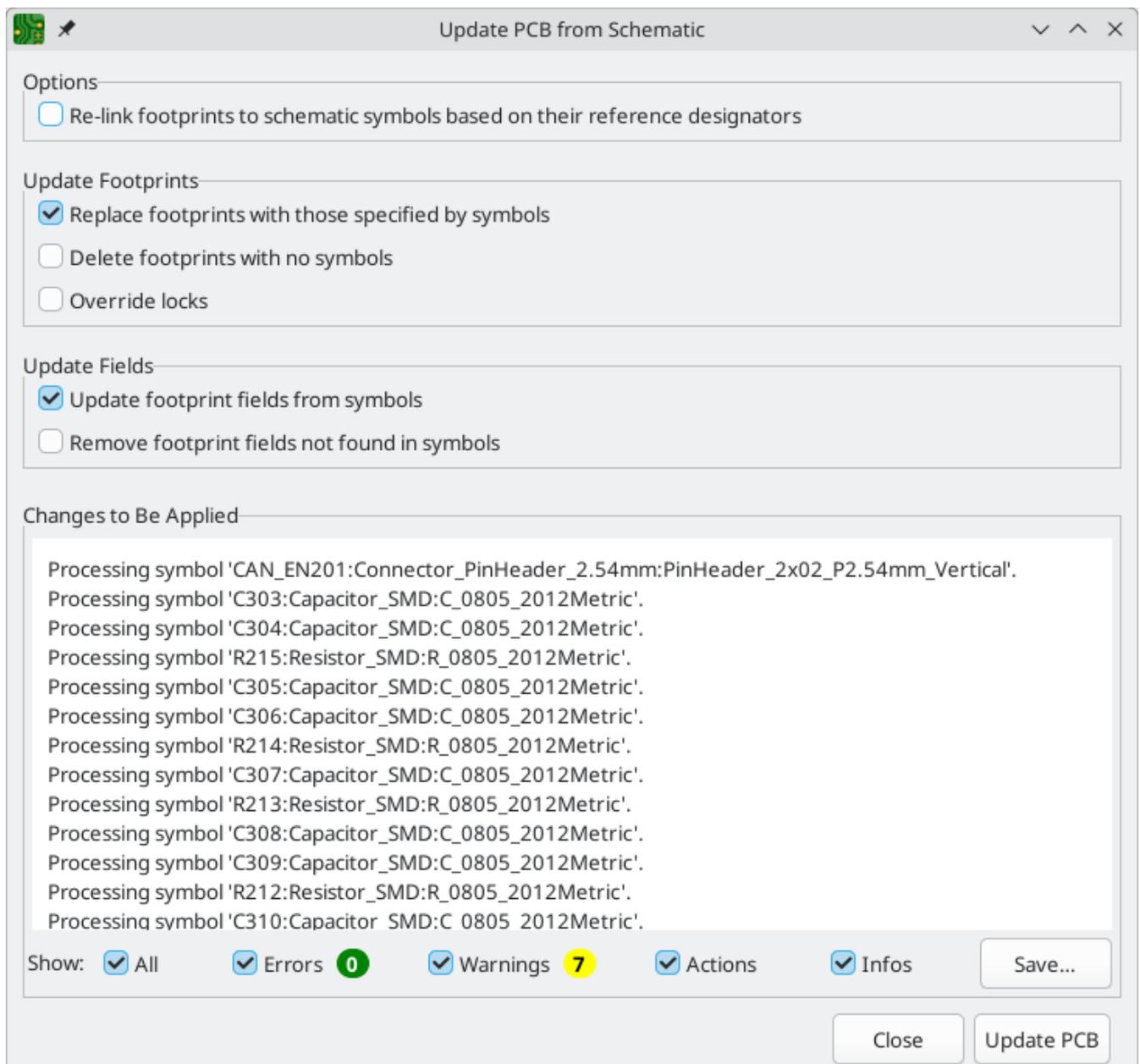
Forward and back annotation are the processes for syncing schematic changes to the board and syncing board changes to the schematic, respectively.

## Update PCB From Schematic (forward annotation)

Use the Update PCB from Schematic tool to sync design information from the Schematic Editor to the Board Editor. The tool can be accessed with **Tools** → **Update PCB from Schematic** (**F8**) in both the schematic and board editors. You can also use the  icon in the top toolbar of the Board Editor. This process is often called forward annotation.

### NOTE

Update PCB from Schematic is the preferred way to transfer design information from the schematic to the PCB. In older versions of KiCad, the equivalent process was to export a netlist from the Schematic Editor and import it into the Board Editor. It is no longer necessary to use a netlist file.



The tool adds the footprint for each symbol to the board and transfers updated schematic information to the board. In particular, the board's net connections are updated to match the schematic.

The changes that will be made to the PCB are listed in the *Changes To Be Applied* pane. The PCB is not modified until you click the **Update PCB** button.

You can show or hide different types of messages using the checkboxes at the bottom of the window. A report of the changes can be saved to a file using the **Save...** button.

## Options

The tool has several options to control its behavior.

Option	Description
<p>Re-link footprints to schematic symbols based on their reference designators</p>	<p>Footprints are normally linked to schematic symbols via a unique identifier created when the symbol is added to the schematic. A symbol's unique identifier cannot be changed, but will be lost when the symbol is deleted, even if a symbol with the same reference designator replaces it.</p> <p>If checked, each footprint in the PCB will be re-linked such that each footprint has its unique identifier updated to match the symbol that has the same reference designator as the footprint.</p> <p>This option should generally be left unchecked. See <a href="#">below</a> for more details on when to use this option.</p>
<p>Group footprints based on symbol group</p>	<p>If checked, footprints will be added to <a href="#">groups</a> in the PCB if their linked symbols are grouped.</p>
<p>Replace footprints with those specified by symbols</p>	<p>If checked, footprints in the PCB will be replaced with the footprint that is specified in the corresponding schematic symbol.</p> <p>If unchecked, footprints that are already in the PCB will not be changed, even if the schematic symbol is updated to specify a different footprint.</p>
<p>Delete footprints with no symbols</p>	<p>If checked, any footprint in the PCB without a corresponding symbol in the schematic will be deleted from the PCB. Footprints with the "Not in schematic" attribute will be unaffected.</p> <p>If unchecked, footprints without a corresponding symbol will not be deleted.</p>
<p>Override locks</p>	<p>If checked, locking a footprint will not affect whether a footprint is deleted or replaced based on changes in the schematic.</p> <p>If unchecked, locked footprints will never be deleted or replaced even if they otherwise would be.</p>
<p>Update footprint fields from symbols</p>	<p>If checked, new and updated fields in symbols will be transferred to the corresponding footprints, keeping symbol and footprint fields in sync.</p> <p>If unchecked, footprint fields will not be updated when fields change in the corresponding symbols.</p>
<p>Remove footprint fields not found in symbols</p>	<p>If checked, footprint fields will be removed if they do not exist in the corresponding symbol.</p> <p>If unchecked, footprint fields that do not exist in the corresponding symbol will not be removed, allowing footprints to have additional fields compared to the corresponding symbols.</p>

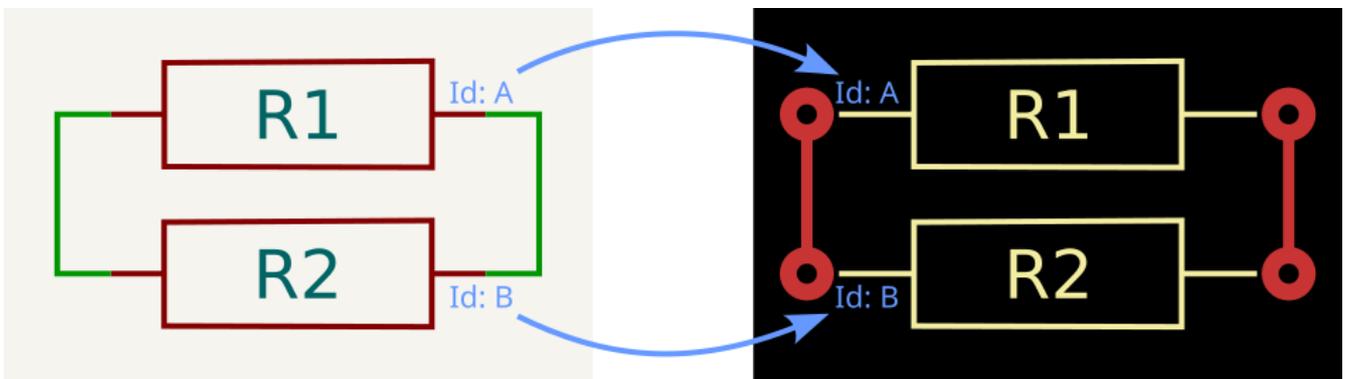
## Re-linking symbols and footprints

Symbols and footprints are linked together using unique identifiers (also called UUIDs). These are handled automatically within KiCad and are not usually visible to users. They allow a symbol and its partner footprint to keep their connection between schematic and PCB, even if the reference designator is changed. New objects get assigned their identifiers upon creation.

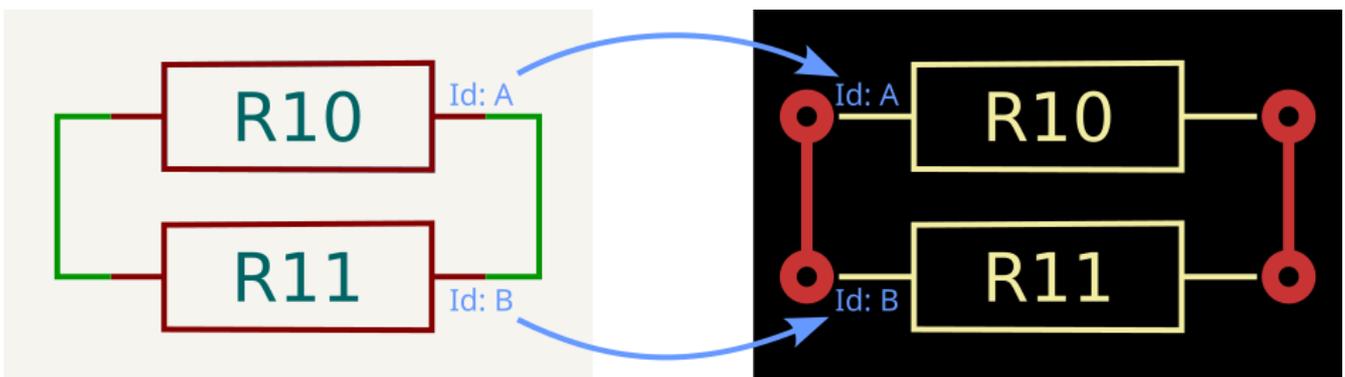
### Re-linking by unique identifier (default)

In normal use, the **Re-link footprints to schematic symbols based on their reference designators** option should be unchecked. In this mode, symbols with the same identifier as a footprint will update that footprint, regardless of the reference designator. Symbols which have an identifier that doesn't match any footprint will add a new footprint linked to that identifier.

For example, in the below schematic, both R1 and R2 are linked via their unique IDs to footprints on the PCB:



If symbol reference designators are changed in the schematic (e.g. by re-annotation), running the **Update PCB from Schematic** process will update the reference designators on the PCB.

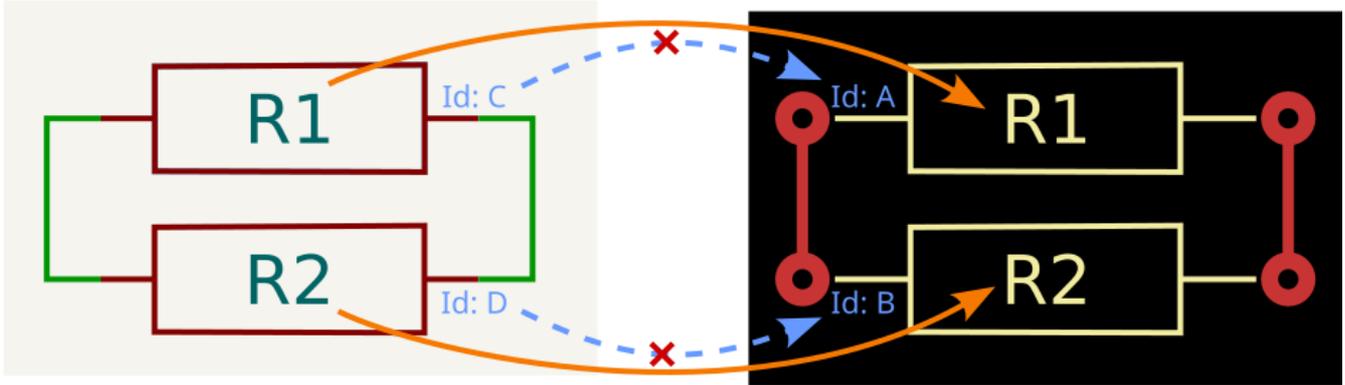


### Re-linking by reference designator

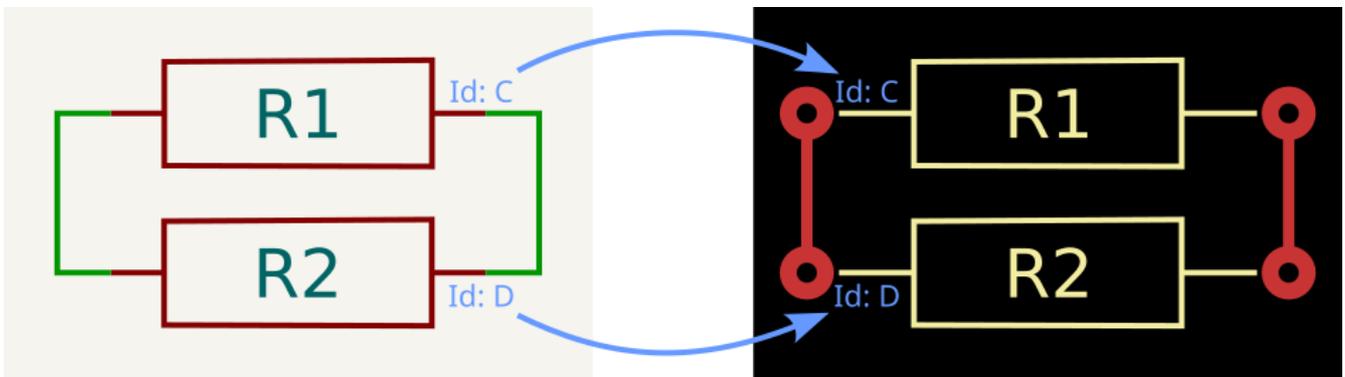
If the checkbox is checked, the linking process is done using the reference designators. This can be useful for workflows that result in a symbol being deleted and replaced by another one, rather than being updated in-place. For example, cut-and-pasting a block of schematic or a sheet and copy-pasting and re-annotating will usually break the identifier-based links.

For example in the below case, the resistors R1 and R2 have been deleted and replaced, then re-annotated. While the reference designators are the same, the internal identifiers have changed. Updating the PCB by identifier would cause the existing footprints to be deleted and new ones added - to KiCad, the existing

footprints have no matching symbol. This would cause the footprints to lose their positions and need placing again.



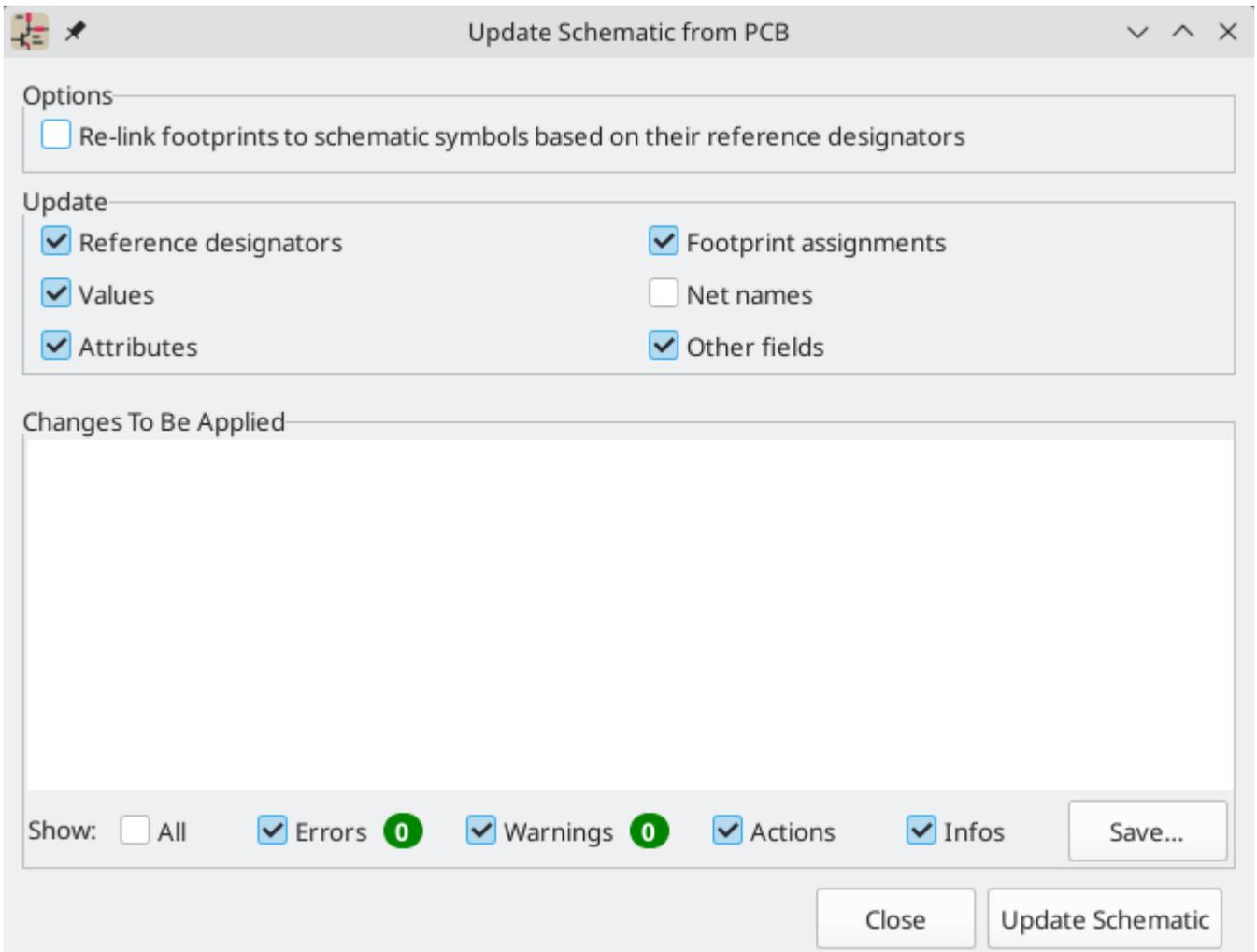
Re-linking the footprints by reference designator causes KiCad to re-create the links, using the matching reference designators as a guide.



Because the links have been re-established, the next forward annotation should use the normal identifier-based linking (i.e. the checkbox should be unchecked).

## Update Schematic from PCB (back annotation)

The typical workflow in KiCad is to make changes in the schematic and then sync the changes to the board using the Update PCB From Schematic tool. However, the reverse process is also possible: design changes can be made in the board and then synced back to the schematic using **Tools** → **Update Schematic From PCB** in either the schematic or board editors. This process is often called back annotation.



The tool syncs changes in reference designators, values, footprint assignments, and net names from the board to the schematic. Each type of change can be individually enabled or disabled.

The changes that will be made to the schematic are listed in the *Changes To Be Applied* pane. The schematic is not modified until you click the **Update Schematic** button.

You can show or hide different types of messages using the checkboxes at the bottom of the window. A report of the changes can be saved to a file using the **Save...** button.

## Options

The tool has several options to control its behavior.

Option	Description
Re-link footprints to schematic symbols based on their reference designators	<p>If checked, each footprint in the PCB will be re-linked to the symbol that has the same reference designator as the footprint. This option is incompatible with updating symbol reference designators.</p> <p>If unchecked, footprints and symbols will be linked by unique identifier as usual, rather than by reference designator.</p>
Reference designators	<p>If checked, symbol reference designators will be updated to match the reference designators of the linked footprints.</p> <p>If unchecked, symbol reference designators will not be updated.</p>
Values	<p>If checked, symbol values will be updated to match the values of the linked footprints.</p> <p>If unchecked, symbol values will not be updated.</p>
Footprint assignments	<p>If checked, footprint assignments will be updated for symbols which have had their footprints changed or replaced in the board.</p> <p>If unchecked, symbol footprint assignments will not be updated.</p>
Net names	<p>If checked, the schematic will be updated with any net name changes that have been made in the board. Net labels will be updated or added to the schematic as necessary to match the board.</p> <p>If unchecked, net names will not be updated in the schematic.</p>

**NOTE**

The [Geographical Reannotation](#) feature can be used in combination with backannotating reference designators to reannotate all components in the design based on their location in the layout.

## Back annotation with CMP files

Select changes can also be synced from the PCB back to the schematic by exporting a CMP file from the PCB editor (**File** → **Export** → **Footprint Association (.cmp) File...**) and importing it in the Schematic Editor (**File** → **Import** → **Footprint Assignments...**).

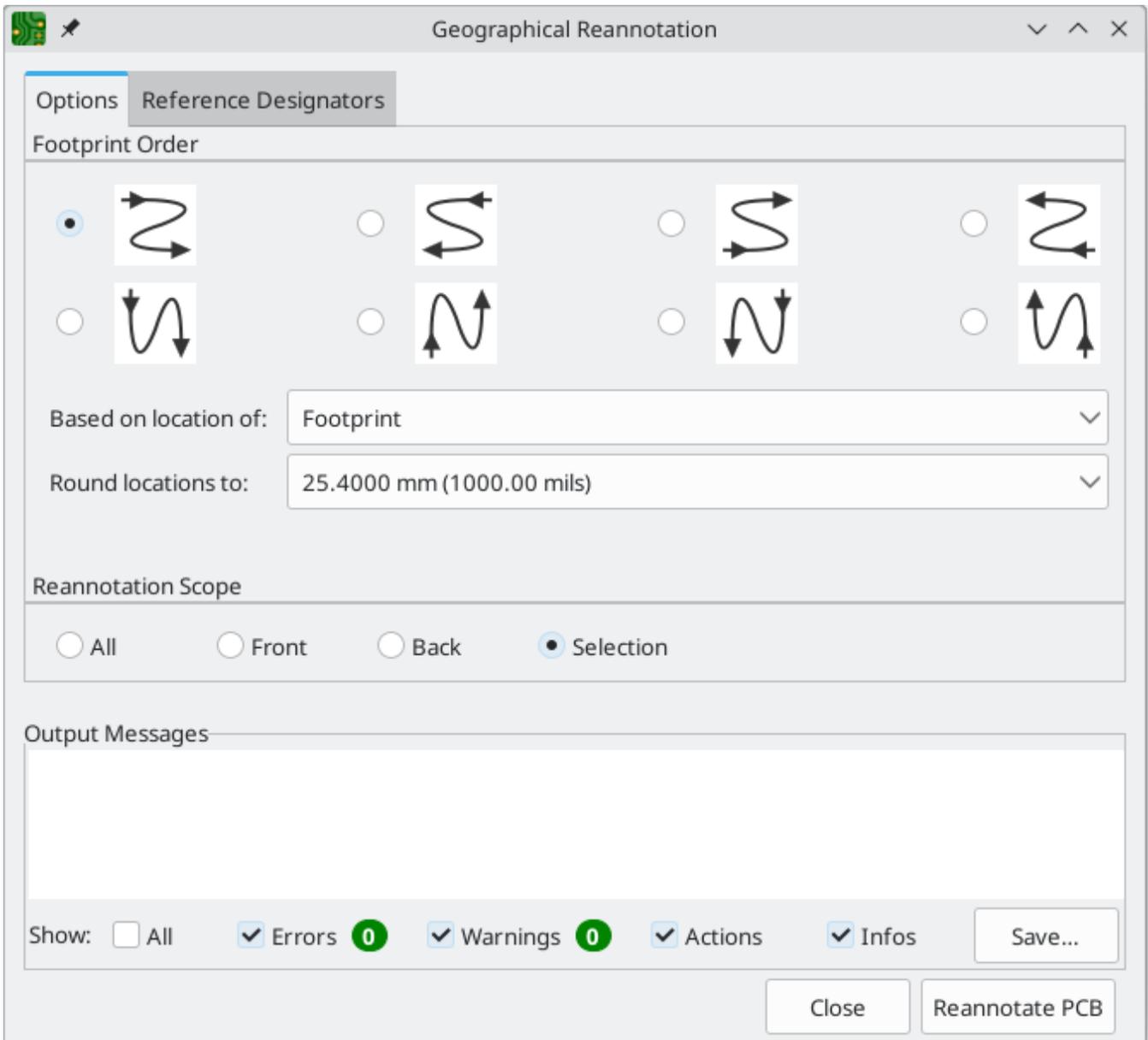
**NOTE**

This method can only sync changes made to footprint assignments and footprint fields. It is recommended to use the Update Schematic from PCB tool instead.

## Geographical re-annotation

The Geographical Reannotation tool lets you automatically set the reference designators of footprints based on their physical location on the board.

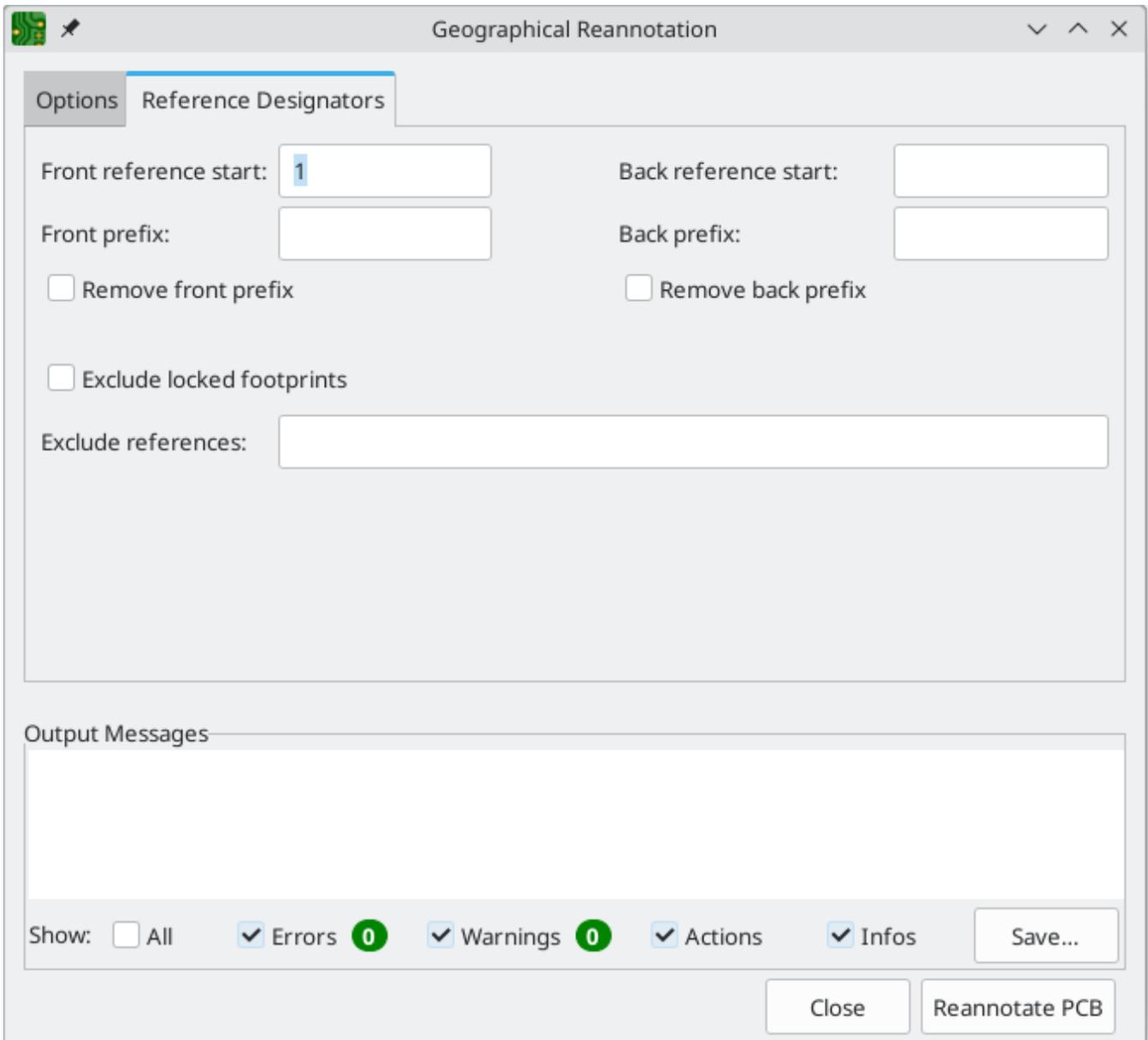
To run the Geographical Reannotation tool, use **Tools** → **Geographical Reannotate....** This opens the geographical reannotation dialog with options for how to perform the reannotation.



The **Options** tab contains settings for how footprint locations affect reannotation. The arrow diagrams indicate which geographical ordering to use when reannotating. You can reannotate from left-to-right, right-to-left, top-to-bottom, or bottom-to-top, and you can select whether to use a column-major order (go through all footprints in the same column before moving to the next column) or row-major order (go through all footprints in the same row before moving to the next row).

Geographical reannotation can either use the location of the footprint itself or the location of the footprint's reference designator. You can also select how much to round footprint locations before determining which footprints are at the same X or Y position. Rounding to a finer coordinate resolution will result in fewer footprints considered to be in the same row or column.

Finally, you can select which footprints to reannotate. You can reannotate all footprints on the board, all footprints on the front or back of the board, or all footprints in your selection.



The **Reference Designators** tab contains options for how to allocate new reference designators. There are separate settings for footprints on the front and back of the board.

**Reference start** controls the number for the first new reference designator on each side of the board. If no start value is given for the back of the board, back side footprints will be annotated starting at one higher than the last front side reference designator.

**Prefix** specifies a prefix string to insert at the beginning of each newly assigned reference designator. This prefix will be inserted before any prefix that is already present. If the **remove prefix** option is selected, footprints with the specified prefix will instead have that prefix removed instead of added. Footprints without that prefix will not have any prefix added or removed.

If **exclude locked footprints** is checked, locked footprints will not be reannotated. You can also avoid reannotating specific footprints by entering their reference designators as a comma-separated list in the **exclude references** box.

When you click the **Reannotate PCB** button, footprints will be reannotated according to the selected settings.

**NOTE**

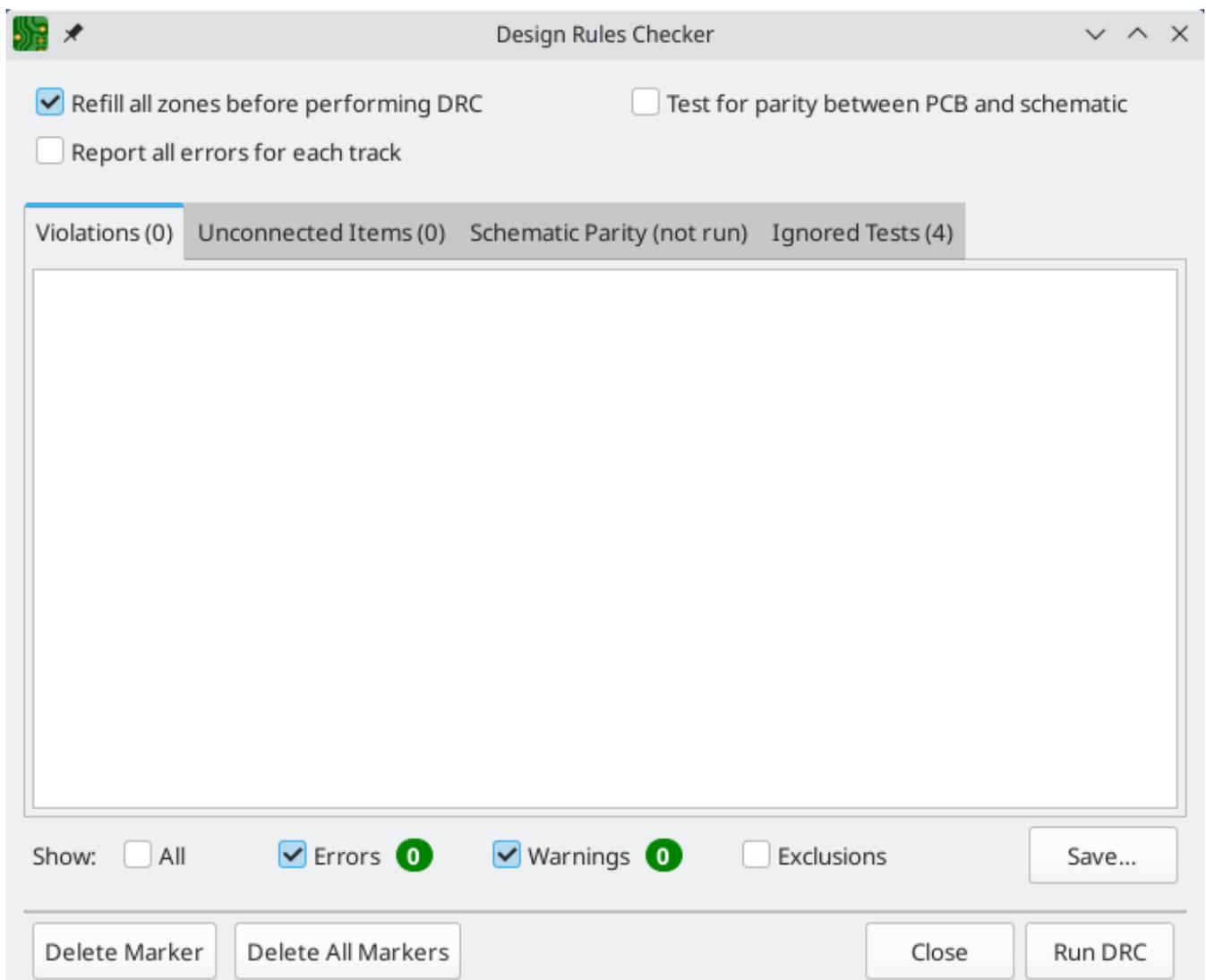
The Geographical Reannotation tool updates reference designators in the board, but not in the schematic. After geographically reannotating the board, be sure to sync the updated reference designators to the schematic by running the [Update Schematic from PCB](#) tool with the **re-link footprints to schematic symbols based on their reference designators** option disabled. If the schematic is not updated, reference designators in the board will not match those in the schematic.

# Inspecting a board

## Design rules checking

The Design Rules Checker (DRC) tool is used to verify that the PCB meets all the requirements established in the Board Setup dialog and that all pads are connected according to the netlist or schematic. KiCad can automatically prevent some design rule violations while routing tracks, but many others cannot be prevented automatically. This means it is important to use the design rule checker before creating manufacturing files for a PCB.

To use the design rule checker, click the  icon in the top toolbar, or select **Design Rules Checker** from the **Inspect** menu.



The top section of the DRC Control window contains some options that control the design rule checker:

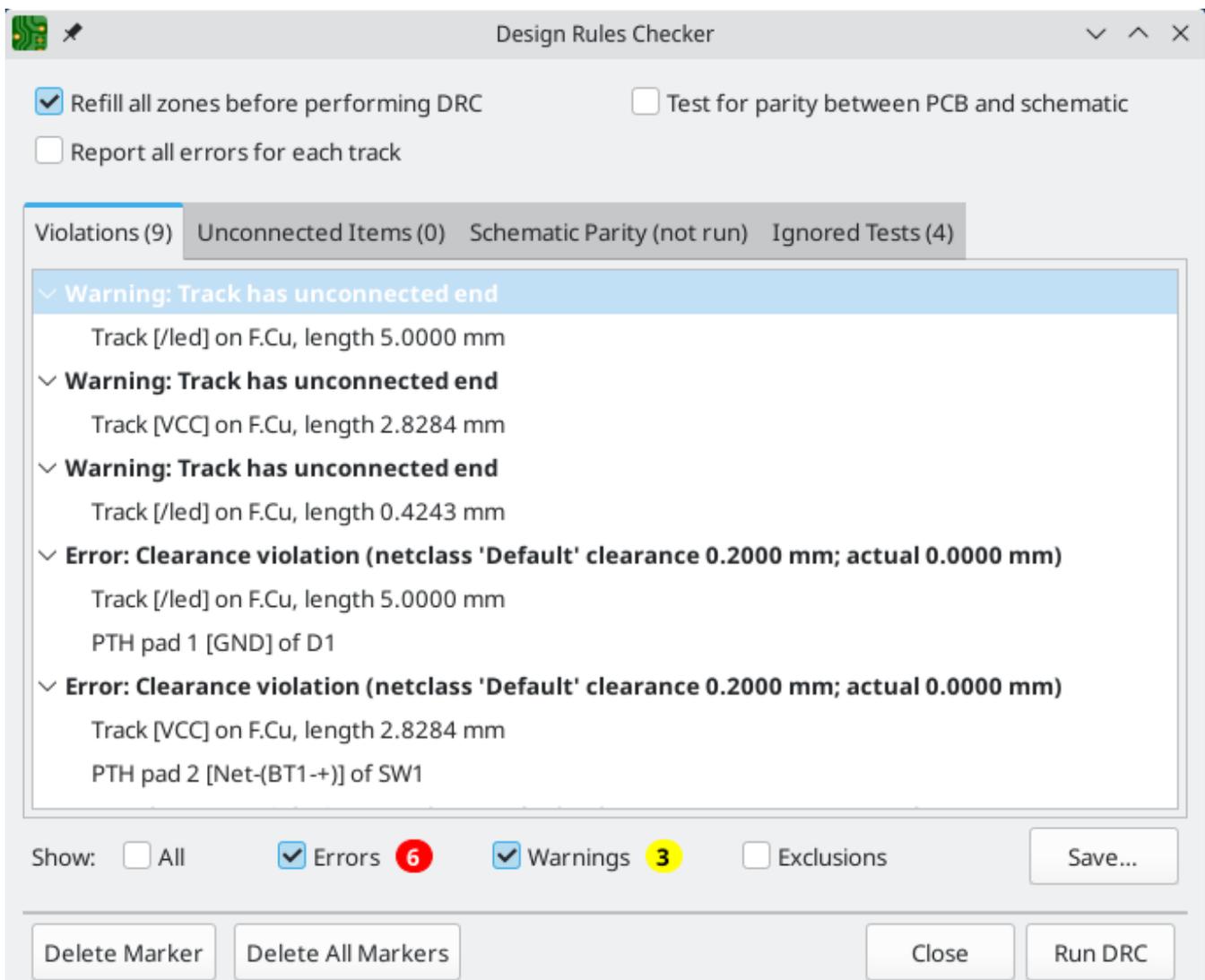
**Refill all zones before performing DRC:** when enabled, zones will be refilled every time the design rule checker is run. Disabling this option may result in incorrect DRC results if zones have not been refilled manually.

**Report all errors for each track:** when enabled, all clearance errors will be reported for each track segment. When disabled, only the first error will be reported. Enabling this option will result in the design

rule checker running more slowly.

**Test for parity between PCB and schematic:** when enabled, the design rule checker will test for differences between the schematic and PCB in addition to testing the PCB design rules. This option has no effect when running the PCB editor in standalone mode.

After running DRC, any violations will be shown in the center part of the DRC window. Rule violations, unconnected items, and differences between the schematic and the PCB are shown in three different tabs. A list of the ignored tests is shown in the fourth tab. A report file in plain text format can be created after running DRC using the **Save...** button.



Each violation involves one or more objects on the PCB. In the list of violations, the objects involved are listed below the violation. Clicking on the violation in the list view will move the PCB Editor view so that the affected area is centered. Clicking on one of the objects involved in a violation will highlight the object.

Many types of violations have contextual actions in the context menu. For example, clearance violations have an action to run the [clearance resolution tool](#) on the violating items, while custom rule violations have an action to run the [constraint resolution tool](#). For board vs. library footprint mismatch violations, there is an action to run the [Compare Footprint with Library](#) tool and another action to [update the footprint from the library](#). These actions can help to quickly fix identify the reason for a particular violation.

The numbers at the bottom of the window show the number of errors, warnings, and exclusions. Each type of violation can be filtered from the list using the respective checkboxes. Clicking **Delete Marker** will clear the selected violation until DRC is run again, while clicking **Delete All Markers** will clear all violations until the next DRC run.

Violations can be right-clicked in the dialog to ignore them or change their severity:

- **Exclude this violation:** ignores this particular violation, but does not affect any other violations. You can un-exclude a violation by right clicking the excluded violation and selecting **Remove exclusion for this violation**.
- **Exclude with comment...:** the same as **Exclude this violation**, but prompts for a comment explaining the reason for the exclusion. When excluded violations are unhidden (using the **Exclusions** checkbox), exclusion comments are shown with the corresponding excluded violation. To edit an existing exclusion comment or add a comment to an existing exclusion, right click an excluded violation and select **Edit exclusion comment...**
- **Exclude all violations of rule:** the same as **Exclude this violation**, but excludes *all* violations caused by the same [custom DRC rule](#). This action only appears in the context menu for violations caused by custom design rules. If you right click on a custom design rule violation that is already excluded, you can instead **Remove all exclusions for violations of rule**.
- **Change severity:** changes a type of violation from warning to error, or error to warning. This affects all violations of a given type.
- **Ignore all:** ignores all violations of a given type. This test will now appear in the **Ignored Tests** tab rather than the **Violations** tab. You can un-ignore the test again by right clicking the test in the **Ignored Tests** tab, or in the [Violation Severity panel](#) in Board Setup.
- **Edit violation severities...:** opens the [Violation Severity panel](#) in Board Setup, for editing the severities of all DRC violation types.

Excluded and ignored violations are remembered between runs of the design rule checker. Excluded violations are hidden unless the **Exclusions** checkbox is enabled. Ignored violations are not shown, but there is a list of ignored tests in the **Ignored Tests** tab.

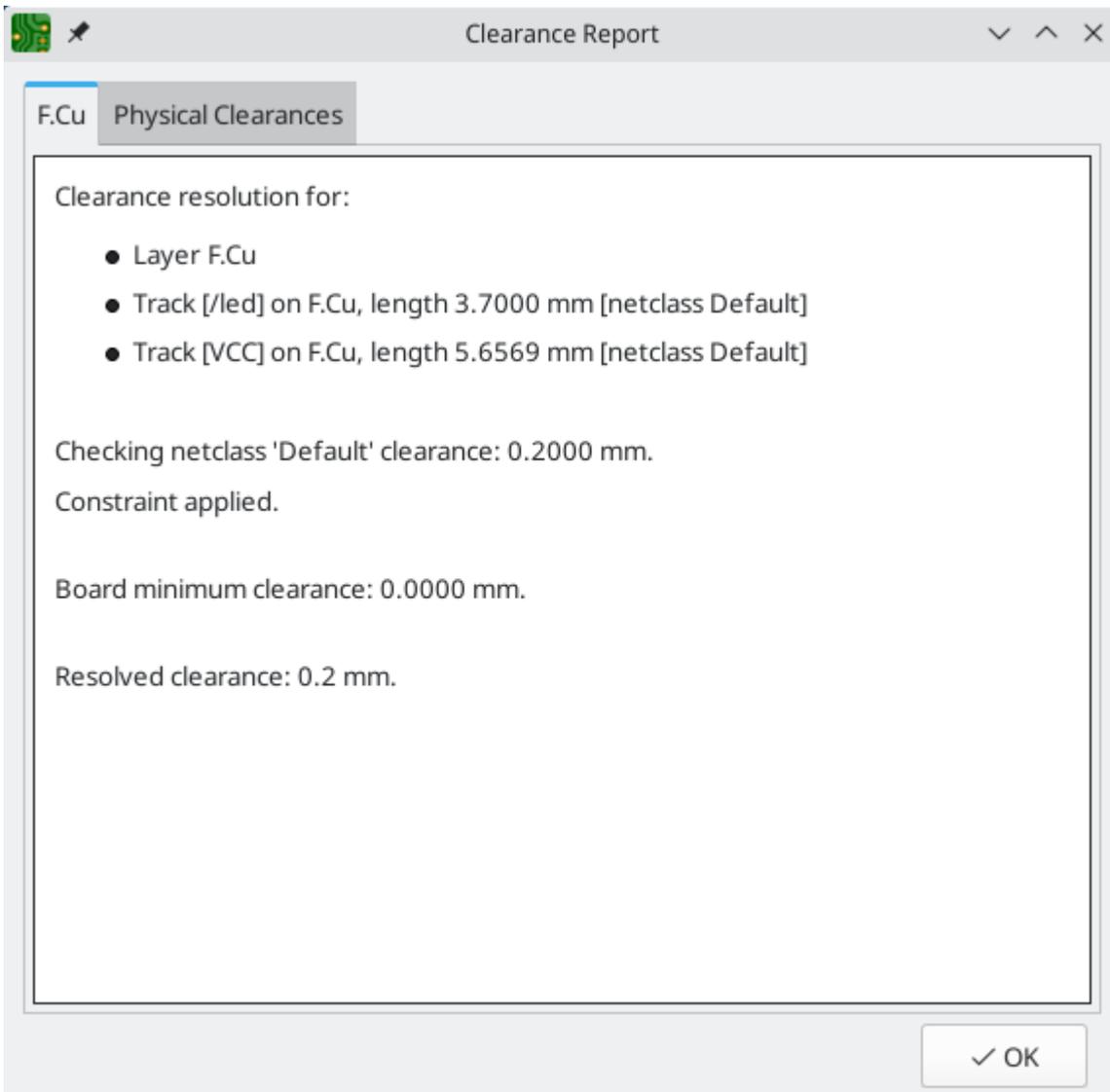
## Clearance and constraint resolution

The clearance and constraint resolution tools allow you to inspect which clearance and design constraint rules apply to selected items. These tools can help when designing PCBs with complex design rules where it is not always clear which rules apply to an object.

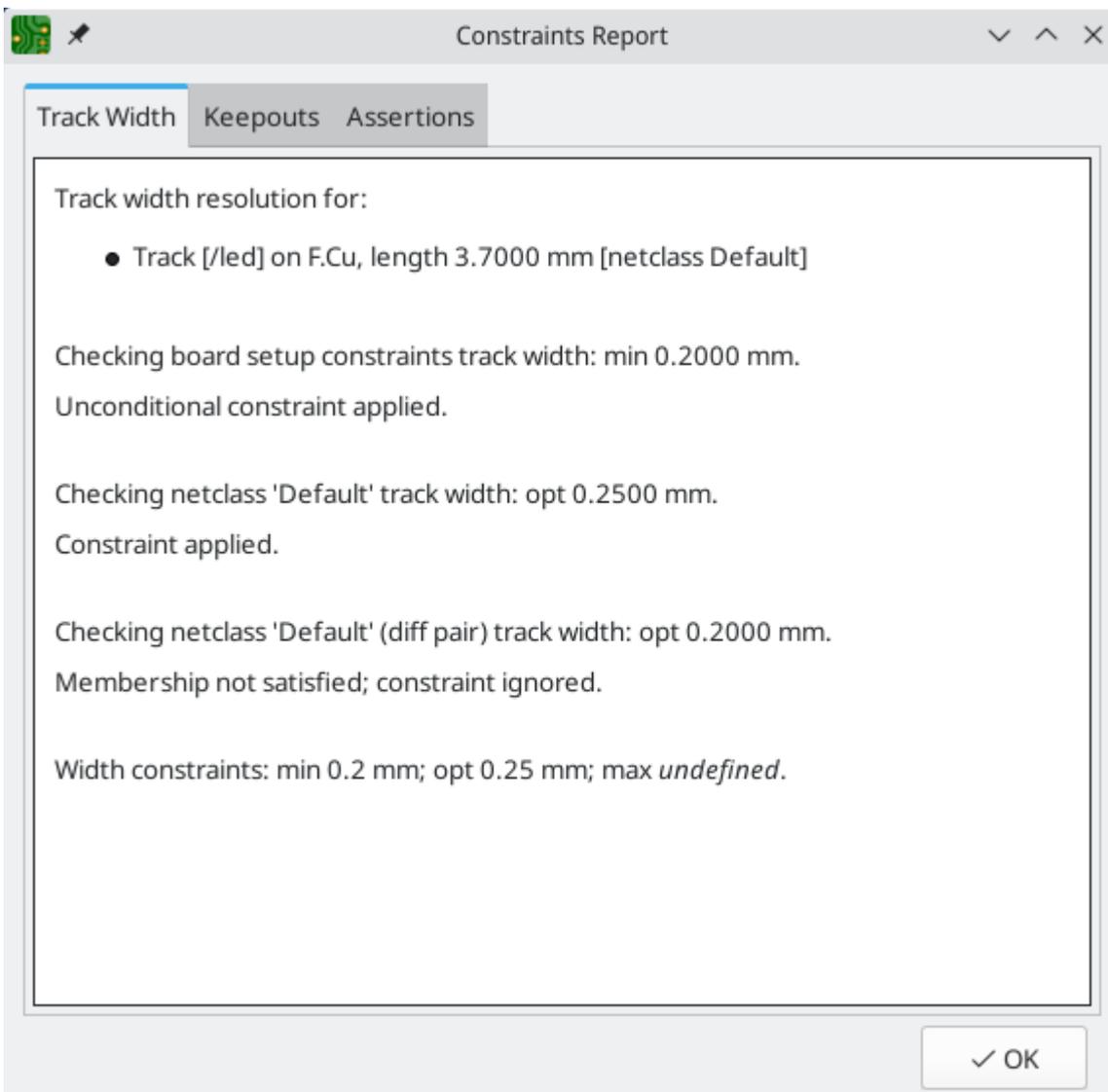
To inspect the clearance rules that apply between two objects, select both objects and choose **Clearance Resolution** from the **Inspect** menu. If you haven't selected two objects, you are prompted to pick them. The Clearance Report dialog will show the clearance required between the objects on each copper layer, as well as the design rules that resulted in that clearance.

### NOTE

If you don't select two items before running the Clearance Resolution tool, you are prompted to pick two items interactively. This can be useful to check the clearance between two items that would otherwise be difficult to select at the same time, such as items in two different groups.



To inspect the design constraints that apply to an object, select it and choose **Constraints Resolution** from the **Inspect** menu. If you haven't selected an object, you are prompted to pick one. The Constraints Report dialog will show any constraints that apply to the object.

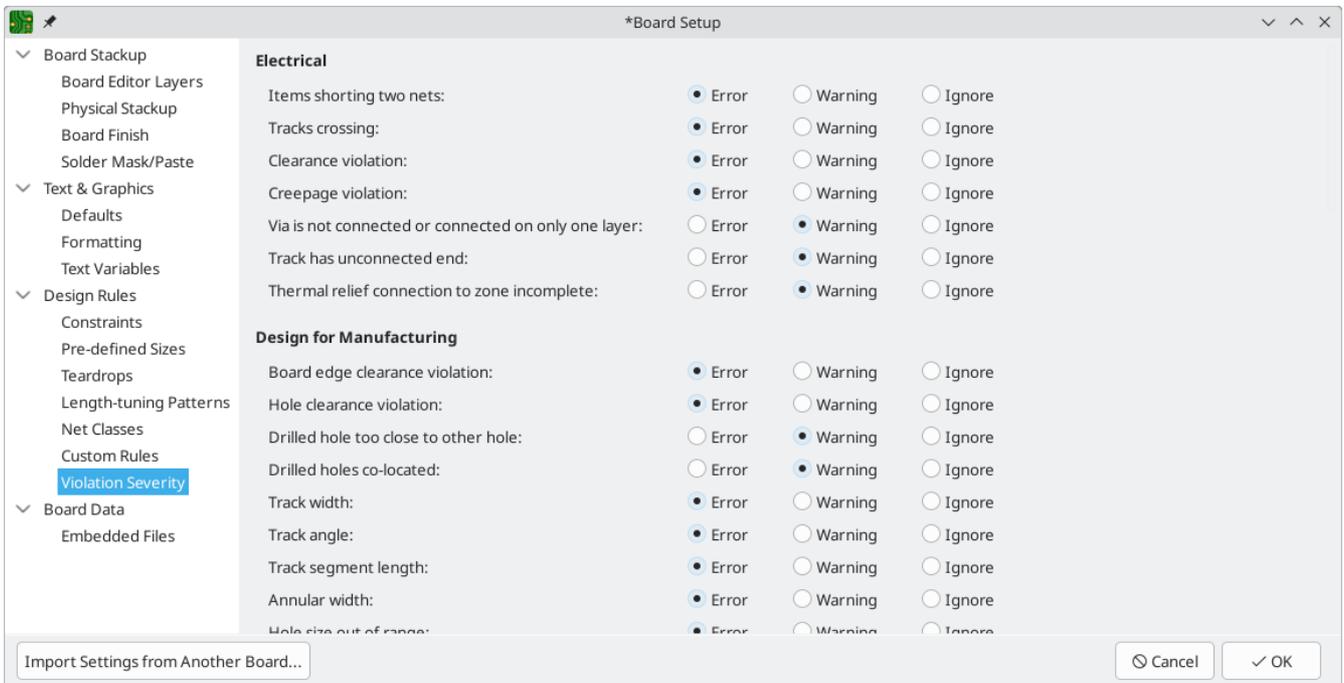


## DRC configuration

The severity of each DRC check can be configured in the **Violation Severity** section of the [Board Setup dialog](#). Each rule may be set to create an error marker, a warning marker, or no marker (ignored).

### NOTE

Individual rule violations may be ignored in the Design Rule Checker. Setting a rule to Ignore in the Violation Severity section will completely disable the corresponding design rule check. Use this setting with caution.



## List of DRC checks

The table below lists the design rules that KiCad checks and the default violation severity for each check. All severities are configurable. Some design are only available through [custom design rules](#).

### Electrical DRC checks

These DRC checks look for gross electrical issues on the board such as shorts and clearance violations.

Violation	Description	Default Severity
Items shorting two nets	This violation occurs when copper items on different nets collide with each other. If this is intentional, consider using a <a href="#">net tie</a> .	Error
Tracks crossing	This violation occurs when tracks with different nets cross each other.	Error

Violation	Description	Default Severity
Clearance violation	<p>This violation occurs when the distance between two copper items with different nets is smaller than the configured clearance for those nets. The allowed clearance between two items can come from the <a href="#">board-level minimum clearance</a>, the <a href="#">net class settings</a> for each net, or from <a href="#">custom rules</a>. To see detailed information about the configured and actual clearances between two selected items, run the <a href="#">clearance resolution</a> tool, which is available by right clicking the violation in the DRC window. The minimum clearance path is highlighted in the editing canvas when a clearance violation is selected in the DRC window.</p> <p>This violation is also reported when the distance between two items is smaller than the configured physical clearance for those two items. Physical clearance constraints are not configured by default; see the <a href="#">custom rule</a> documentation for how to configure physical clearance.</p>	Error
Creepage violation	<p>This violation occurs when the creepage distance between two copper items with different nets is smaller than the configured creepage for those nets. Creepage paths are highlighted in the editing canvas when a creepage violation is selected in the DRC window.</p> <p>Creepage distances can be configured using a creepage constraint in <a href="#">custom rules</a>.</p>	Error
Via is not connected or is connected on only one layer	<p>This violation occurs when a via is connected to copper objects on only one layer or is not connected to anything. As vias are intended to connect copper objects on different layers, this may indicate that an intended connection is missing.</p>	Warning
Track has unconnected end	<p>This violation occurs when the end of a track segment is not connected to another copper object, such as another track segment, a via or pad, or a zone or copper graphical shape.</p>	Warning

Violation	Description	Default Severity
Board edge clearance violation	<p>This violation occurs when the distance between a copper object and the board edge is smaller than the configured copper to edge clearance for those items. For the purposes of this check, oval holes (which are routed rather than drilled) are counted as board edges in addition to any graphic items on the Edge.Cuts layer.</p> <p>The allowed edge clearance between two items can come from the <a href="#">board-level minimum copper to edge clearance</a> or from <a href="#">custom rules</a>. A negative edge clearance allows objects to overlap with the board edge. To see detailed information about the configured and actual edge clearances between two selected items, run the <a href="#">clearance resolution</a> tool.</p>	Error
Hole clearance violation	<p>This violation occurs when the distance between a hole (pad or via) and another copper object (pad, track, via, or zone) is smaller than the configured copper to hole clearance for those objects. Objects are only considered in this check if they have layers in common. The allowed hole clearance between two items can come from the <a href="#">board-level minimum copper to hole clearance</a> or from <a href="#">custom rules</a>. To see detailed information about the configured and actual hole clearances between two selected items, run the <a href="#">clearance resolution</a> tool.</p> <p>This violation is also reported when the distance between a hole and another object is smaller than the configured physical hole clearance for those two items. Physical hole clearance constraints are not configured by default; see the <a href="#">custom rule</a> documentation for how to configure physical hole clearance.</p>	Error

Violation	Description	Default Severity
Track width	<p>This violation occurs when the width of a track is outside of the configured range. The allowed width for a track can come from the <a href="#">board-level minimum track width</a> or from <a href="#">custom rules</a>.</p> <p>Note that an optimal track width can be configured for each net class in the <a href="#">net class settings</a>, which sets a track width for the interactive router to use, but it does not set a minimum and maximum track width. No DRC violations will be reported for net class track width settings unless a minimum and/or maximum are configured using custom rules.</p> <p>To see detailed information about the configured track width for a particular track, run the <a href="#">constraints resolution</a> tool.</p>	Error
Track angle	<p>This violation occurs when the angle between two connected track segments is outside the configured range.</p> <p>Minimum and/or maximum allowable track angles can be configured using a <code>track_angle</code> constraint in <a href="#">custom rules</a>.</p>	Error
Track segment length	<p>This violation occurs when the length of a track segment is outside the configured range.</p> <p>Minimum and/or maximum allowable track segment lengths can be configured using a <code>track_segment_length</code> constraint in <a href="#">custom rules</a>.</p>	Error
Annular width	<p>This violation occurs when a pad or via's annular width is outside of the configured range.</p> <p>Board-level minimum annular width can be configured in <a href="#">board setup constraints</a>. Board-level maximum width, as well as more specific rules, can be configured using <a href="#">custom rules</a>.</p>	Error

Violation	Description	Default Severity
Courtyards overlap	This violation occurs when a footprint's courtyard overlaps with another footprint's courtyard. A nonzero clearance between two courtyards can be configured using a <code>courtyard_clearance</code> constraint in <a href="#">custom rules</a> . A negative courtyard clearance allows courtyards to intersect.	Error
Footprint has no courtyard defined	This violation occurs when a footprint does not contain any graphic shapes on its <code>F.Courtyard</code> or <code>B.Courtyard</code> layers.	Ignore
Footprint has malformed courtyard	This violation occurs when a footprint has a courtyard containing non-closed shapes. Courtyards may contain multiple unconnected shapes without being considered malformed, as long as each shape is individually closed.	Error
Board has malformed outline	This violation occurs when the shapes on the <code>Edge.Cuts</code> layer do not form a valid board outline. Valid board outlines consist of closed shapes that do not self-intersect. Board outlines may contain multiple unconnected shapes without being considered malformed, as long as each shape is individually closed and does not intersect with itself or other shapes. This check also reports very small (nanometer-scale) graphic shapes on the <code>Edge.Cuts</code> layer, which are difficult to find visually but may cause issues in other tools.	Error
Copper sliver	This violation occurs when small, wedge-shaped protrusions of copper are detected. These slivers can cause manufacturing, reliability, or electrical issues.	Warning
Solder mask aperture bridges items with different nets	This violation occurs when a single opening in the soldermask exposes multiple copper items with different nets. This can result in solder shorting the two copper items during assembly.	Error
Copper connection too narrow	This violation occurs when a copper connection necks down to a width that is narrower than the configured minimum connection width. The minimum connection width setting can come from the <a href="#">board-level minimum connection width</a> or can be configured with more granularity using <a href="#">custom rules</a> .	Warning

<b>Violation</b>	<b>Description</b>	<b>Default Severity</b>
Track endpoint not centered on via	This violation occurs when a track's endpoint lies within a via but not exactly at the via center. The length tuner will not report an accurate length for a track connecting to a via outside of the via's center.	Warning
Tuning profile track geometries	This violation occurs when a track's geometry (track width or differential pair gap) do not match the values from the track's tuning profile.	Ignore

## **Schematic parity DRC checks**

These DRC checks look for differences between the schematic and the board.

Violation	Description	Default Severity
Duplicate footprints	This violation occurs when the board contains multiple footprints with the same reference designator are in the board. It is not reported if the footprints do not correspond to schematic symbols, however (if the footprints only exist in the board).	Warning
Missing footprint	This violation occurs when a footprint is not in the board but is expected based on a corresponding symbol in the schematic.	Warning
Extra footprint	This violation occurs when a footprint is in the board without a corresponding symbol in the schematic.	Warning
Footprint attributes don't match symbol	This violation occurs when a footprint's Value field, "DNP" attribute, or "Exclude from BOM" attribute are set differently than the corresponding field/attribute in the matching schematic symbol. It also occurs when a symbol's assigned footprint is different than the actual footprint in the board.  Typically this is fixed by performing an <a href="#">Update PCB from Schematic</a> or <a href="#">Update Schematic from PCB</a> action to sync the fields and attributes, depending on whether the symbol or footprint, respectively, is correct.	Warning
Footprint doesn't match symbol's footprint filters	This violation occurs when a footprint does not match footprint filters in the corresponding symbol. If the symbol doesn't have any footprint filters, no violation occurs.	Ignore
Pad net doesn't match schematic	This violation occurs when a net does not match between a footprint pad and the corresponding symbol pin. This can be because the symbol pin's net is different than the footprint pad's net, because the footprint pad does not have a corresponding symbol pin, or because the symbol pin does not have a corresponding footprint pad.	Warning
Missing connection between items	This violation occurs when two copper objects with the same net are not connected on the board.	Error

## Signal integrity DRC checks

These DRC checks look for signal integrity issues in the board.

Violation	Description	Default Severity
Track length out of range	<p>This violation occurs when a track in a differential pair is too long or too short compared to the configured minimum and maximum length for that track. The allowable track length for different tracks can be configured using the <code>length</code> constraint in <a href="#">custom rules</a>.</p>	Error
Skew between tracks out of range	<p>This violation occurs when the difference between the length of a track and the maximum length of all tracks being considered is longer than the configured maximum skew for that set of tracks. For calculating the skew of a differential pair (two tracks), the skew therefore is calculated as the length difference between tracks.</p> <p>The allowable maximum skew for a set of tracks, as well as which tracks the rule applies to, can be configured using the <code>skew</code> constraint in <a href="#">custom rules</a>.</p>	Error
Too many or too few vias on a connection	<p>This violation occurs when the number of vias assigned to a net is too low or too high compared to the configured minimum and maximum for that net. The allowable via count for different nets can be configured using the <code>via_count</code> constraint in <a href="#">custom rules</a>.</p>	Error
Differential pair gap out of range	<p>This violation occurs when the gap between the two tracks in a differential pair is too small or too large compared to the configured minimum and maximum for that differential pair. The gap is only checked on coupled (i.e. parallel) portions of the differential pair.</p> <p>The minimum and maximum allowable gap for a differential pair can be configured using the <code>diff_pair_gap</code> constraint in <a href="#">custom rules</a>.</p> <p>Note that an optimal differential pair gap can be configured for each net class in the <a href="#">net class settings</a>, which sets a gap for the differential pair router to use, but it does not set a minimum and maximum gap. No DRC violations will be reported unless a minimum and/or maximum are configured using custom rules.</p>	Error

Violation	Description	Default Severity
Silkscreen clearance	<p>This violation occurs when a silkscreen object intersects another silkscreen object, which may affect readability. Collisions that only involve shapes are not reported; for example, the intersection of two silkscreen lines doesn't cause a violation, but a line intersecting a text object does.</p> <p>The allowable distance between silkscreen objects can be set to a nonzero number to enforce a silk clearance using the <a href="#">board-level silkscreen minimum item clearance</a> or using <a href="#">custom rules with the silk_clearance constraint</a>. You can also use the <code>silk_clearance</code> constraint to enforce clearance between silkscreen and objects on other layers.</p> <p>A negative silkscreen clearance allows silkscreen to intersect other objects.</p>	Warning
Silkscreen clipped by solder mask	<p>This violation occurs when a silkscreen object intersects a solder mask opening. This may result in silkscreen printed on bare copper or substrate. Board manufacturers may also discard any silkscreen that does not have solder mask underneath. Such outcomes could affect board assembly as well as silkscreen durability and readability.</p>	Warning
Silkscreen clipped by board edge	<p>This violation occurs when a silkscreen object intersects a board edge, meaning that part of the silkscreen is outside of the board area.</p> <p>The allowable distance between silkscreen and the board edge can also be set to a nonzero number to enforce a clearance to the board edge using the <a href="#">board-level silkscreen minimum item clearance</a> or using <a href="#">custom rules with the silk_clearance constraint</a>. A negative silkscreen clearance allows silkscreen to intersect other objects.</p>	Warning
Text height out of range	<p>This violation occurs when a text object's text height is outside of the configured range.</p> <p>Board-level minimum text height can be configured in <a href="#">board setup constraints</a>. Board-level maximum height, as well as more specific rules, can be configured using <a href="#">custom rules</a>.</p>	Warning

Violation	Description	Default Severity
Non-Mirrored text on back layer	This violation occurs when a text object on a back layer doesn't have the mirrored attribute set. When looking at the back of the board, the text will therefore appear backwards.	Ignore

## Miscellaneous DRC checks

These DRC checks look for other miscellaneous issues in the board.

Violation	Description	Default Severity
Items not allowed	This violation occurs when objects are placed in a location where they are not allowed. This can be due to a <a href="#">rule area</a> with a keep out rule for the object's type or due to a disallow <a href="#">custom rule</a> constraint.	Error
Copper zones intersect	This violation occurs when copper zones with different nets collide with each other, shorting the two nets.	Error
Isolated copper fill	This violation occurs when part of a copper fill is not connected to any other copper items with the same net. This is also referred to as an island.	Warning
Footprint is not valid	This violation occurs when a footprint's net tie group contains a pad that doesn't exist in the footprint, or when a pad is in more than one net tie group.	Error

Violation	Description	Default Severity
Padstack is questionable	<p>This violation occurs when a footprint pad has unusual settings that are probably a mistake. The settings that are checked are:</p> <ul style="list-style-type: none"> <li>• Plated through holes without copper pads on any layer</li> <li>• Pads with inappropriate properties, such as through hole pads with the BGA property</li> <li>• Connector pads with solder paste</li> <li>• SMD pads with copper on both sides</li> <li>• SMD pads with copper on the opposite side from the corresponding solder mask opening or solder paste</li> <li>• SMD pads with no copper on outer layers</li> <li>• Plated through hole pads with no copper annulus around the hole</li> <li>• Plated through hole pads with hole partially or fully outside of the copper</li> <li>• Potential issues with solder mask clearance</li> <li>• Pads with negative local electrical clearance</li> <li>• Pads with an excessively large corner chamfer/radius</li> </ul>	Warning
PTH inside courtyard	<p>This violation occurs if a footprint's plated through hole pad is within the courtyard of another footprint. Pads with the "heatsink pad" fabrication property are allowed, however.</p>	Error
NPTH inside courtyard	<p>This violation occurs if a footprint's nonplated through hole pad is within the courtyard of another footprint.</p>	Error
Item on a disabled copper layer	<p>This violation occurs if an item, for example a pad or via, is on a copper layer that does not exist in the <a href="#">board stackup</a>.</p>	Error
Unresolved text variable	<p>This violation occurs when a <a href="#">text variable</a> in the board design or drawing sheet does not resolve (there is no defined value for the variable).</p>	Error

Violation	Description	Default Severity
Footprint not found in libraries	This violation occurs when a footprint in the board is not in an active library in <a href="#">the global library table</a> or <a href="#">the project-specific library table</a> . This can be because the footprint's library does not contain the footprint, the footprint's library is not listed in either library table, or because the library is listed in a table but is disabled. As a consequence, you will not be able to update the footprint from the library or compare changes between the board and library versions of the footprint.	Warning
Footprint doesn't match copy in library	This violation occurs when a footprint in the board is different than the library version of the footprint.  You can compare between the board and library versions of the footprint using the <a href="#">Compare Footprint with Library</a> tool, which is available by right clicking the violation in the DRC window. If desired, you can <a href="#">update the board footprint</a> to match the library footprint.	Warning
Through hole pad has no hole	This violation occurs when a through hole footprint pad does not have a hole.	Error

## User-definable DRC violations

You can manually trigger board DRC warnings or errors using special [text variables](#). These items will appear as errors or warnings when DRC runs. This can be useful to flag items for later followup or review.

To cause a DRC violation, use the text variable `#{DRC_ERROR <violation name>}` or `#{DRC_WARNING <violation name>}` depending on whether an error or warning is desired. You can place this in a text item or text box on any board layer. When DRC runs, this will generate a DRC violation with the given violation name. These text variables resolve to an empty string in the board, and any text after the braces is included in the DRC violation's description. The text variable must be placed at the start of the text object in order to trigger a violation.

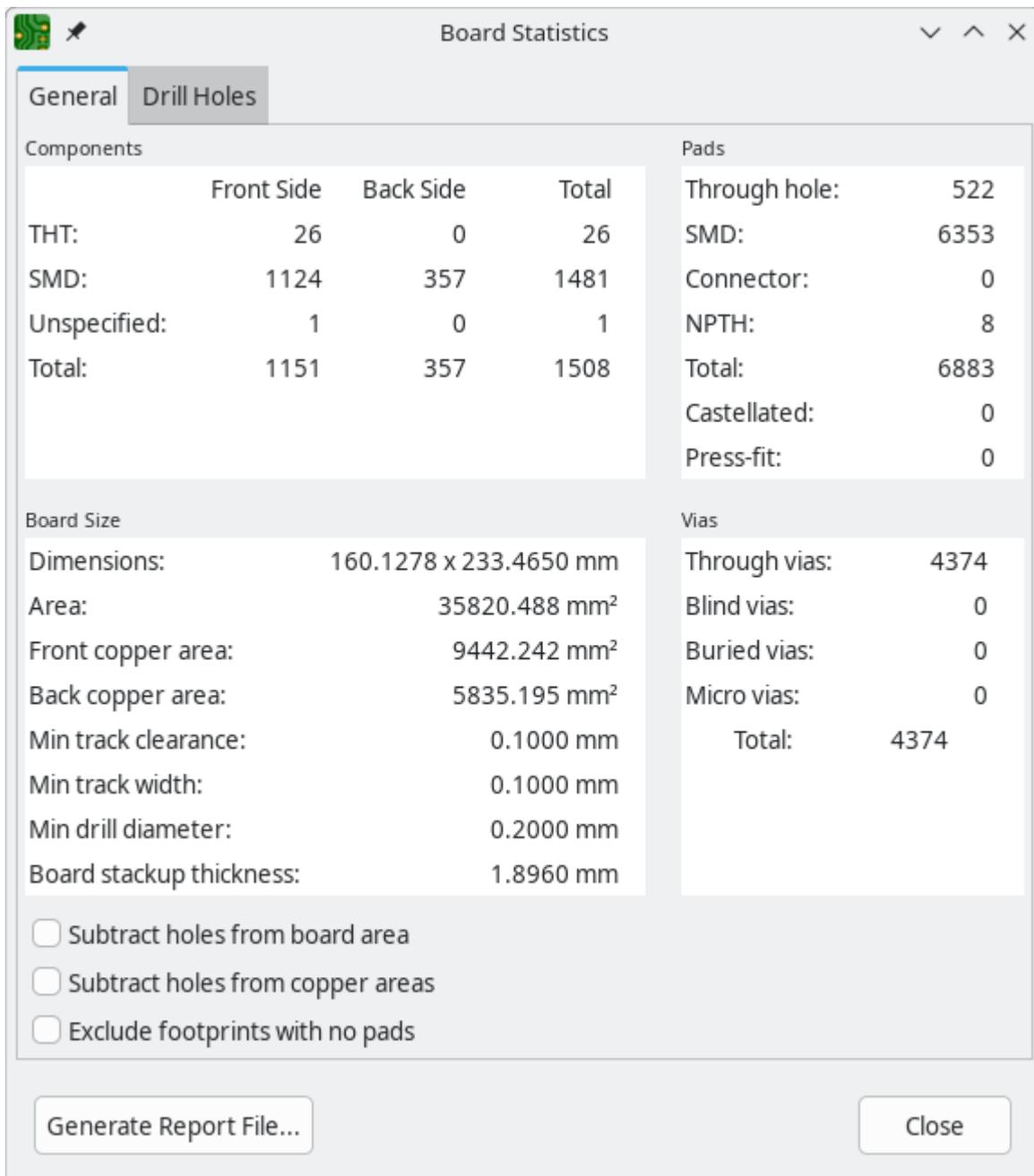
For example, a text item containing `#{DRC_ERROR TODO}Length match tracks` will appear in the board as just the text "Length match tracks", and will generate a DRC error named "TODO" with "Length matches tracks" in the description.

## DRC report file

An DRC report file can be generated and saved by clicking the **Save...** button in the DRC dialog. The file extension for DRC report files is `.rpt`.

### NOTE

DRC reports can also be generated by the `kicad-cli` [tool](#) in either text (`.rpt`) format or JSON.



The **General** tab gives counts of various types of objects:

- Footprints, separated by type (THT, SMD, or unspecified) and board side
- Pads, separated by type (THT, SMD, connector, or NPTH)
- Vias, separated by type (through, blind/buried, or micro)

It also displays the board dimensions, board area, and the area of front and back copper, as well as other manufacturing technology statistics such as minimum track width, minimum track clearance, and minimum drill diameter.

If **Subtract holes from board area** is checked, the reported board area will not include the area of any through holes in the board.

If **Subtract holes from copper areas** is checked, the reported copper areas will not include the area of any through holes in the board.

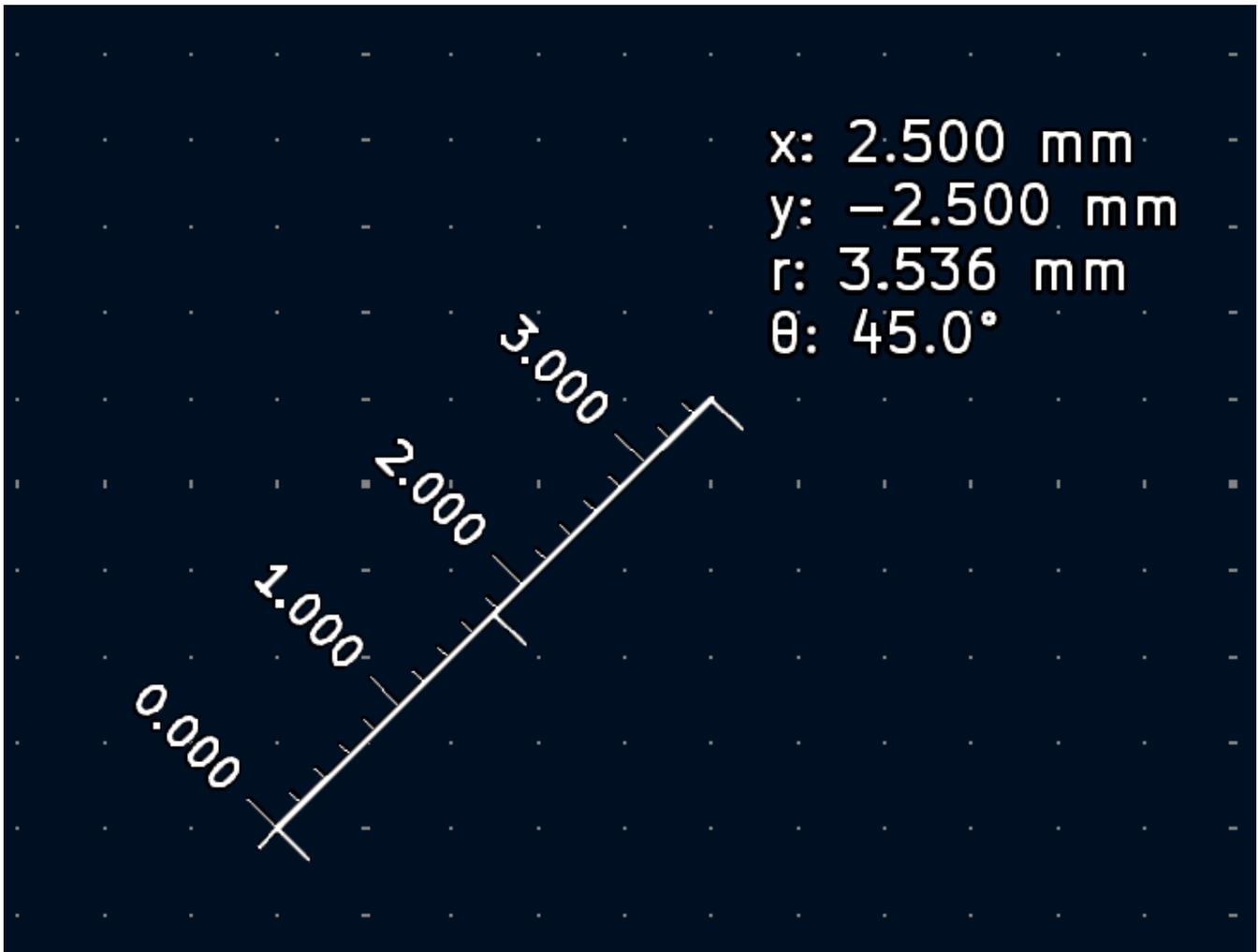
If **Exclude footprints with no pads** is checked, the component counts will exclude footprints that do not contain any pads.

The **Drill Holes** tab lists every unique type of drill hole on the board. Each type of hole is listed with its characteristics (shape, X and Y size, plating, pad or via type, and start and stop layers) and the count of that type of hole.

You can save the board statistics to a file by clicking the **Generate Report File...** button.

## Measurement tool

The measurement tool allows you to make distance and angle measurements between points on the PCB. To activate the tool, click the  icon in the right toolbar, or use the hotkey `Ctrl + Shift + M`. Once the tool is active, click once to set the measurement start point, then click again to finish a measurement.



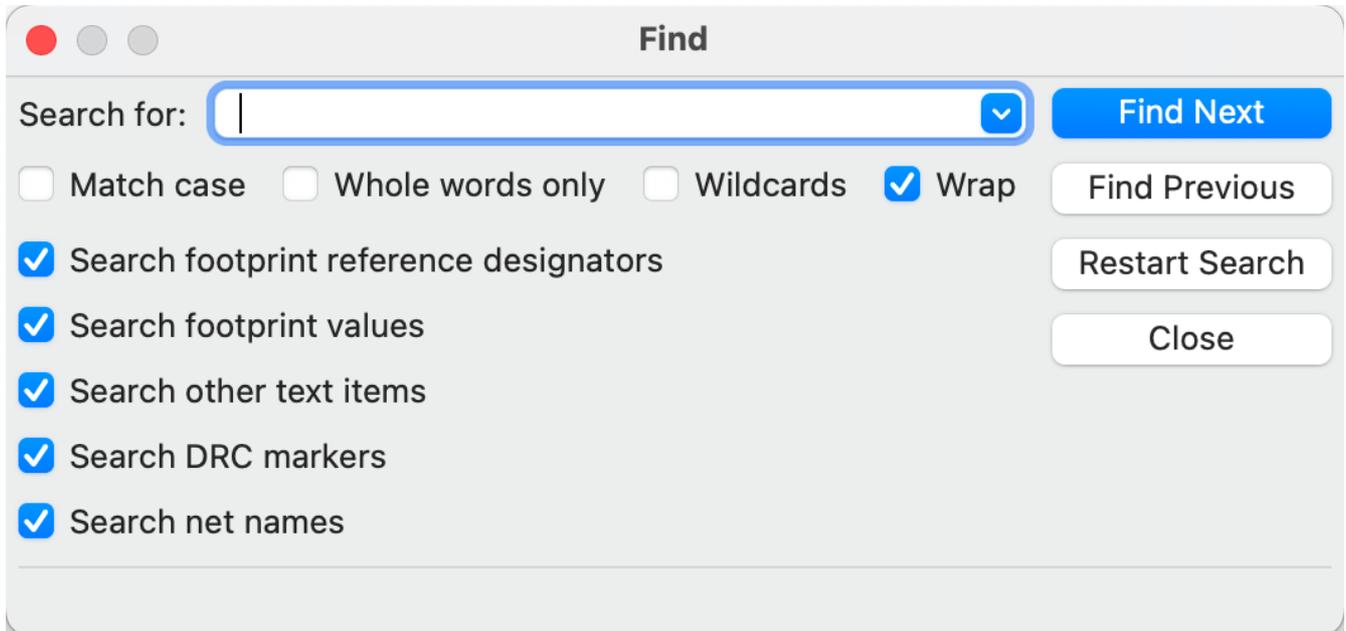
The tool displays the total (radial) distance between the points, the distance in X and Y directions, and the measured angle from horizontal. In other words, both the Cartesian and radial (polar) distances are displayed.

### NOTE

The measurement tool is used for quick measurements that do not need to be displayed permanently. Any measurement you make will only be shown while the tool is active. To create permanent dimensions that will appear in printouts and plots, use the Dimension tools.

## Find tool

The Find tool searches for text in the PCB, including reference designators, footprint fields, and graphic text. When the tool finds a match, the canvas is zoomed and centered on the match and the text is highlighted. Launch the tool using the (  ) button in the top toolbar.



The Find tool has several options:

**Match case:** Selects whether the search is case-sensitive.

**Whole words only:** When selected, the search will only match the search term with complete words in the PCB. When unselected, the search will match if the search term is part of a larger word in the PCB.

**Wildcards:** When selected, wildcards can be used in the search terms. `?` matches any single character, and `*` matches any number of characters. Note that when this option is selected, partial matches are not returned: searching for `abc*` will match the string `abcd`, but searching for `abc` will not.

**Wrap:** When selected, search results will return to the first hit after reaching the last hit.

**Search footprint reference designators:** Selects whether the search should apply to footprint reference designators.

**Search footprint values:** Selects whether the search should apply to footprint value fields.

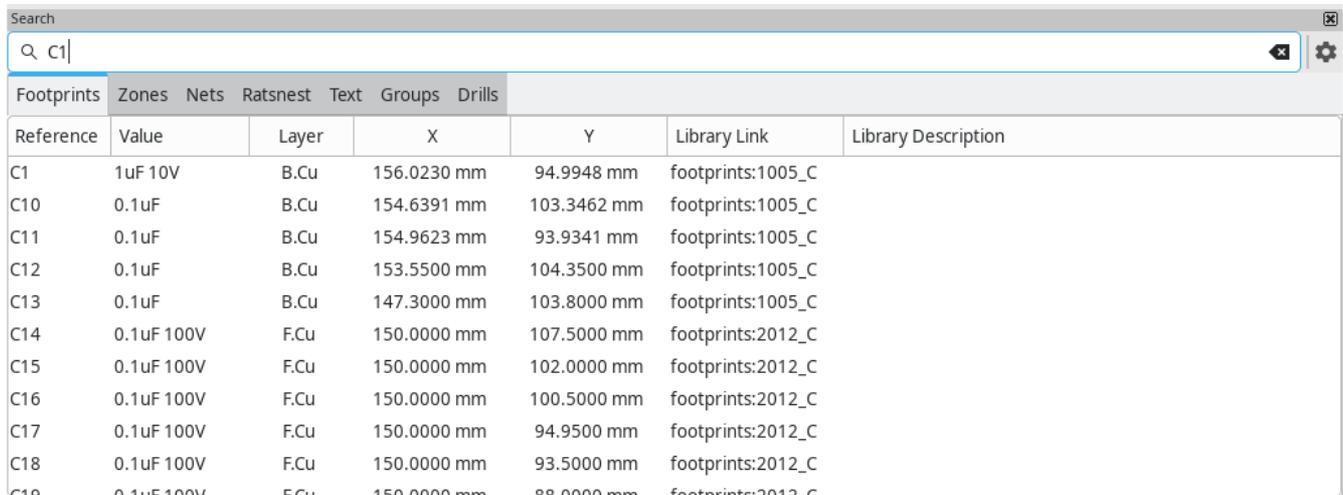
**Search other text items:** Selects whether the search should apply to other text items, including graphical text and footprint fields other than value and reference.

**Search DRC markers:** Selects whether the search should apply to the violation descriptions of DRC markers shown on the board.

**Search net names:** Selects whether the search should apply to the names of nets in the board.

## Search panel

The search panel is a docked panel that lists information about footprints, zones (copper zones and rule areas), nets, ratsnest lines (unrouted segments), text items, groups, and drills from the PCB. Show or hide the search panel with **View** → **Panels** → **Search** or use the **Ctrl + G** shortcut.



Reference	Value	Layer	X	Y	Library Link	Library Description
C1	1uF 10V	B.Cu	156.0230 mm	94.9948 mm	footprints:1005_C	
C10	0.1uF	B.Cu	154.6391 mm	103.3462 mm	footprints:1005_C	
C11	0.1uF	B.Cu	154.9623 mm	93.9341 mm	footprints:1005_C	
C12	0.1uF	B.Cu	153.5500 mm	104.3500 mm	footprints:1005_C	
C13	0.1uF	B.Cu	147.3000 mm	103.8000 mm	footprints:1005_C	
C14	0.1uF 100V	F.Cu	150.0000 mm	107.5000 mm	footprints:2012_C	
C15	0.1uF 100V	F.Cu	150.0000 mm	102.0000 mm	footprints:2012_C	
C16	0.1uF 100V	F.Cu	150.0000 mm	100.5000 mm	footprints:2012_C	
C17	0.1uF 100V	F.Cu	150.0000 mm	94.9500 mm	footprints:2012_C	
C18	0.1uF 100V	F.Cu	150.0000 mm	93.5000 mm	footprints:2012_C	
C19	0.1uF 100V	F.Cu	150.0000 mm	88.0000 mm	footprints:2012_C	

You can optionally filter the list based on a search string. When no filter is used, all items in the design are listed in the corresponding tab. Items are filtered based on their properties:

- Footprints are filtered by the contents of their fields. You can select whether to search hidden fields by enabling the **Search Hidden Fields** option in the  menu. Footprints are also filtered by their metadata (library link, description, and keywords) if **Search Metadata** is enabled in the  menu.
- Zones are filtered by the zone/rule area name.
- Net and ratsnest items are filtered by the net name.
- Text (text, textboxes, and dimensions) is filtered by the text content.
- Groups are filtered by the group name.
- Drills can be filtered by any column.

You can sort the filtered results in ascending or descending order of the value in a particular column by clicking on that column header.

Filters support wildcards: **\*** matches any characters, and **?** matches any single character. You can also use [regular expressions](#), such as `/footprint value/`.

The displayed information depends on the item type:

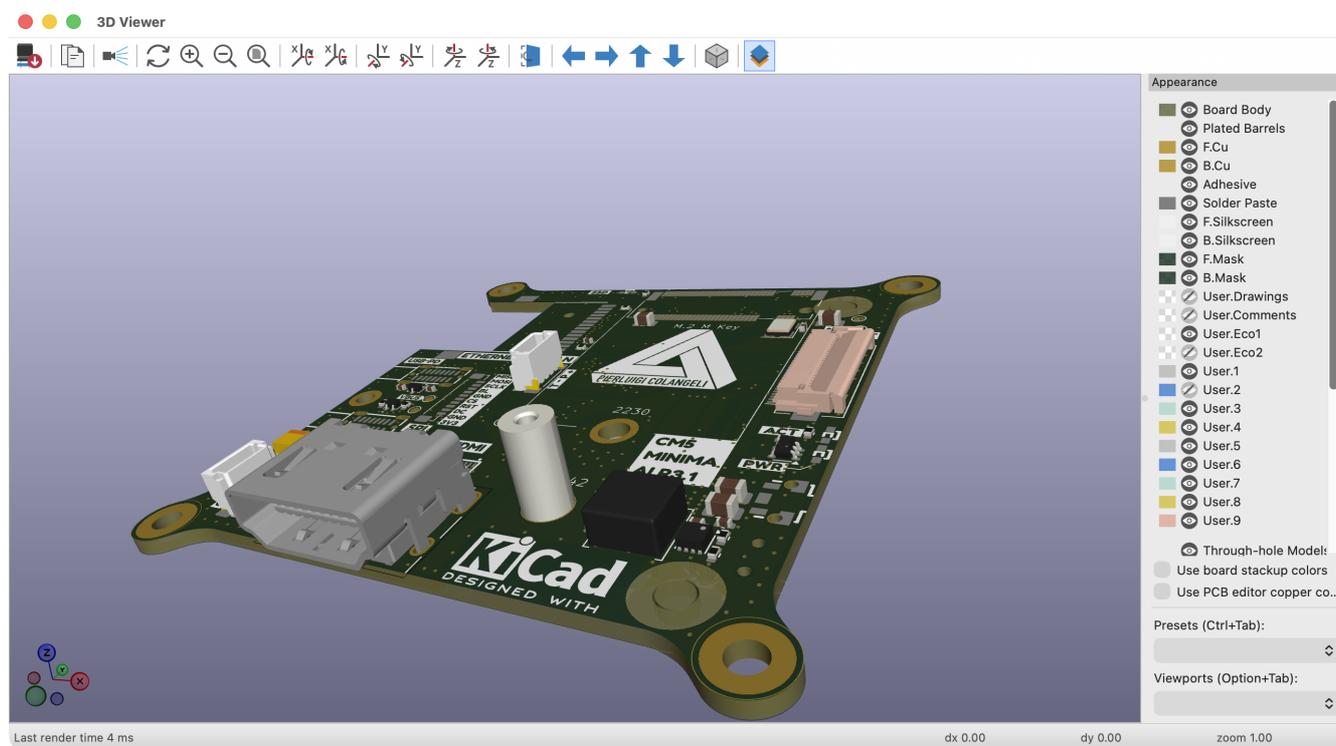
- All items list their name and/or value, layer, and X/Y location as applicable.
- Footprints additionally list their library link (library name and footprint name) and description.
- Zones additionally list their area. For copper zones, this is the filled (copper) area. For rule areas, this is the area within the outline.
- Text additionally lists the type of text object (text, textbox, or dimension).
- Net and ratsnest items additionally list their net name and net class.

Drills, where each item represents a unique type of drilled hole, list the count of each drill type, the shape of the hole, the X and Y size of the hole, the type of plating, whether it is a via or pad, and the start and stop layers.

When you click an item in the search panel, the item is selected in the editing canvas. Depending on what is configured in the  menu, the board editor will also pan and/or zoom to the selected item in the editing canvas. Double-clicking an item in the search panel opens its properties dialog (for net and ratsnest items, the [net classes dialog](#) is opened instead).

## 3D Viewer

The 3D Viewer shows a 3-dimensional view of the board and the components on the board. You can view the board from different perspectives, show or hide different types of components, cross-probe from the PCB Editor to the 3D viewer, and generate raytraced renders of the board. Show the 3D Viewer with **View** → **3D Viewer** or use the  shortcut.



### NOTE

The 3D model for a component will only appear if the 3D model file exists and has been [assigned to the footprint](#).

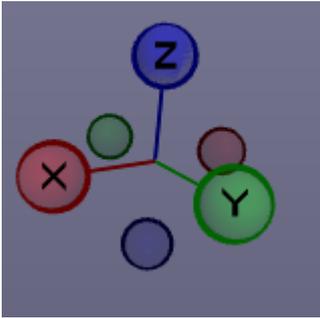
### NOTE

Many footprints in KiCad's standard library do not yet have model files created for them. However, these footprints may contain a path to a 3D model that does not yet exist, in anticipation of the 3D model being created in the future.

## Navigating the 3D view

Dragging with the left mouse button will orbit the 3D view. By default this is the centroid of the board, but the pivot point can be reset to a new point on the board by moving the cursor over the desired point and pressing . Scrolling the mouse wheel will zoom the view in or out. Scrolling while holding  pans the view left and right, and scrolling while holding  pans up and down. Dragging with the middle mouse button also pans the view.

The 3D Navigator is an interactive widget displayed in the 3D Viewer that provides quick access to standard orthogonal views. It consists of six spheres representing the six standard viewing directions: Front, Back, Left, Right, Top, and Bottom. Clicking any sphere will instantly reorient the camera to that viewpoint. The 3D Navigator can be shown or hidden using **Preferences** → **Show 3D Navigator**.



Different sized 3D grids can be set using the **View** → **3D Grid** menu. Bounding boxes for each component can be enabled with **Preferences** → **Show Model Bounding Boxes**.

When the PCB Editor and the 3D Viewer are both open, selecting a footprint in the PCB Editor will also highlight the component in the 3D Viewer. The highlight color is adjustable in **Preferences** → **Preferences...** → **3D Viewer** → **Realtime Renderer** → **Selection Color**.

## Appearance Manager

The Appearance Manager is a panel at the right of the viewer which provides controls to manage the visibility, color, and opacity of different types of objects and board layers in the 3D view.

Each layer or type of object in the list can be individually shown or hidden by clicking its corresponding visibility icon. PCB layers can have their colors customized; double-click on the color swatch next to the item type to edit the item's color and opacity. To use the colors selected in the Board Setup dialog's Physical Stackup editor, enable the **use board stackup colors** option. If you enable the **use PCB editor copper colors** option, copper layers in the 3D viewer will use the colors configured in the PCB editor canvas.

You can save an appearance configuration as a preset, or load a configuration from a preset, using the **Preset** selector at the bottom. The **Ctrl** + **Tab** hotkey cycles through presets; press **Tab** repeatedly while holding **Ctrl** to cycle through multiple presets. Several built-in presets are available: "Follow PCB Editor" matches the visibility settings in the PCB editor, "Follow PCB Plot Settings" matches the visibility settings selected in the Plot dialog, and "legacy colors" matches the default 3D Viewer color settings from older versions of KiCad.

Finally, you can save a viewport for later retrieval using the **Viewports** selector at the bottom. You can quickly cycle between saved viewports using **Shift** + **Tab**; pressing **Tab** repeatedly while holding **Shift** will cycle through multiple viewports.

## Generating images with the 3D Viewer

The current 3D view can be saved to an image file with **File** → **Export Image...** Before saving, you can choose the output image size and resolution. The current view can also be copied to the clipboard using the  button, or **Edit** → **Copy 3D Image to Clipboard**.

The 3D Viewer has a raytracing rendering mode which displays the board using a more physically accurate rendering model than the default rendering mode. Raytracing is slower than the default rendering mode,

but it can be used when the most visually attractive results are desired. Raytracing mode is enabled with the  button, or with **Preferences** → **Raytracing**. The 3D grid and selection highlights are not shown in raytracing mode.

Colors and other rendering options, for both raytraced and non-raytraced modes, can be adjusted in **Preferences** → **Preferences...** → **3D Viewer**.

## 3D viewer controls

Many viewing options are controlled with the top toolbar.

**NOTE** | You can edit the toolbars' contents in the **Toolbar** page of the 3D Viewer Preferences.

	Reload the 3D model
	Copy 3D image to clipboard
	Render current view using raytracing
	Redraw
	Zoom in
	Zoom out
	Fit drawing in display area
	Rotate X clockwise
	Rotate X counterclockwise
	Rotate Y clockwise
	Rotate Y counterclockwise
	Rotate Z clockwise
	Rotate Z counterclockwise
	Flip board view
	Pan board left
	Pan board right
	Pan board up
	Pan board down
	Enable/disable orthographic projection
	Show/hide the Appearance Manager

## Net inspector

The Net Inspector is a docked panel that allows you to view statistics about all the nets in a board. It also lets you add, remove, and rename nets. To open the inspector, click the  icon at the top of the Nets section of the Appearance panel, or select **View** → **Panels** → **Net Inspector**.

Name	Netclass	Total Length	Via Count	Via Length	Track Length	Die Length	Pad Count	Top_layer	B.Cu	In1.Cu	In2.Cu
Netclass: POWER		833.2507 mm	90	115.8750 mm	717.3757 mm	0.0000 mm	234	562.7292 mm	70.7131 mm	57.6646	26.2689 mm
GND	POWER	63.9114 mm	3	1.5450 mm	62.3664 mm	0.0000 mm	7	18.2342 mm	2.5400 mm	41.5922	0.0000 mm
GND	POWER	404.2873 mm	44	57.6800 mm	346.6073 mm	0.0000 mm	128	291.0313 mm	39.5035 mm	16.0724	0.0000 mm
+3.3V	POWER	365.0521 mm	43	56.6500 mm	308.4021 mm	0.0000 mm	99	253.4636 mm	28.6696 mm	0.0000 n	26.2689 mm
Netclass: Default		8891.5549 mm	163	196.7300 mm	8694.8249 mm	0.0000 mm	513	2655.2485 mm	2899.2693 mm	2591.42	548.8836 mm

Double-clicking a net in the list will [highlight](#) that net on the board. You can also highlight a net by right clicking it and selecting **Highlight Selected Net**. If multiple nets are selected, this lets you highlight all of them at once. You can remove the net highlighting by right clicking the net's row in the Net Inspector and selecting **Clear Net Highlighting**, in addition to the usual ways of [removing net highlighting](#).

Clicking a column title allows you to sort the list of nets by that column. The Filter box lets you limit the listed nets to those that match the filter string. By default, the filter matches against both net names and net class names, but you can filter by just one or the other by selecting or deselecting **Filter by Net Name** or **Filter by Netclass** under the  menu.

By default, nets with no connections and nets with no pads are not shown. You can choose to show them by selecting **Show Unconnected Nets** and **Show Zero Pad Nets** under the  menu.

The Net Inspector shows the following statistics for each net:

- **Pad Count** is the number of pads with that net, counting both surface mount and through hole pads.
- **Via Count** is the number of vias with that net.
- **Via Length** is the sum total length of all vias with that net. The full height of each via is always counted, even if the connections to the via are such that the full via height is not electrically used. In other words, Via Length is equal to Via Count multiplied by the stackup height of the board.
- **Track Length** is the total length of all track segments in a net, not accounting for topology. For example, in a branching net structure all branches are included in the total length. The track length is also reported per copper layer.
- **Die Length** is the total of all Pad to Die Length values set for pads on the net.

Each column can be shown or hidden in the  → **Show / Hide Columns** menu. You can save the Net Inspector statistics to a CSV file by clicking  → **Save Net Inspector Report**. The generated report includes all nets and columns, even if they are currently filtered or hidden in the Net Inspector.

## Grouping nets

You can group nets in the Net Inspector to organize them and view them more easily. Each group displays the total statistics for all its members, as if the group were a single net. For example, if you have a signal with a series resistor breaking the signal into two nets, you could create a group that contains both of these nets. This would allow you to analyze the total length of both nets, rather than each individually.

You can group nets by their net class by clicking  → **Group by netclass**. Alternatively, you can create custom groups based on net name patterns. To create a new custom group, click  → **Add Custom Group**. Any nets that contain the specified pattern in their name will be shown as part of the group and not shown outside of the group. For example, the pattern `CAN` matches the nets `CAN_RX` and `CAN_TX`. Patterns are not case sensitive.

The pattern can also use regular expressions to match nets if the pattern is surrounded in slashes. For example, the pattern `/^AN/` matches nets `AN0`, `AN1`, etc., but not `CAN`.

To remove a group and release its members back into the full list of nets, click  → **Remove Selected Custom Group**. This action is also available in the right click menu. To remove all groups at once, click  → **Remove All Custom Groups**.

## Editing nets

The Net Inspector allows you to create new nets in the board and remove or rename existing nets. To create a new net, right click in the Net Inspector and select **Add Net**, then provide a name for the new net. To delete a net, right click it in the list of nets and choose **Delete Selected Net**. If multiple nets are selected, they will all be deleted. To rename a net, right click it and choose **Rename Selected Net**, then provide a new name.

### NOTE

Nets are usually not edited in the board. Instead, it is recommended to define nets in the schematic. Nets are typically managed in the board by creating or modifying a schematic and then using the [Update PCB From Schematic](#) tool to update the nets in the board based on the schematic design. The Net Inspector can be used to manage nets in alternate workflows that do not use a schematic.

### NOTE

Nets that are modified in the Board Editor will not effect the schematic until the [schematic is updated from the PCB](#) through the back-annotation process.

## Differences between Net Inspector and Length Tuner

The Net Inspector may report different net lengths than the [length tuner](#), because the two tools have different purposes and calculate track/net lengths differently. In short, the Net Inspector sums up the total length of each track segment and via on a net, while the length tuner calculates the effective electrical length of a path between two points on a net. The specific differences are as follows:

- The Net Inspector reports track length as a simple sum of the length of each track segment on a net. The length tuner calculates an effective electrical length of a net, which includes optimizing paths through pads to calculate the shortest possible path.
- If a routed net has a branching topology, the Net Inspector total includes the length of each branch in the total. The length tuner calculates a point-to-point length; if there are any branches, the length tuner will stop at the closest branch and report the length up to the branch.
- The Net Inspector always includes the effective via height in its via length and total length calculations. If a via connects to tracks on both the top and bottom layers, the full via height is included in the length calculation. Otherwise, only the stackup height between the connected layers is included. The length tuner calculates effective via height in the same way as the Net Inspector, but via height is only included in the length calculation when the **use stackup height** setting is enabled [board constraint settings](#). If the setting is disabled, the length tuner will not include vias in its calculations at all.

# Importing boards from other EDA tools

KiCad can import PCB designs from a wide range of third-party EDA tools. This allows you to migrate existing designs into KiCad without having to recreate them from scratch.

Imported boards are converted to KiCad's native format. After import, the board can be edited, have its design rules adjusted, and be exported using any of KiCad's output capabilities.

## NOTE

Importing a board creates a new KiCad board from the foreign file. It does not merge the foreign board into an existing KiCad design. After import, save the board to create a KiCad `.kicad_pcb` file.

## Supported import formats

KiCad supports importing boards from the following EDA tools and file formats:

Source EDA Tool	File Extension(s)	Layer Mapping	Library Import
Altium Designer	.PcbDoc	Yes	Yes
Altium Circuit Maker	.CMPcbDoc	Yes	No
Altium Circuit Studio	.CSPcbDoc	Yes	No
Cadence Allegro	.brd	Yes	No
CADSTAR PCB Archive	.cpa	Yes	Yes
Eagle (Autodesk)	.brd (XML)	Yes	Yes
EasyEDA / JLCEDA Standard	.json, .zip	No	Yes
EasyEDA / JLCEDA Professional	.epro, .zip	No	Yes
Fabmaster	.txt, .fab	No	No
gEDA / Lepton EDA	.pcb	No	Yes
P-CAD 200x	.pcb (ASCII)	No	No
PADS (ASCII)	.asc	Yes	No
Solidworks PCB	.SWPcbDoc	Yes	No

In addition, KiCad can import Spectra session files ( `.ses` ) to bring autorouter results back into an existing board. This is a specialized workflow described [below](#).

## How to import a board

To import a board from a supported foreign format, use **File** → **Import** → **Non-KiCad Board File....** In the file dialog, change the file type dropdown to the desired import format. Select the foreign board file and

click **Open**.

KiCad will read the foreign file and convert it to a KiCad board. Depending on the format, a layer mapping dialog may appear to let you control how the source layers are mapped to KiCad layers.

**TIP**

Some formats such as Eagle and Allegro share the `.brd` file extension. KiCad will automatically detect the correct format based on the file contents when you select the appropriate file type filter in the open dialog.

## Layer mapping

Many import formats include a layer mapping step. When the importer encounters layers in the foreign file, it presents a dialog that lets you choose the corresponding KiCad layer for each source layer.

The dialog shows each source layer name alongside a dropdown where you can select the target KiCad layer. The importer provides a best-guess automatic mapping, but you should review it carefully. Key considerations:

- **Copper layers:** Verify that the source copper layers map to the correct KiCad copper layers, especially for inner layers on multi-layer boards.
- **Silkscreen, solder mask, and paste layers:** These should map to the corresponding front or back layer in KiCad.
- **Mechanical and documentation layers:** Map these to the appropriate KiCad User layers or leave them unmapped if the content is not needed.
- **Unmapped layers:** Setting a source layer to *Unassigned* causes that layer's content to be skipped during import.

The formats that support layer mapping are: Altium Designer, Altium Circuit Maker, Altium Circuit Studio, Cadence Allegro, CADSTAR, Eagle, PADS, and Solidworks PCB.

## Post-import cleanup

After importing a board from another EDA tool, review the result and perform any necessary cleanup. The following steps are recommended:

1. **Run the Design Rule Check (DRC):** Select **Inspect** → **Design Rules Checker** to identify any issues introduced during import. Foreign designs may use rules or features that do not have exact equivalents in KiCad.
2. **Review board setup:** Open **File** → **Board Setup** and verify the board stackup, design rules, and net classes. Import plugins translate rules on a best-effort basis, but some parameters may need manual adjustment.
3. **Check footprints:** Imported footprints are embedded directly in the board file. Consider replacing them with footprints from the KiCad library or your own libraries to take advantage of KiCad's 3D models and updated pad shapes.
4. **Inspect zones:** Zone fills may need to be recalculated after import. Select **Edit** → **Fill All Zones** () to refill all copper zones.
- 5.

**Verify net assignments:** Confirm that net names and net classes have been imported correctly by inspecting the **Nets** inspector (**Inspect** → **Board Statistics**) or by checking net properties of individual tracks and pads.

6. **Save as a KiCad project:** Use **File** → **Save As...** to save the imported board as a `.kicad_pcb` file within a KiCad project directory.

**TIP**

Some importers produce messages in the message panel at the bottom of the editor window. Review these messages for warnings about unsupported features or items that could not be imported.

## Format-specific notes

### Altium Designer

Altium Designer is one of the most widely used commercial EDA tools, and the KiCad importer provides comprehensive support for `.PcbDoc` files.

#### Supported features:

- Board outline and cutouts
- Copper layers (up to 32 layers), silkscreen, solder mask, paste, and mechanical layers
- Tracks, arcs, vias (including blind and buried vias)
- Pads (through-hole, SMD, and custom shapes)
- Copper fills and polygon pours
- Component placement and footprint geometry
- Net information
- Design rules (partial)
- Embedded footprint libraries (cached footprints are available after import)

#### Known limitations:

- Embedded OLE objects (such as images or documents) are not imported.
- Some complex design rules may not have direct KiCad equivalents and will be simplified.
- Differential pair rules are imported on a best-effort basis.
- Altium-specific features like Situs autorouter data or embedded simulation configurations are not supported.

In addition to board import, KiCad can import Altium footprint libraries (`.PcbLib`) and integrated libraries (`.IntLib`) directly through the footprint library manager.

### Altium Circuit Maker

Altium Circuit Maker files (`.CMPcbDoc`) use the same underlying format as Altium Designer with minor variations. The importer handles these files the same way as Altium Designer boards. Layer mapping is supported.

## Altium Circuit Studio

Altium Circuit Studio files ( `.CSPcbDoc` ) are supported similarly to Altium Designer. The importer applies layer mapping and converts the board contents using the same parsing engine.

## Cadence Allegro

KiCad can import Cadence Allegro binary `.brd` files directly.

### Supported features:

- Board outline
- Copper layers and signal routing
- Padstacks and vias
- Component placement
- Net information
- Layer mapping with automatic suggestions

### Known limitations:

- Allegro design rules and constraints are not fully imported.
- Flex-circuit layer stackup features are not supported.
- Some Allegro-specific pad shapes may be approximated.

#### NOTE

Allegro `.brd` files use a binary format that is distinct from Eagle `.brd` files (which are XML). KiCad identifies the correct format automatically based on file content.

## CADSTAR PCB Archive

CADSTAR PCB Archive files ( `.cpa` ) from Zuken's CADSTAR tool can be imported into KiCad.

### Supported features:

- Board outline and templates
- Full copper layer stackup
- Tracks, vias, and copper pours
- Component placement with footprint geometry
- Net and net class information
- Layer mapping
- Footprint library import (footprints from the `.cpa` archive can be loaded through the footprint library manager)

### Known limitations:

- CADSTAR-specific template areas may be approximated or simplified.

- Some advanced constraint features may not be converted.

## Eagle (Autodesk)

KiCad has mature support for importing Eagle `.brd` files in XML format (Eagle version 6.x and later).

### Supported features:

- Board outline
- All copper layers and signal layers
- Tracks, vias, and polygons
- SMD and through-hole pads with design rule-based annular ring sizing
- Component placement and package (footprint) geometry
- Net list and net class definitions
- Design rules (converted to KiCad equivalents where possible)
- Layer mapping with intelligent defaults
- Embedded footprint libraries

### Known limitations:

- Eagle's ULP scripts and CAM jobs are not imported.
- Some Eagle-specific DRC rules (e.g., per-signal clearances) may be simplified.
- Binary Eagle files (created before Eagle version 6.x) are not supported; these must first be opened and re-saved as XML in Eagle.

In addition to board import, KiCad can import Eagle footprint libraries (`.lbr`) through the footprint library manager.

#### TIP

When migrating from Eagle, you can also import Eagle schematic files (`.sch`) in the KiCad Schematic Editor to bring over the full design.

## EasyEDA / JLCEDA Standard

KiCad can import board files from EasyEDA (also known as JLCEDA Standard edition), the free web-based EDA tool from JLCPCB.

**Accepted file types:** `.json` and `.zip`

To obtain files for import, export your design from EasyEDA using its **Export EasyEDA** option, which produces a JSON file or a ZIP archive containing the board data.

### Supported features:

- Board outline
- Copper layers and routing
- Pads and vias

- Component placement
- Net information
- Embedded footprint libraries

**Known limitations:**

- Some custom pad shapes may not convert exactly.
- Panelization data is not imported.

## EasyEDA / JLCEDA Professional

KiCad can import project files from EasyEDA Professional (also known as JLCEDA Professional), the desktop version of JLCEDA.

**Accepted file types:** `.epro` and `.zip`

For multi-board projects, KiCad will present a project chooser dialog to let you select which board within the project to import.

**Supported features:**

- Board outline
- Copper layers and routing
- Pads and vias
- Component placement
- Net information
- Footprint libraries ( `.elibrz` files can also be imported through the footprint library manager)

**Known limitations:**

- Some advanced features specific to EasyEDA Pro may not have KiCad equivalents.

## Fabmaster

KiCad can import Valor/Mentor Fabmaster files ( `.txt` , `.fab` ). These files contain fabrication data including board geometry, component placements, and layer information.

**Known limitations:**

- Fabmaster files contain manufacturing-oriented data and may not include full design intent (e.g., schematic connectivity).
- Layer mapping is not interactive; layers are assigned automatically.
- Net information may be incomplete depending on what was exported from the source tool.

## gEDA / Lepton EDA

KiCad can import board files from gEDA PCB and Lepton EDA, the open-source EDA suite.

**Accepted file types:** .pcb

**Supported features:**

- Board layers and routing
- Vias
- Component (element) placement
- Net list information
- Footprint libraries (gEDA .fp footprint files can be imported through the footprint library manager)

**Known limitations:**

- gEDA's layer model is simpler than KiCad's; some layer assignments may need adjustment after import.

## P-CAD

KiCad can import P-CAD 200x ASCII board files ( .pcb ).

P-CAD was a legacy EDA tool that was acquired by Altium and eventually discontinued. If you have P-CAD designs, you can bring them into KiCad using this importer.

**Known limitations:**

- Only the ASCII .pcb format is supported; the binary format is not.
- P-CAD design rules are not imported.
- Some P-CAD-specific features may not have KiCad equivalents.

## PADS

KiCad supports importing boards from Siemens (Mentor Graphics) PADS in both ASCII and binary formats.

**PADS ASCII ( .asc ):** This is the text-based export format from PADS Layout. To generate this file, use the **File** → **Export** function in PADS Layout and choose the ASCII format.

**Supported features:**

- Board outline
- Board setup (layer stackup)
- Copper layers and routing (tracks and vias)
- Pads and footprints
- Copper pours and zones
- Net information
- Text and dimensions
- Keepout areas
- Layer mapping with intelligent defaults

### Known limitations:

- Some PADS-specific pad shapes or padstack definitions may be approximated.
- Design rules and constraints from PADS are converted on a best-effort basis.
- Reuse block definitions may not be fully preserved.

## Solidworks PCB

KiCad can import Solidworks PCB files ( `.SWPcbDoc` ). Solidworks PCB (formerly PCBWorks) uses a format derived from Altium, and the importer shares much of its code with the Altium Designer importer.

Layer mapping is supported. The same considerations that apply to Altium Designer imports generally apply to Solidworks PCB imports as well.

## Importing Specctra session files

The Specctra DSN/SES workflow is used with external autorouters. Unlike the board importers described above, the Specctra session import does not create a new board. Instead, it applies routing results from an autorouter back onto an existing KiCad board.

The typical workflow is:

1. Export the board to Specctra DSN format: select **File** → **Export** → **Specctra DSN...**
2. Run the external autorouter on the `.dsn` file to produce a session file ( `.ses` ).
3. Import the session file back into KiCad: select **File** → **Import** → **Specctra Session....**

KiCad will read the `.ses` file and add the routed tracks and vias to the board. The board's footprints, nets, and board outline must match the original export for the import to succeed.

### NOTE

The Specctra session import only adds routing; it does not modify footprints, net lists, or the board outline.

# Generating outputs

KiCad can generate and export files in a number of different formats useful for manufacturing PCBs and interfacing with external software. This functionality is available in the File menu in a few different sections.

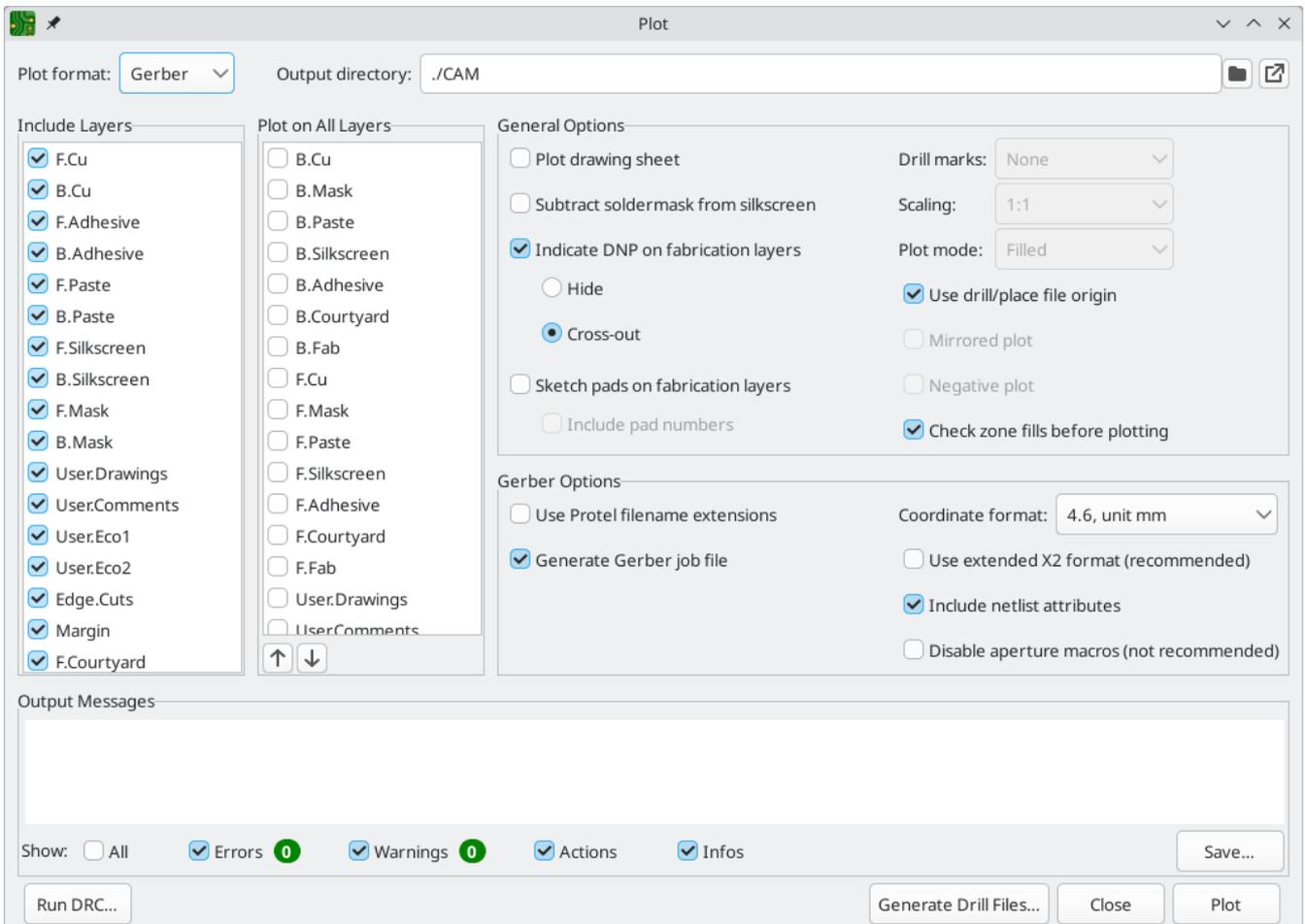
- The **Fabrication Outputs** section contains the most common operations needed to prepare a PCB for fabrication.
- The **Export** section contains tools for generating files that can be read by external software.
- The **Plot** function allows you to export 2D line drawings of the PCB in various formats.
- The **Print** function allows you to send a view of the PCB to a 2D printer.

## Plotting (Gerber / PostScript / SVG / DXF / PDF)

KiCad uses Gerber files as its primary plotting format for PCB manufacturing. To create Gerber files, select **File** → **Fabrication Outputs** → **Gerbers (.gbr)...** The Plot dialog will open, allowing you to configure and generate Gerber files.

In addition to Gerber files, the Plot dialog is also used to create PostScript, SVG, DXF, and PDF outputs. You can also open the Plot dialog with **File** → **Plot...**, or by clicking the  button in the top toolbar. You can select the output format with the **Plot format** dropdown.

Most plotting options are common to all of the plotted output formats, but there are also some options that are specific to each format.



The **Plot** button generates output files according to the selected options. Messages from the plotting process are shown in the Output Messages panel, and can be filtered by the checkboxes.

The **Generate Drill Files...** button opens the [Generate Drill Files dialog](#). **Run DRC...** opens the [Design Rules Checker](#).

## Plotting options

- **Include Layers:** Check that every layer used on your board is enabled in the list. Disabled layers will not be plotted.
- **Plot on All Layers:** Selected layers will be included in the plot for each layer selected in the **include layers** list. The additional layers are plotted on top of the base layer. You can reorder these layers using the arrow buttons at the bottom; items that are lower in the list are plotted after (on top of) items that are higher in the list.
- **Output directory:** Specify the location to save plotted files. If this is a relative path, it is created relative to the project directory. Use the  button to open the output directory in a file browser.
- **Plot drawing sheet:** If enabled, the drawing sheet border and title block will be plotted on each layer. This should usually be disabled when plotting Gerber files.
- **Subtract soldermask from silkscreen:** When enabled, silkscreen will be automatically removed from board areas that aren't covered by soldermask.
- **Indicate DNP on fabrication layers:** If enabled, fabrication layers ( F.Fab and B.Fab ) will indicate when a footprint has the DNP (Do Not Populate) attribute set. DNP footprints are either not plotted on the fabrication layers (**Hide**) or are plotted with an X drawn through them on the front and back fabrication layer (**Cross-out**).
- **Sketch pads on fabrication layers:** If enabled, the outlines of footprint pads will be drawn on fabrication layers ( F.Fab or B.Fab ). If **Include pad numbers** is enabled, pad numbers will be drawn as well.
- **Drill marks:** For plot formats other than Gerber, marks may be plotted at the location of all drilled holes. Drill marks may be created at the actual size (diameter) of the finished hole, or at a smaller size.
- **Scaling:** For plot formats that support scaling other than 1:1, the plot scale may be set. The Auto scaling setting will scale the plot to fit the specified page size.
- **Use drill/place file origin:** When enabled, the coordinate origin for plotted files will be the drill/place file origin set in the board editor. When disabled, the coordinate origin will be the absolute origin (top left corner of the worksheet).
- **Mirrored plot:** For some plot formats, the output may be mirrored horizontally when this option is set.
- **Negative plot:** For some plot formats, the output may be set to negative mode. In this mode, shapes will be drawn for the empty space inside the board outline, and empty space will be left where objects are present in the PCB.
- **Check zone fills before plotting:** When enabled, zone fills will be checked (and refilled if outdated) before generating outputs. Plot outputs may be incorrect if this option is disabled!

## NOTE

Versions of KiCad before 9.0 had a global control for tenting vias while plotting. Since KiCad 9.0, via tenting is globally controlled in [Board Setup](#), and can be overridden in the [properties dialog](#) for each via.

## Gerber options

- **Use Protel filename extensions:** When enabled, the plotted Gerber files will be named with file extensions based on Protel ( `.GBL` , `.GTL` , etc). When disabled, the files will have the `.gbr` extension.
- **Generate Gerber job file:** When enabled, a Gerber job file ( `.gbrjob` ) will be generated along with any Gerber files. The Gerber job file is an extension to the Gerber format that includes information about the PCB stackup, materials, and finish. More information about Gerber job files is available at [the Ucamco website](#).
- **Coordinate format:** Configure how coordinates will be stored in the plotted Gerber files. Check with your manufacturer for their recommended setting for this option.
- **Use extended X2 format:** When enabled, the plotted Gerber files will use the X2 format, which includes information about the netlist and other extended attributes. This format may not be compatible with older CAM software used by some manufacturers.
- **Include netlist attributes:** When enabled, the plotted Gerber files will include netlist information that can be used for checking the design in CAM software. When X2 format mode is disabled, this information is included as comments in the Gerber files.
- **Disable aperture macros:** When enabled, all shapes will be plotted as primitives rather than by using aperture macros. This setting should only be used for compatibility with old or buggy CAM software when requested by your manufacturer.

## PostScript options

- **Scale factor:** Controls how coordinates in the board file will be scaled to coordinates in the PostScript file. Using a different value for X and Y scale factors will result in a stretched / distorted output. These factors may be used to correct for scaling in the PostScript output device to achieve an exact-scale output.
- **Track width correction:** A global factor that is added (or subtracted, if negative) from the size of tracks, vias, and pads when plotting a PostScript file. This factor may be used to correct for errors in the PostScript output device to achieve an exact-scale output.
- **Force A4 output:** When enabled, the generated PostScript file will be A4 size even if the KiCad board file is a different size.

## SVG options

- **Precision:** Controls how many significant digits will be used to store coordinates.
- **Output mode:** Controls whether the generated SVG file is in color or black and white.
- **Fit page to board:** When enabled, the generated SVG will have the same size as the board outline.

## DXF options

- **Plot graphic items using their contours:** Graphic shapes in DXF files have no width. This option controls how graphic shapes with a width (thickness) in a KiCad board are plotted to a DXF file. When

this option is enabled, the outer contour of the shape will be plotted. When this option is disabled, the centerline of the shape will be plotted (and the shape's thickness will not be visible in the resulting DXF file).

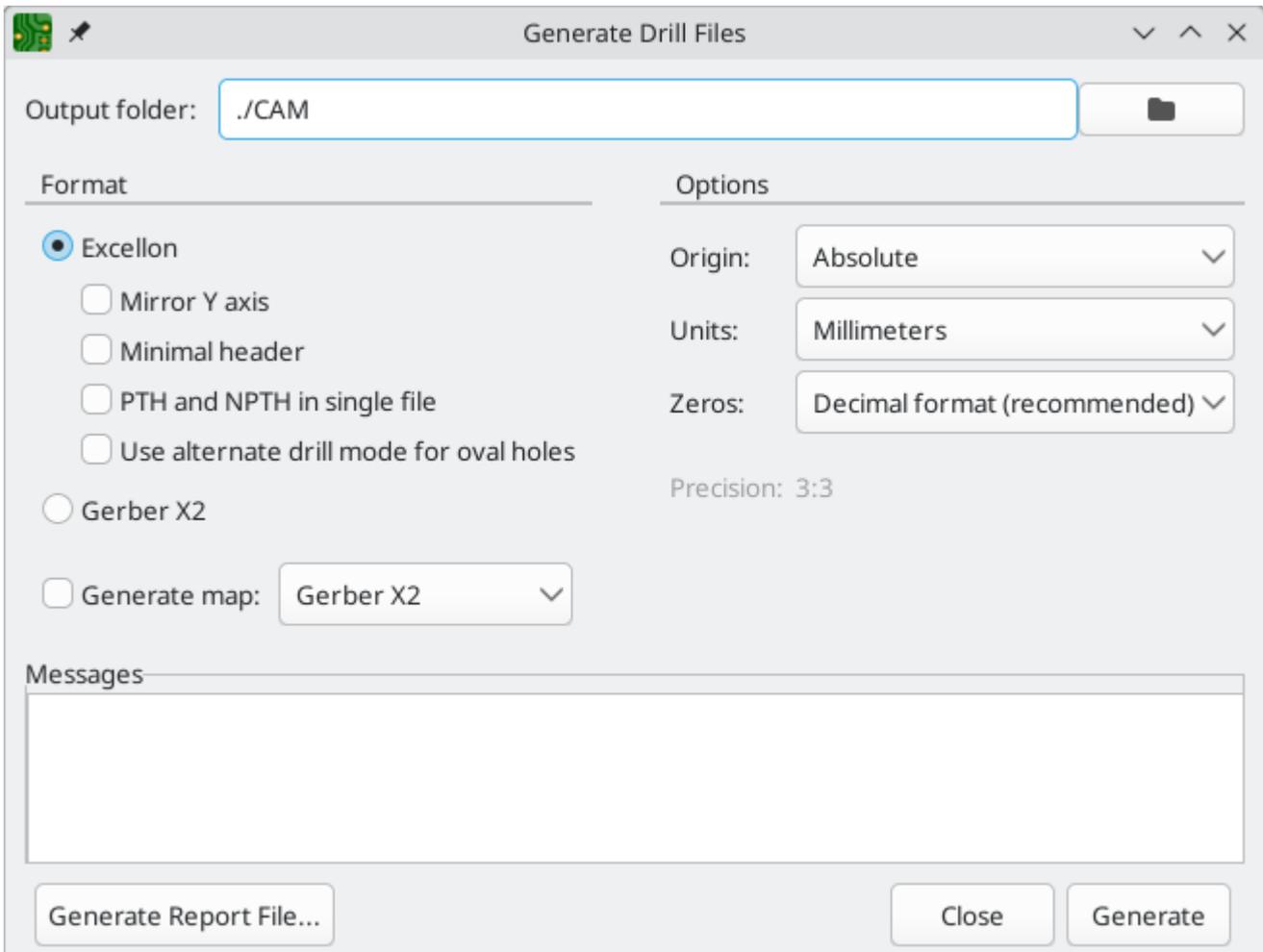
- **Use KiCad font to plot text:** When enabled, text in the KiCad design will be plotted as graphic shapes using the KiCad font. When disabled, text will be plotted as DXF text objects, which will use a different font and will not appear in exactly the same position and size as shown in the KiCad board editor.
- **Single document:** When enabled, all selected layers will be plotted in a single DXF file, with each PCB layer plotted as a separate DXF layer.
- **Export units:** Controls the units that will be used in the DXF file. Since the DXF format has no specified units system, you must export using the same units setting that you want to use for importing into other software.

## PDF options

- **Output mode:** Controls whether the generated PDF file is in color or black and white.
- **Generate property popups for front footprints:** When enabled, interactive popups will be added to the generated PDF containing part information for each footprint on the front of the board.
- **Generate property popups for back footprints:** When enabled, interactive popups will be added to the generated PDF containing part information for each footprint on the back of the board. For details, see the [Schematic Editor documentation](#).
- **Generate metadata from AUTHOR and SUBJECT variables:** Sets the Author and Subject PDF document properties for the generated PDF based on the `AUTHOR` and `SUBJECT` [project text variables](#), if you have defined them.
- **Single document:** When enabled, each layers will be plotted as an individual sheet within a single PDF document. When disabled, each layer will be plotted as a separate PDF file.
- **Background color:** Sets the background color for the PDF plot. Background color is not available when the output mode is black and white.

## Drill files

KiCad can generate CNC drilling files required by most PCB manufacturing processes in either Excellon or Gerber X2 format. KiCad can also generate a drill map: a graphical plot of the board showing drill locations. To open the dialog, select the **File** → **Fabrication Outputs** → **Drill Files (.drl)...**, or click the **Generate Drill Files...** button in the [Plot dialog](#).



There are several options for generating drill files.

- **Output folder:** Choose the folder to save generated drill and map files to. If a relative path is entered, it will be relative to the project directory.
- **Drill file format:** Choose whether to generate Excellon drill files (required by most PCB manufacturers) or Gerber X2 files.
- **Mirror Y axis:** For Excellon files, choose whether or not to mirror the Y-axis coordinate. This option should in general not be used when having PCBs manufactured by a third party, and is provided for convenience for users who are making PCBs themselves.
- **Minimal header:** For Excellon files, choose whether to output a minimal header rather than a full file header. This option should not be enabled unless requested by your manufacturer.
- **PTH and NPTH in single file:** By default, plated holes and non-plated holes will be generated in two different Excellon files. With this option enabled, both will be merged into a single file. This option should not be enabled unless requested by your manufacturer.
- **Use alternate drill mode for oval holes:** Controls how oval holes are represented in an Excellon drill file. When not enabled, a route command is used to represent oval holes. This is correct for most manufacturers. Only choose the **Use alternate drill mode** setting if requested by your manufacturer.
- **Generate map:** Choose whether to generate a drill map and, if so, in which format. Supported formats are Postscript, Gerber X2, DXF, SVG, and PDF.
-

**Origin:** Choose the coordinate origin for drill files. **Absolute** will use the page origin at the top left corner. **Drill/place file origin** will use the origin specified in the board design.

- **Drill units:** Choose the units for drill coordinates and sizes.
- **Zeros:** Controls how zeroes are formatted in an Excellon drill file. Select an option here based on your manufacturer's recommendations.

## IPC-2581

IPC-2581 files are XML files that contain complete fabrication and assembly data for a board design. If your manufacturer accepts IPC-2581 files, these can replace Gerber files, drill files, and component placement files. To create an IPC-2581 file, select **File** → **Fabrication Outputs** → **IPC-2581 File (.xml)**....

Export IPC-2581

File: ./StickHub.xml

**File Format**

Units: Millimeters

Precision: 4

Version: C

Compress output

**BOM Columns**

Internal ID: Generate Unique

Manufacturer P/N: Omit

Manufacturer: N/A

Distributor P/N: Omit

Distributor: N/A

Close Export

There are several options for generating IPC-2581 output.

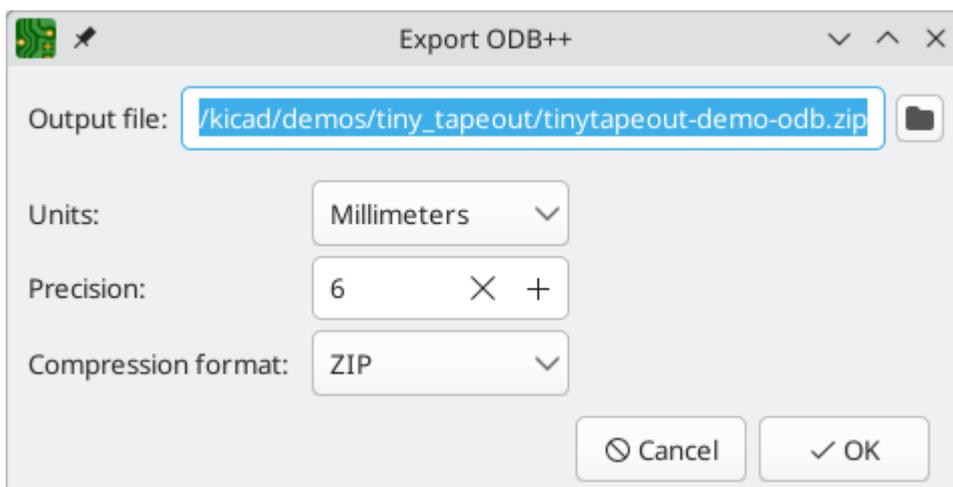
- **File:** Choose the filename for the generated IPC-2581 file. If a relative path is entered, it will be relative to the project directory.
- **Units:** Choose the units for the generated file. Can be **millimeters** or **inches**.
- **Precision:** Choose the number of digits after the decimal point for numbers in the generated file.
- **Version:** Choose the IPC-2581 standard version (B or C).
- **Compress output:** If enabled, the generated file will be compressed as a ZIP file.
- **Internal ID:** Choose the footprint field to use for the BOM's internal ID column. This can be a generated unique ID or set to any footprint field in the design.
- **Manufacturer P/N:** Choose the footprint field to use for the BOM's manufacturer part number column. This can be omitted or set to any footprint field in the design.
- **Manufacturer:** Choose the footprint field to use for the BOM's manufacturer column. This can be omitted or set to any footprint field in the design.

**Distributor P/N:** Choose the footprint field to use for the BOM's distributor part number column. This can be omitted or set to any footprint field in the design.

- **Distributor:** Choose the footprint field to use for the BOM's distributor column. This can be omitted or set to any footprint field in the design.
- **BOM revision:** Specify the value for the BOM's revision field. If omitted, the value from the schematic root sheet's `Revision` field is used.

## ODB++

ODB++ output is a database of files that contains complete fabrication and assembly data for a board design. If your manufacturer accepts ODB++ files, these can replace Gerber files, drill files, and component placement files. To create an ODB++ file, select **File** → **Fabrication Outputs** → **ODB++ Output File...**



There are several options for generating ODB++ output.

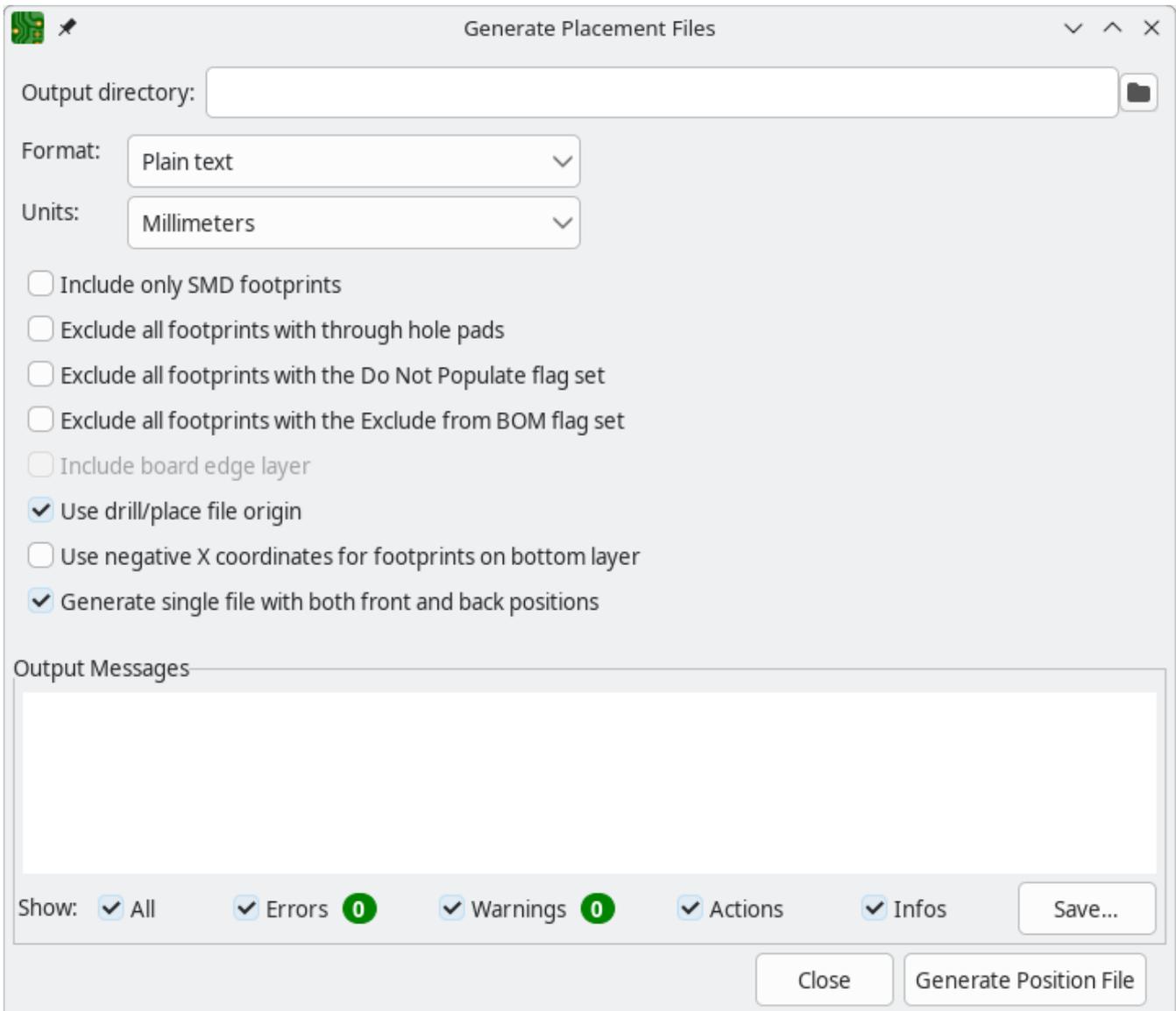
- **Output file:** Choose the filename for the generated ODB++ file. If a relative path is entered, it will be relative to the project directory.
- **Units:** Choose the units for the generated file. Can be **millimeters** or **inches**.
- **Precision:** Choose the number of digits after the decimal point for numbers in the generated file.
- **Compression format:** Choose the type of compression for the generated output. Can be **ZIP**, **TGZ**, or **none**. If none, the output will be a folder.

## Component placement (position) files

Component placement files, or position files, are text files that list each component (footprint) on the board along with its center position and orientation. These files are usually used for programming pick-and-place machines, and may be required by your manufacturer if you are ordering fully-assembled PCBs. To create placement files, select **File** → **Fabrication Outputs** → **Component Placement (.pos, .gbr)...**

### NOTE

A footprint will not appear in generated placement files if the "Exclude from position files" option is enabled for that footprint. This may be used for excluding certain footprints that do not represent physical components to be assembled. You can also optionally exclude DNP or "Exclude from BOM" components, depending on your manufacturer's requirements.



There are several options for generating placement files.

- **Format:** Choose between generating a plain text (UTF-8), comma-separated text (CSV), or Gerber X3 placement file format.
- **Units:** Choose the units for component locations in the placement file.
- **Include only SMD footprints:** When enabled, only footprints with the SMD fabrication attribute will be included. Check with your manufacturer to determine if non-SMD footprints should be included or excluded from the position file.
- **Exclude all footprints with through hole pads:** When enabled, footprints will be excluded from the placement file if they contain any through-hole pads, even if their fabrication type is set to SMD.
- **Exclude all footprints with the Do Not Populate flag set:** When enabled, footprints will be excluded from the placement file if they have the Do Not Populate attribute set. Check with your manufacturer to determine if DNP components should be included or excluded from the position file.
- **Exclude all footprints with the Exclude from BOM flag set:** When enabled, footprints will be excluded from the placement file if they have the Exclude from BOM attribute set.
- **Include board edge layer:** For Gerber placement files, controls whether or not the board outline is included with the footprint placement data.

- **Use drill/place file origin:** When enabled, component positions will be relative to the drill/place file origin set in the board design. When disabled, the positions will be relative to the page origin (upper left corner).
- **Use negative X coordinates for footprints on bottom layer:** When enabled, the X coordinates will be flipped (negated) for footprints on the bottom layer.
- **Generate single file with both front and back positions:** When enabled, positions for front and back footprints will be saved in a single file. When disabled, separate files will be generated for front and back footprints.

## Footprint reports

This exporter generates a text report of all the footprints in the board. To create a footprint report, select **File** → **Fabrication Outputs** → **Footprint Report (.rpt)**.... There are no configurable options.

## IPC-D-356 netlists

This exporter generates a netlist of the board in IPC-D-356 format, which is suitable for manufacturing checks of bare PCBs. To create an IPC-D-356 netlist, select **File** → **Fabrication Outputs** → **IPC-D-356 Netlist File**.... There are no configurable options.

## Bill of Materials

This exporter generates a Bill of Materials (BOM) in CSV format containing all of the footprints on the board. To create a Bill of Materials, select **File** → **Fabrication Outputs** → **Bill of Materials**.... There are no configurable options.

### WARNING

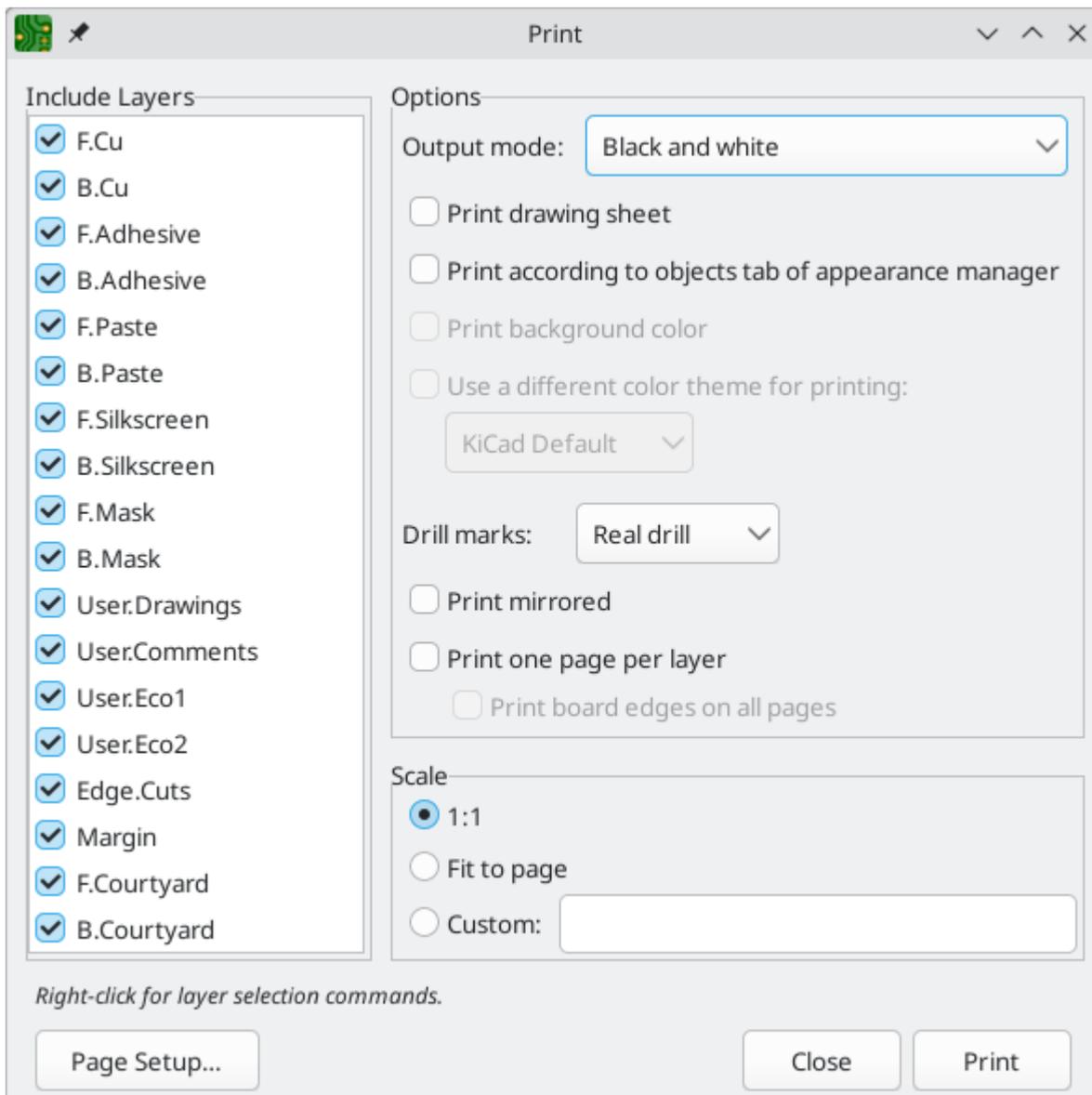
This BOM exporter is included for legacy reasons, and is not recommended. Instead, it is recommended to use the [BOM Exporter in the Schematic Editor](#), which supports multiple output formats and is highly configurable.

## Printing

KiCad can print the board view to a standard printer using the print function. To print a board, select **File** → **Print**....

### NOTE

While the Print action can be used with a PDF printer to generate a PDF, the [Plot function](#) is recommended instead as it is more configurable and may be more accurate.



There are several options for printing.

- **Include layers:** Select the layers to include in the printout. Unselected layers will be invisible. Right-click the list for layer selection commands.
- **Output mode:** Choose whether to print in black and white or full color.
- **Print drawing sheet:** When enabled, the page border and title block will be printed.
- **Print according to objects tab of appearance manager:** When enabled, any objects that have been hidden in the Objects tab of the Appearance panel will be hidden in the printout. When disabled, these objects will be printed if the layer they appear on is selected in the Included Layers area.
- **Print background color:** When printing in full color, this option controls whether or not the view background color will be printed.
- **Use a different color theme for printing:** When printing in full color, this option allows a different color theme to be used for printing. When disabled, the color theme used by the board editor will be used for printing.
- **Drill marks:** Controls whether to show drilled holes at their actual size, at a small size, or hide them from the printout.

**Print mirrored:** When enabled, the printout will be mirrored horizontally.

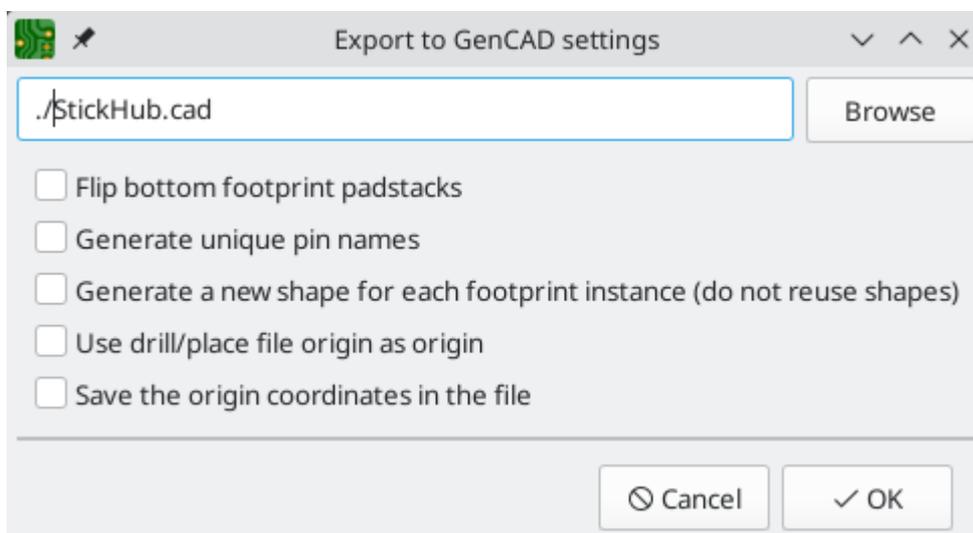
- **Print one page per layer:** When enabled, each layer selected in the Included Layers area will be printed to an individual page. If this option is enabled, the **Print board edges on all pages** option controls whether to add the Edge.Cuts layer to each printed page.
- **Scale:** controls the scale of the printout relative to the page size configured in Page Setup.

## Specctra DSN

The Specctra DSN exporter creates a file suitable for importing into certain third-party autorouter software. To create a Specctra DSN file, select **File** → **Export** → **Specctra DSN...** There are no configurable options.

## GenCAD

The GenCAD exporter creates a GenCAD file for fabrication, testing, or importing into other software. To create a GenCAD file, select **File** → **Export** → **GenCAD...**

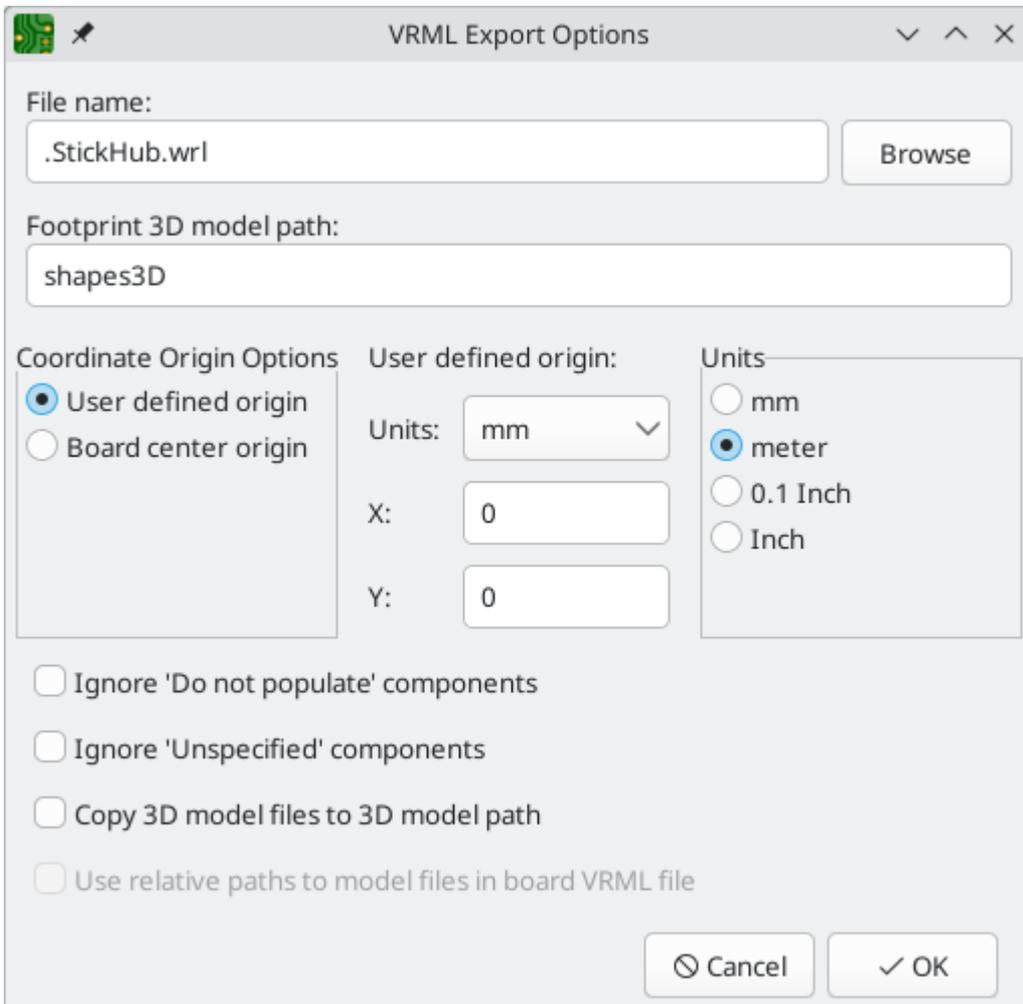


There are several options for generating GenCAD files.

- **Flip bottom footprint padstacks:** If enabled, separate flipped padstack definitions will be added for bottom-side footprints. This may be necessary for importing into some third-party software.
- **Generate unique pin names:** If enabled, a suffix will be added to each pin name so that no footprint in the generated file will have two pins with the same name.
- **Generate a new shape for each footprint instance:** If enabled, a unique footprint will be output for every footprint instance, even if two footprints are identical.
- **Use drill/place file origin as origin:** If enabled, coordinates in the generated file will be relative to the drill/place file origin.
- **Save the origin coordinates in the file:** If enabled, the selected origin coordinates will be included in the generated file. If not enabled, the origin in the generated file will be set to (0,0).

## VRML

The VRML exporter creates a VRML ( .wrl ) 3D model file containing the PCB and any VRML files specified in footprints. VRML models are suitable for use in applications where visual appearance is important and dimensional accuracy is not critical. To create a VRML file, select **File** → **Export** → **VRML...**



There are several options for generating VRML files.

- **Coordinate origin options:** Selects the origin for the generated model. If **user defined origin** is selected, you can manually specify the origin point relative to the configured display origin.
- **Units:** Selects the unit system for the generated model. Dimensions in the generated model will be scaled appropriately.
- **Ignore 'Do not populate' components:** If enabled, VRML files for footprints with the 'Do not populate' attribute set will not be included.
- **Ignore 'Unspecified' components:** If enabled, VRML files for footprints with the 'Unspecified' footprint type will not be included.
- **Copy 3D model files to 3D model path:** If enabled, VRML files referenced in footprints will be copied into a subdirectory of the directory containing the generated board VRML model, and the generated model will reference the copied files. The subdirectory name is set by the **footprint 3D model path** field. If disabled, VRML files referenced in footprints will be embedded in the generated VRML files.
- **Use relative paths to model files in board VRML file:** If enabled, references to external models will use paths relative to the generated board VRML file. If disabled, the references will use absolute paths. This option is only available when the **copy 3D model files to 3D model path** option is enabled.

## IDF

The IDF exporter exports an [IDFv3](#) compliant board (.emn) and library (.emp) file for communicating mechanical dimensions to a mechanical CAD package. The exporter exports the board outline and cutouts, all pad and mounting through holes including slotted holes, and component outlines; this is the most basic set of mechanical data required for interaction with mechanical designers. All other entities described in the IDFv3 specification are currently not exported.

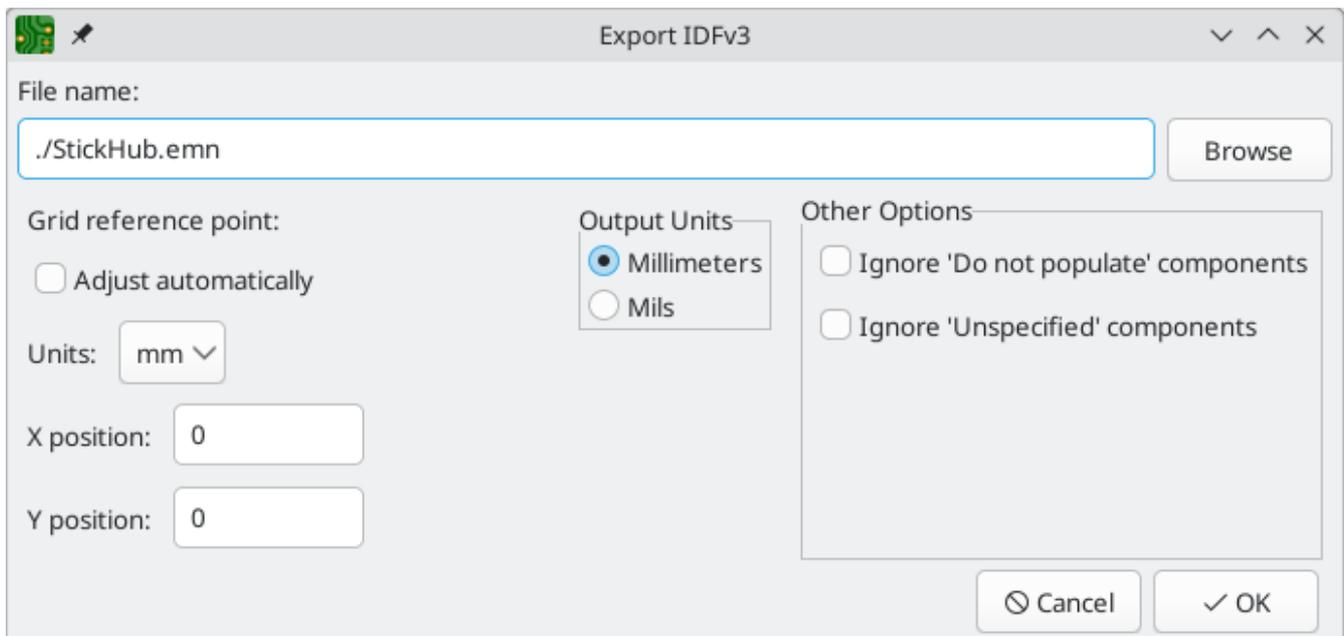
### NOTE

You must attach IDF component models to your design's footprints before they will be included in the exported model. For more information on attaching models to footprints, see the [footprint documentation](#). Some IDF-specific guidance is included in the [Advanced Topics documentation](#).

### NOTE

For more information on creating IDF component models, including descriptions of the IDF utility tools included with KiCad, see the [Advanced Topics documentation](#).

Once models have been specified for all desired components, the model of the board can be exported. In the PCB Editor, select **File** → **Export** → **IDFv3...**



There are several options for generating IDF files.

- **Grid reference point:** Choose where the exported model's reference point should be. If the **Adjust automatically** option is selected, KiCad will set the reference point to the centroid of the PCB. Otherwise, the reference point is set relative to the display origin.
- **Output units:** Choose whether the exported model's units are millimeters or mils.
- **Ignore 'Do not populate' components:** If enabled, IDF files for footprints with the 'Do not populate' attribute set will not be included.
- **Ignore 'Unspecified' components:** If enabled, IDF files for footprints with the 'Unspecified' footprint type will not be included.

The outputs can be viewed directly in a mechanical CAD application or converted to VRML using the `idf2vrm1` [tool](#).

## 3D models (STEP / GLB / BREP / XAO / PLY / STL / STPZ / U3D / PDF)

The 3D model exporter creates a 3D model file from the PCB and any STEP files specified in footprints. A number of formats are supported:

- STEP
- GLB (binary glTF)
- BREP (OCCT-native boundary representation)
- XAO (SALOME/Gmsh)
- PLY
- STL
- STPZ (GZIP-compressed STEP)
- U3D
- PDF

Different formats may be appropriate for different usecases. For example, STEP models are suitable for use in mechanical CAD applications, while XAO models are useful for physical simulations.

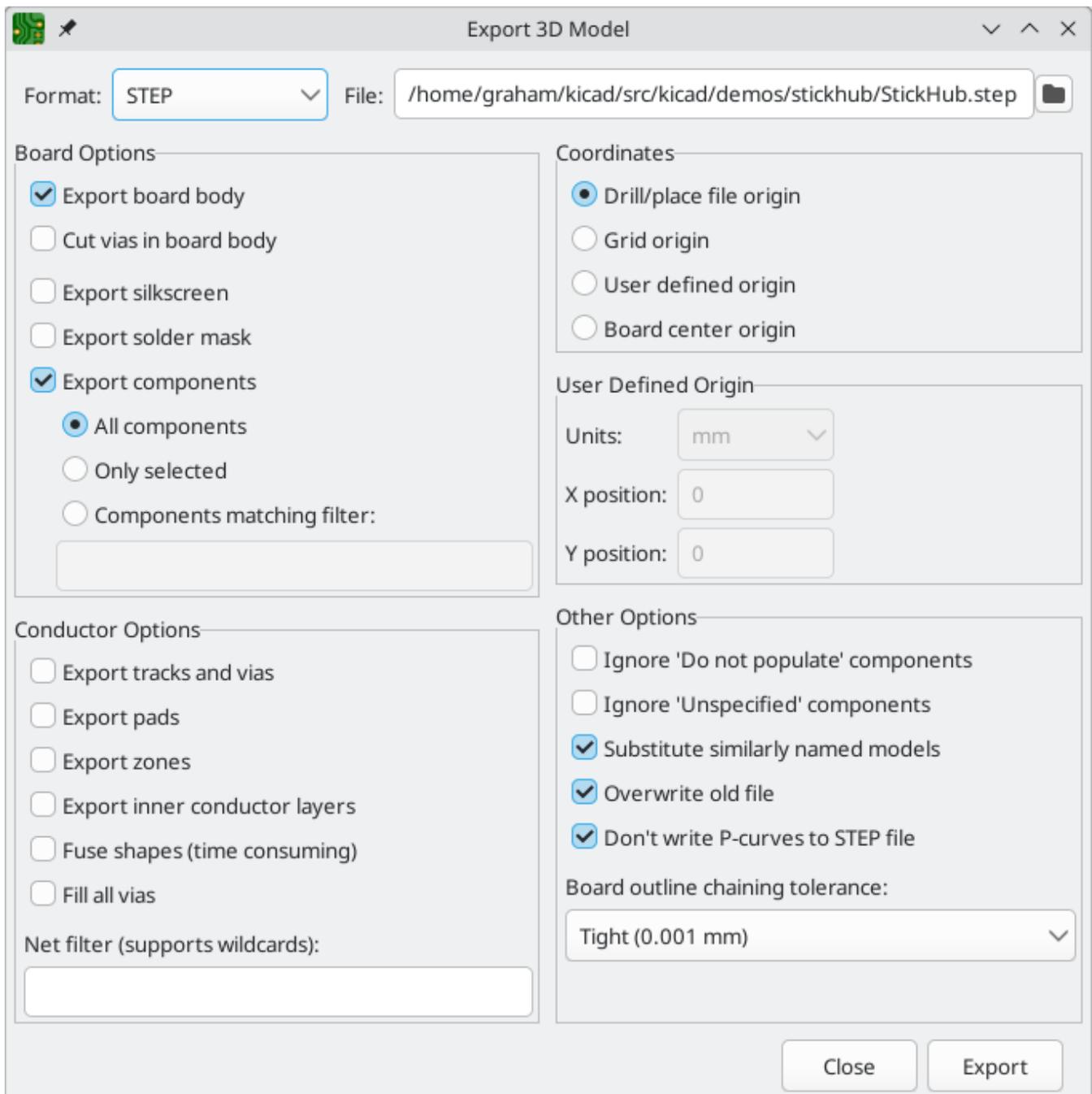
### NOTE

KiCad's footprint library includes both STEP and VRML ( `.wr1` ) versions of each model. However, footprints in KiCad's library only reference the VRML versions of the models. VRML models are not included in STEP exports, but the STEP exporter will instead include the corresponding STEP version of the model if the **substitute similarly named models** option is enabled.

### NOTE

KiCad can also export 3D models in [VRML](#) and [IDF](#) formats, but these formats use separate exporters.

To use the 3D model exporter, select **File** → **Export** → **STEP / GLB / BREP / XAO / PLY / STL....**



Choose a 3D model format from the **Format** dropdown menu and specify an output filename in the **File** selector.

There are a number of options for configuring the output model.

## Board options

- **Export board body:** If enabled, the board body (non-copper) will be modeled in the exported model.
- **Cut vias in board body:** If enabled, via holes will be cut in the board body even if conductor layers are not modeled.
- **Export silkscreen:** If enabled, silkscreen will be modeled in the exported model. Silkscreen is modeled as a set of flat faces; it is not three-dimensional.
- **Export solder mask:** If enabled, solder mask will be modeled in the exported model. Solder mask is modeled as a set of flat faces; it is not three-dimensional.

**Export components:** If enabled, 3D models for components will be included in the exported model (but see **Substitute similarly named models**, below). If **All components** is selected, models for all components in the PCB will be included. If **Only selected** is chosen, only models for the footprints currently selected in the board will be included. If **Components matching filter** is selected, only models for footprints with references matching the filter will be included. The filter supports wildcards and commas, so `C1,R*` will include `C1` and all resistors.

## Conductor options

- **Export tracks and vias:** If enabled, tracks and vias on outer layers will be modeled in the exported model.
- **Export pads:** If enabled, pads will be modeled in the exported model.

### NOTE

0.005mm of additional metal thickness is added by the exporter to each pad. This causes pads to be separate faces in the exported model, distinct from the surrounding metal. If this additional thickness is not wanted, you can use `kicad-cli` with the `--no-extra-pad-thickness` option to export a 3D model without the additional pad thickness.

- **Export zones:** If enabled, zones on outer layers will be modeled in the exported model.
- **Export inner conductor layers:** If enabled, inner conductor layers will be modeled in the exported model.
- **Fuse shapes (time consuming):** If enabled, intersecting geometry will be fused into a single shape. This may make the exported file easier to work with in some tools, but it also significantly increases the export time.
- **Fill all vias:** If enabled, via holes will not be cut in conductor layers.
- **Net filter (supports wildcards):** If filled, only conductors corresponding to nets that match the filter will be modeled. The filter supports wildcards, so `/tx_*` will model `/tx_p` and `/tx_n` conductors.

## Coordinates

- **Coordinates:** Selects the origin for the generated model. If **user defined origin** is selected, you can manually specify the origin point relative to the configured display origin.

## Other options

- **Ignore 'Do not populate' components:** If enabled, components with the DNP attribute set will not be included in the exported model.
- **Ignore 'Unspecified' components:** If enabled, components with the Unspecified footprint type will not be included in the exported model.
- **Substitute similarly named models:** VRML models cannot be used in STEP, BREP, or XAO exports, but if this option is enabled the exporter will look for an identically named STEP model to include in the export instead of a footprint's specified VRML model. Note that footprints in KiCad's footprint library specify VRML models, but suitably named STEP models are also included for each VRML model. Therefore this option must be enabled in order to export 3D models for footprints from KiCad's library using this dialog.
- **Overwrite old file:** If enabled, the exported model will overwrite an existing file with the same name.

- **Don't write P-curves to STEP file** If enabled, parametric curves will be disabled in the exported STEP/STPZ model. This reduces the file size, but may reduce compatibility with some software.
- **Board outline chaining tolerance:** Controls the minimum distance between two points for the points to be considered coincident. If the board outline in the exported model is not contiguous, try increasing this tolerance.

## Footprint association (CMP) files

CMP files are used to sync footprint assignments and some other footprint fields between the PCB and the schematic. You can export CMP files by selecting **File** → **Export** → **Footprint Association (.cmp) File...** and import CMP files into the schematic using the Schematic Editor's **File** → **Import** → **Footprint Assignments** menu item. This provides a very limited form of backannotation. It is recommended to use the [Update Schematic from PCB tool](#) instead. There are no configurable options.

## Hyperlynx

The Hyperlynx exporter creates a file suitable for importing into Mentor Graphics (Siemens) HyperLynx simulation and analysis software. To create a HyperLynx file, select **File** → **Export** → **Hyperlynx...** There are no configurable options.

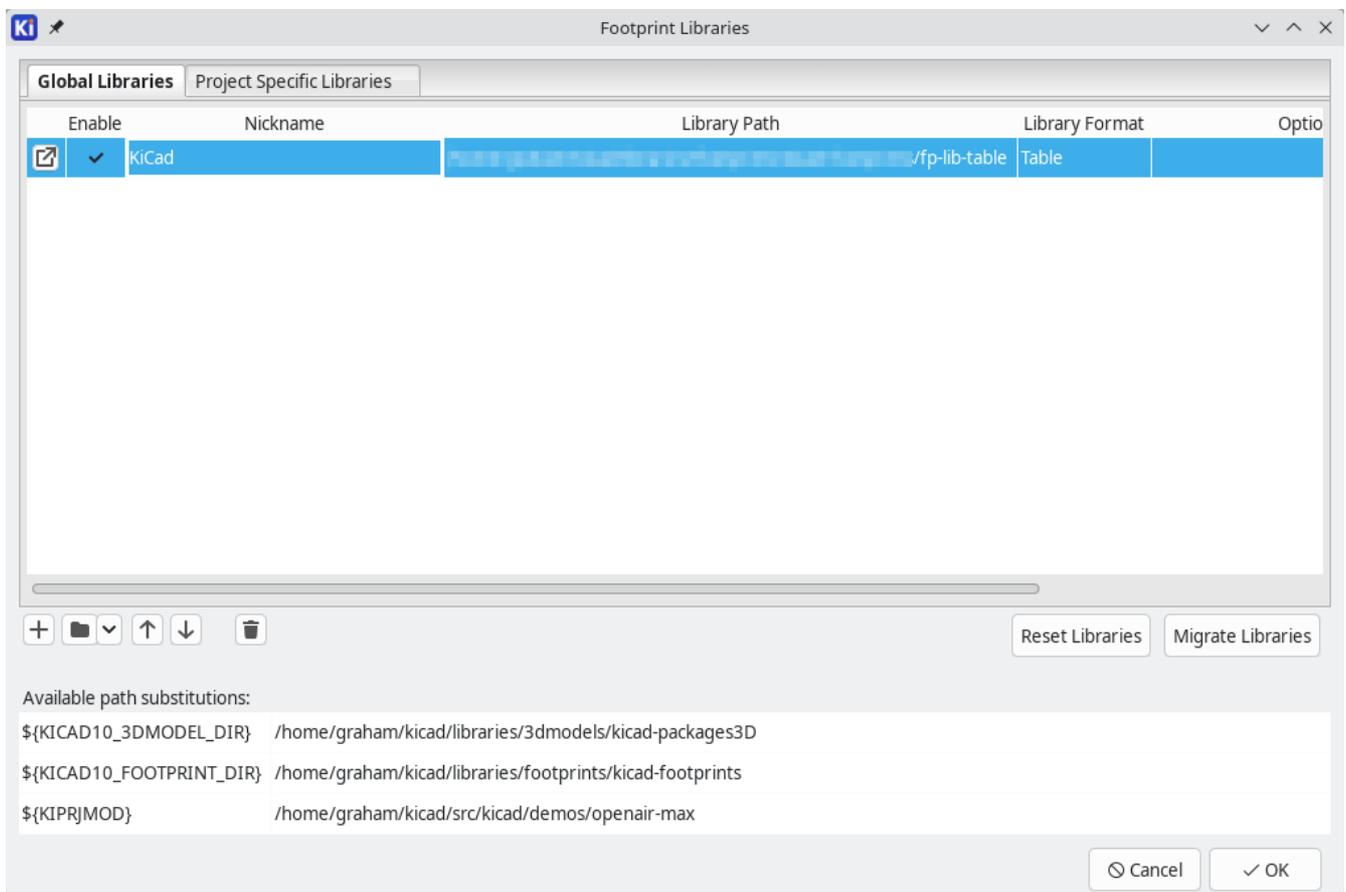
# Footprints and footprint libraries

KiCad organizes footprints into footprint libraries, which hold collections of footprints. Each footprint in a board is uniquely identified by a full name that is composed of a library nickname and a footprint name. For example, the identifier `Capacitor_SMD:C_0603_1608Metric` refers to the `C_0603_1608Metric` footprint in the `Capacitor_SMD` library.

## Managing footprint libraries

KiCad uses a table of footprint libraries to map footprint libraries of any supported library type to a library nickname. KiCad uses a global footprint library table as well as a table specific to each project. Libraries are not loaded unless they are included in either the global or project-specific library table.

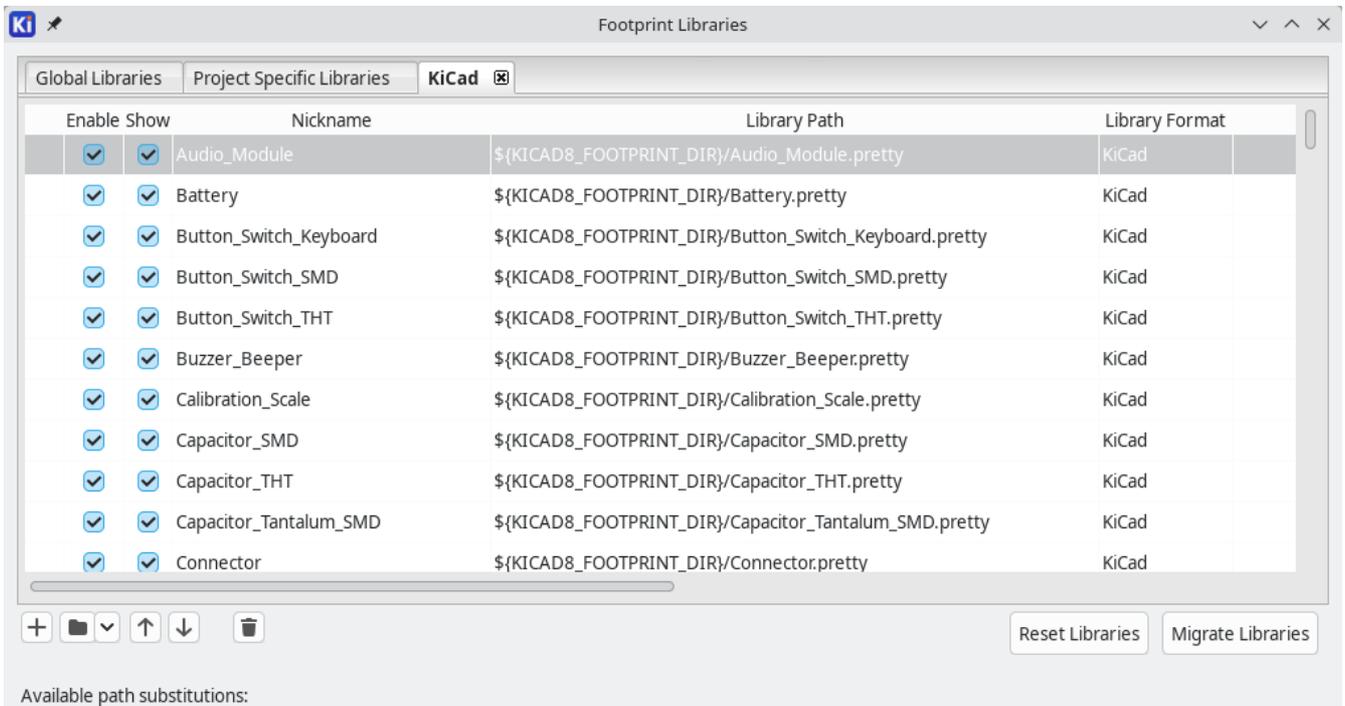
To edit either footprint library table, use **Preferences** → **Manage Footprint Libraries...**



The global footprint library table contains the list of libraries that are always available regardless of the currently loaded project. The table is saved in the file `fp-lib-table` in the KiCad configuration folder. [The location of this folder](#) depends on the operating system being used.

The project specific footprint library table contains the list of libraries that are available specifically for the currently loaded project. If there are any project-specific footprint libraries, the table is saved in the file `fp-lib-table` in the project folder.

Many (or even all) of the loaded libraries may be contained in a [nested library table](#), as shown in the screenshot above. To expand a nested library table and view its contents, press the  button next to the nested library table entry. This opens the nested table in a new tab, as shown below.



KiCad's footprint library management system allows directly using many types of footprint libraries, including formats that are native to other non-KiCad EDA tools:

- KiCad `.pretty` footprint libraries (folders with `.pretty` extension, containing `.kicad_mod` files)
- KiCad Legacy footprint libraries ( `.mod` files)
- Altium Designer ( `.PcbLib` or `.IntLib` files)
- CADSTAR PCB Archive ( `.cpa` files)
- Eagle footprint libraries ( `.lbr` files)
- EasyEDA / JLCEDA Standard Edition ( `.json` or `.zip` files)
- EasyEDA / JLCEDA Professional Edition ( `.elibz`, `.epro`, or `.zip` files)
- GEDA libraries (folders containing `.fp` files)

Non-KiCad footprint libraries, including KiCad Legacy footprint libraries, can be migrated to KiCad `.pretty` format using the **Migrate Libraries** button (see the [migrating libraries](#) section).

#### NOTE

KiCad only supports writing to KiCad's native `.pretty` format footprint libraries (and the `.kicad_mod` footprint files within them). All other footprint library formats are read-only. To modify a non-KiCad format footprint library, you must first convert it to KiCad format.

Library tables can also include other library tables, which are called [nested library tables](#). When a nested library table is used, the libraries in the nested table are loaded as if they were included directly in the main table.

## Initial configuration

The first time KiCad runs (or any time KiCad runs and no configuration is found), KiCad will guide you through creating a new configuration or importing a configuration from a previous version. This includes

setting up a new footprint library table. This process is described in the [Project Manager documentation](#).

You can reset your footprint library table to the default by clicking the **Reset Libraries** button in the footprint library table dialog.

#### WARNING

Resetting your footprint library table will permanently change your footprint library table on disk.

The default footprint library table contains a single entry, which points to another footprint library table containing all of KiCad's default footprint libraries. This [nested library table](#) is maintained as part of the KiCad libraries and is updated along with KiCad and its libraries. You should not edit it yourself.

#### NOTE

KiCad's default footprint libraries are included in your library table via a nested library. This allows the list of default KiCad libraries to be updated when KiCad and its libraries are updated, without affecting your personal library configuration. Your personal library table configuration can be maintained between KiCad versions, while still allowing you to receive updates to the default libraries.

## Managing table entries

Footprint libraries can only be used if they have been added to either the global or project-specific footprint library table (or to a nested table included in either top-level table).

Add a library either by clicking the  button and selecting a library or clicking the  button and typing the path to a library file. The selected library will be added to the currently opened library table (Global or Project Specific). Libraries can be removed by selecting desired library entries and clicking the  button.

When an entry in the library table is itself another library table, you can open this [nested table](#) by clicking the  button next to the nested table's entry. This opens the nested table in a new tab, where you can see and edit the libraries included in the nested table. To add a nested library table, click the downwards arrow next to the  button, choose the **Table** library type, then browse to the table file.

The  and  buttons move the selected library up and down in the library table. This does not affect the display order of libraries in the Footprint Library Browser, Footprint Editor, or Add Footprint tool.

Libraries can be made inactive by unchecking the **Enable** checkbox in the first column. Disabled libraries are still in the library table but do not appear in any library browsers and are not loaded from disk, which can reduce loading times.

A range of libraries can be selected by clicking the first library in the range and then -clicking the last library in the range.

Each library must have a unique nickname: duplicate library nicknames are not allowed in the same table. However, nicknames can be duplicated between the global and project library tables. Libraries in the project table take precedence over libraries with the same name in the global table.

Library nicknames do not have to be related to the library filename or path. The colon character ( : ) cannot be used in library nicknames or footprint names because it is used as a separator between nicknames and footprints.

Each library entry must have a valid path. Paths can be defined as absolute, relative, or by [path variable substitution](#).

The appropriate library format must be selected in order for the library to be properly read. The supported formats are listed above. Only KiCad format libraries ( `.pretty` folders containing `.kicad_mod` files) can be saved. Other footprint library formats are read-only and must be converted to KiCad format before you can modify them.

There is an optional description field to add a description of the library entry. The option field is not used at this time so adding options will have no effect when loading libraries.

## Nested library tables

In addition to containing libraries, library tables can also contain other library tables. Library tables that are referenced by other library tables in this way are called *nested* library tables. Any libraries in a nested library table are loaded as if they were directly listed in the top-level library table.

Nested library tables let you separate groups of libraries into different library table files, then include all of the individual tables into the main top-level library table. This could be used, for example, to make a standard library table for your company, and share that table with everyone who needs it. Each user can then access the standard set of company libraries after they add the shared table as a nested library table in their own global table.

When KiCad is configured to its default library table configuration, the global library table contains a single entry, which is a nested library table. This nested table is part of KiCad's default libraries and lists all of the KiCad default footprint libraries. When KiCad is updated, this nested table is also updated, which allows you to receive an updated list of KiCad's default libraries without disturbing your personal library configuration.

## Path variable substitution

The footprint library tables support path variable substitution, which allows you to define path variables containing custom paths to where your libraries are stored. PATH variable substitution is supported by using the syntax `${PATH_VAR_NAME}` in the footprint library path.

By default, KiCad defines several path variables which are described in the [project manager documentation](#). Path variables can be configured in the **Preferences** → **Configure Paths...** dialog.

Using path variables in the footprint library tables allows libraries to be relocated without breaking the footprint library tables, so long as the path variables are updated when the library location changes.

### NOTE

KiCad will automatically resolve versioned path variables from older versions of KiCad to the value of the corresponding variable from the current KiCad version, as long as the old variable is not explicitly defined itself. For example, `${KICAD9_FOOTPRINT_DIR}` will automatically resolve to the value of `${KICAD10_FOOTPRINT_DIR}` if there is no `KICAD9_FOOTPRINT_DIR` variable defined.

`${KIPRJMOD}` is a special path variable that always expands to the absolute path of the current project directory. `${KIPRJMOD}` allows libraries to be stored in the project folder without having to use an absolute path in the project library table. This makes it possible to relocate projects without breaking their project library tables.

## Using the GitHub plugin

### NOTE

KiCad removed support for the GitHub library plugin in version 6.0.

## Migrating footprint libraries to KiCad format

Non-KiCad format libraries, including legacy libraries ( .mod files), are read-only. They need to be converted to KiCad format ( .kicad\_mod files in a .pretty folder) before you can save changes to them.

### NOTE

As with most KiCad files, newer versions of KiCad can open older-format library files, but older versions of KiCad cannot read files once they have been saved by a newer version of KiCad.

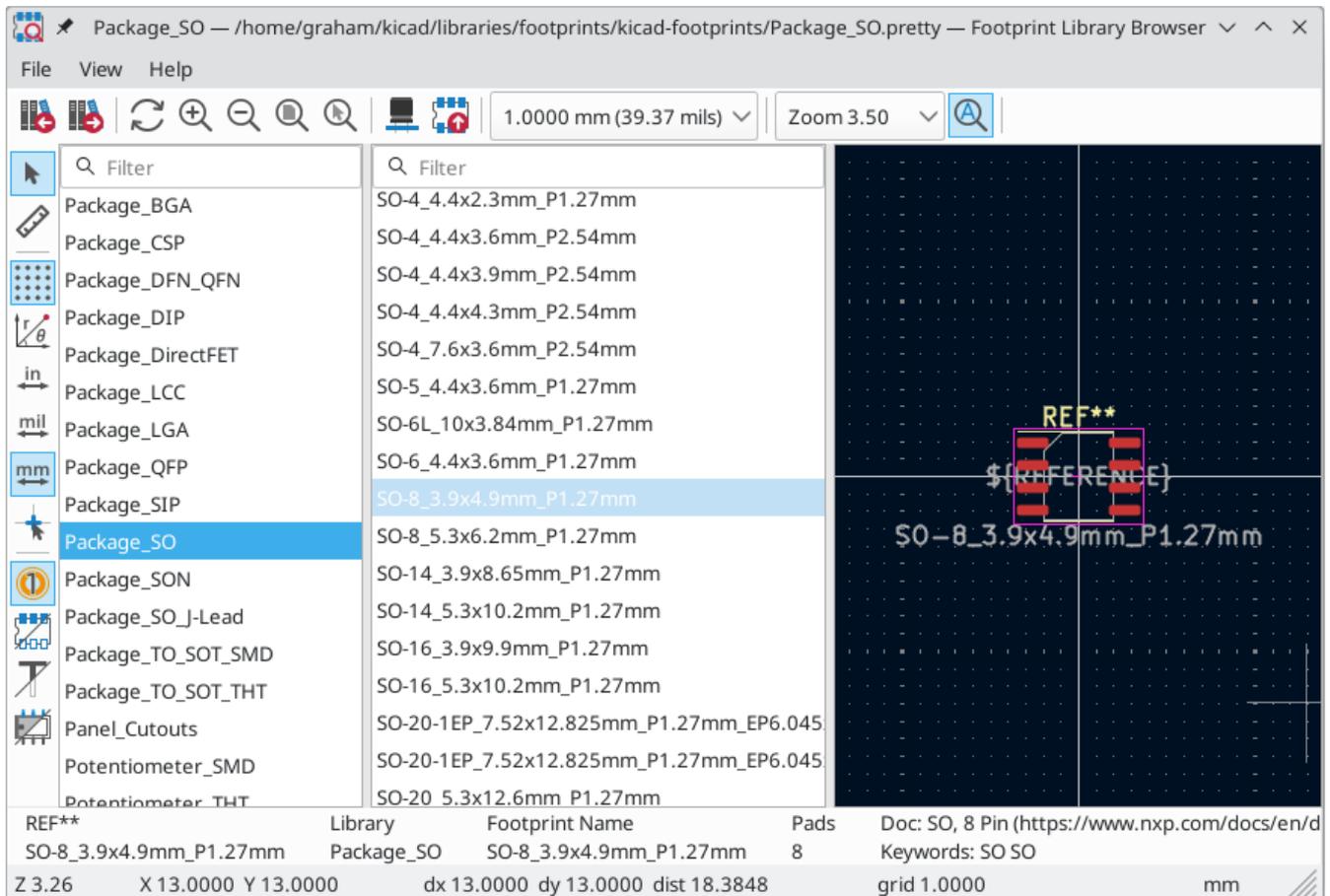
Libraries in other formats can be converted to KiCad libraries by selecting them in the footprint library table and clicking the **Migrate Libraries** button. Multiple libraries can be selected and migrated at once by **Ctrl**-clicking or **shift**-clicking.

Libraries can also be converted one at a time by opening them in the Footprint Editor and saving them as a new library.

## Browsing footprint libraries

The Footprint Library Browser allows you to quickly examine the contents of footprint libraries. The Footprint Library Viewer can be accessed by clicking  icon on the main Board Editor toolbar or with **View** → **Footprint Library Browser**.

To examine the contents of a library, select a library from the list in the left hand pane. All footprints in the selected library will appear in the second pane. Select a footprint name to view the footprint.



Double clicking the name of a footprint or using the  button adds the footprint to the board.

The top toolbar contains the following commands:

	Select previous footprint in library.
	Select next footprint in library.
	Zoom tools.
	Open footprint in 3D Viewer.
	Add the footprint to the board.
	Automatically zoom to fit each opened footprint.

The left toolbar contains the following commands:

	Selection tool (the default tool).
	Interactively measure the distance between two points.
	Turn grid display on/off.
	Switch between polar and Cartesian coordinate display in the status bar.
	Display/entry of coordinates and dimensions in inches, mils, or millimeters. Clicking the button toggles to the next unit, or you can choose a unit directly by expanding the palette (click and hold/drag).
	Switches the cursor crosshair between small, fullscreen, and 45-degree-rotated fullscreen. Clicking the button toggles to the next crosshair type, or you can choose a crosshair directly by expanding the palette (click and hold/drag).
	Show or hide pad numbers.
	Switch display of pads between filled and outline mode.
	Switch display of text between filled and outline mode.
	Switch display of graphic items between filled and outline mode.

# Creating and editing footprints

A footprint is the physical interface between a component package and a circuit board. Footprints can contain:

- Pads, which define how the component will be physically assembled onto the footprint. When a footprint is added to a board, tracks are routed to pads, and pads provide a magnetic snapping point for the router to connect the pad to a track. Pad shapes and layers are fully customizable, and pads can have plated holes, unplated holes, or no hole.
- Graphic shapes and text for technical or aesthetic purposes. Graphics can be placed on physical layers (e.g. silkscreen or soldermask) or nonphysical layers. Graphic shapes can also be placed on copper layers, in which case they can make electrical connections.
- 3D models for mechanical CAD and visualization. 3D models are normally external files that footprints can link to, but they can optionally be embedded in footprints.
- Metadata associated with the footprint.

Footprints in KiCad are organized into footprint libraries, which contain zero or more footprints. Generally footprints are logically grouped by footprint category, function, and/or manufacturer. Each library is a folder (usually ending in `.pretty`) containing a `.kicad_mod` file for each footprint in the library.

## Footprint editor overview

KiCad provides a footprint editing tool that allows you to create footprint libraries; add, edit, delete, or transfer footprints between libraries; export footprints to files; and import footprints from files. The Footprint Editor can be launched from the KiCad Project Manager or from the Board Editor (**Tools** → **Footprint Editor**). You can also open the Footprint Editor from the [a footprint in the board](#); in this way you can edit either the library copy or the board copy of that footprint in the editor.

### NOTE

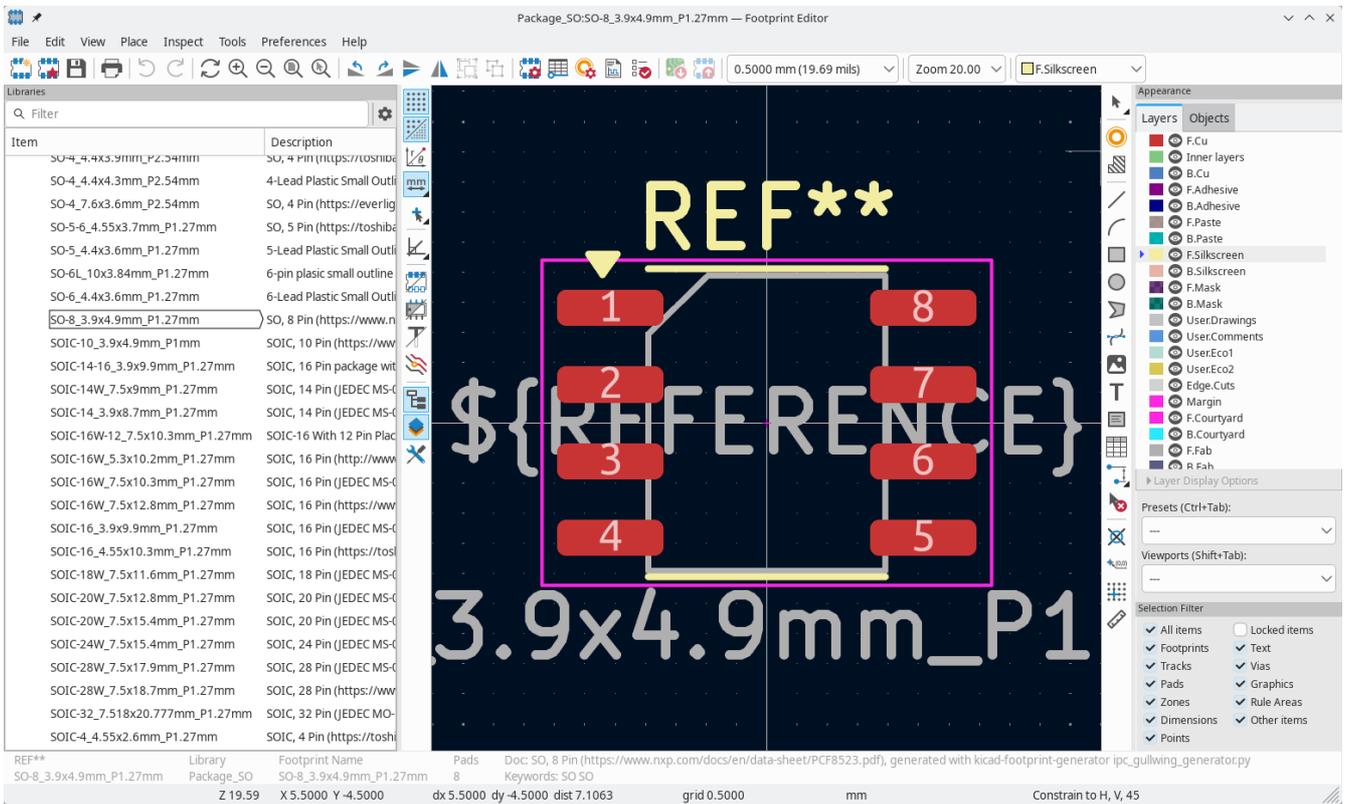
Editing the library version of a footprint will not affect any copies of that footprint that have been added to a board until the board copy is updated from the library. Conversely, editing the board version of a footprint will not affect the library version of a footprint or any other copies of that footprint in a board.

The Footprint Editor main window is shown below. It has three toolbars for quick access to common features and a footprint viewing/editing canvas. Not all commands are available on the toolbars, but all commands are available in the menus.

### NOTE

You can edit the toolbars' contents in the **Toolbar** page of the Footprint Editor Preferences.

In addition to the toolbars, there are collapsible panels for the footprint tree and Properties Manager (not shown) on the left, and the appearance panel and selection filter on the right. The bottom of the window contains a message panel that shows details about the selected object.



## Top toolbar

The main toolbar is at the top of the main window. It has buttons for the undo/redo commands, zoom commands, footprint/pad properties dialogs, and layer/grid management controls.

	Create a new footprint in the selected library.
	Create a new footprint in the selected library using a footprint wizard.
	Save the currently selected footprint.
	Print the currently selected footprint.
	Undo last edit.
	Redo last undo.
	Refresh display.
	Zoom in.
	Zoom out.
	Zoom to fit footprint in display.
	Zoom to fit selection.
	Rotate selected item(s) counter-clockwise.
	Rotate selected item(s) clockwise.
	Mirror selected item(s) horizontally.
	Mirror selected item(s) vertically.
	Add the selected item(s) to a <a href="#">group</a> .
	Remove the selected item(s) from a <a href="#">group</a> .
	Edit the current <a href="#">footprint's properties</a> .
	Edit the selected <a href="#">pad's properties</a> .
	Open the current footprint's datasheet.
	Run the <a href="#">footprint checker</a> to test the current footprint for design errors.
	Edit a footprint in the current board in the footprint editor.
	Insert current footprint into the board.

## Left toolbar display controls

The left toolbar provides options to change the display of items in the Footprint Editor.

	Turn grid display on/off.  <b>Note:</b> by default, hiding the grid does not disable grid snapping. This behavior can be changed in the Display Options section of Preferences.
	Turn item-specific grid overrides on/off.
	Switch between polar and Cartesian coordinate display in the status bar.
  	Display/entry of coordinates and dimensions in inches, mils, or millimeters. Clicking the button toggles to the next unit, or you can choose a unit directly by expanding the palette (click and hold/drag).
  	Switches the cursor crosshair between small, fullscreen, and 45-degree-rotated fullscreen. Clicking the button toggles to the next crosshair type, or you can choose a crosshair directly by expanding the palette (click and hold/drag).
  	Switches the line mode between free angle, 90 degree mode, and 45 degree mode for placement of new zones, graphical shapes, dimensions, and other objects. Clicking the button toggles to the next line mode, or you can choose a line mode directly by expanding the palette (click and hold/drag). You can also toggle between line modes using <code>Shift</code> + <code>Space</code> .
	Switch display of pads between filled and outline mode.
	Switch display of graphic items between filled and outline mode.
	Switch display of text between filled and outline mode.
	Switch the non-active layer display mode between Normal and Dim.  <b>Note:</b> this button will be highlighted when the non-active layer display mode is either Dim or Hide. In both cases, pressing the button will change the layer display mode to Normal. The Hide mode can only be accessed via the controls in the Appearance Panel or via the hotkey <code>Ctrl</code> + <code>H</code> .
	Toggle display of library and footprint tree.
	Show or hide the Appearance and Selection Filter panels on the right side of the editor.
	Show or hide the Properties Manager panel on the left side of the editor.

## Right toolbar tools

Placement and drawing tools are located in the right toolbar.

	Selection tool (the default tool).
	<a href="#">Add pad</a> : click on the board to place a pad.
	<a href="#">Add rule area</a> : Rule areas, formerly known as keepouts, can restrict the placement of items and the filling of zones and can also define named areas to apply specific custom design rules to.
	<a href="#">Draw lines</a> . <b>Note:</b> Lines are graphical objects and are not the same as tracks placed with the Route Tracks tool. Graphical objects cannot be assigned to a net.
	<a href="#">Draw arcs</a> : pick the center point of the arc, then the start and end points. By right clicking this button, you can change the arc editing mode between a mode that maintains the existing arc center and a mode that maintains the arc radius.
	<a href="#">Draw rectangles</a> . Rectangles can be filled or outlines.
	<a href="#">Draw circles</a> . Circles can be filled or outlines.
	<a href="#">Draw graphical polygons</a> . Polygons can be filled or outlined. <b>Note:</b> Filled graphical polygons are not the same as filled zones: graphical polygons cannot be assigned to a net and will not keep clearance from other items.
	<a href="#">Draw bezier curves</a> : draw a bezier curve. Each curve is defined by its start and end points and two control points. Subsequent curves start as tangent to the previous one. Use Backspace to cancel the previous point.
	Add <a href="#">bitmap image</a> for reference. Reference images are not included in fabrication outputs.
	<a href="#">Add text</a> .
	<a href="#">Add a textbox</a> .
	<a href="#">Add a table</a> .
 	<a href="#">Add dimensions</a> . Dimension types are described in more detail below.
	Deletion tool: click objects to delete them.
	Anchor tool. Left-click to set the anchor position (origin) of the footprint.
	<a href="#">Set grid origin</a> .
	<a href="#">Interactively measure</a> the distance between two points.

## Browsing, modifying, and saving footprints

The  button displays or hides the list of available libraries, which allows you to select an active library. When a new footprint is created, it will be placed in the active library.

Clicking on a footprint name opens that footprint in the editor, and hovering the cursor over the name of a footprint displays a preview of the footprint.

After modification, a footprint can be saved in the current library or a different library. To save the modified footprint in the current library, click the  button.

To save the footprint changes to a new footprint, click **File** → **Save As...** The footprint can be saved in the current library or a different library, and a new name can be set for the footprint.

To create a new file containing only the current footprint, click **File** → **Export** → **Footprint...** This file will be a standard footprint library which will contain only one footprint.

The editor can also open footprints from the board. To edit a footprint from the board, right click a footprint in the Board Editor and select **Open in Footprint Editor** ( + ). Alternatively, you can open a board footprint by pressing the  button in the Footprint Editor top toolbar and selecting the desired footprint.

Editing and saving the board copy of a footprint will only update that footprint in the board; it will not update other copies of that footprint in the board, and it will not change the original library copy of the footprint. When you open the board copy of a footprint, the Footprint Editor displays an info bar that warns you the library copy will not be modified. You can click the link in this info bar to open the library version of the footprint instead, or press  +  + .

## Creating a new footprint library

You can create a new footprint library by clicking **File** → **New Library...** This opens a file browser for you to choose a location and name for the new library. Footprint libraries are saved as a folder; the folder name is the library name plus the suffix `.pretty`.

The file browser also lets you choose whether the new library should be added to the global footprint library table or the project footprint library table. Libraries in the global library table will be available to all projects, while libraries in the project library table will only be available in the current project.

### NOTE

The global and project footprint library tables are managed using **Preferences** → **Manage Footprint Libraries...** This includes deleting and renaming footprint libraries. For more information about managing library tables, see the [footprint library table documentation](#).

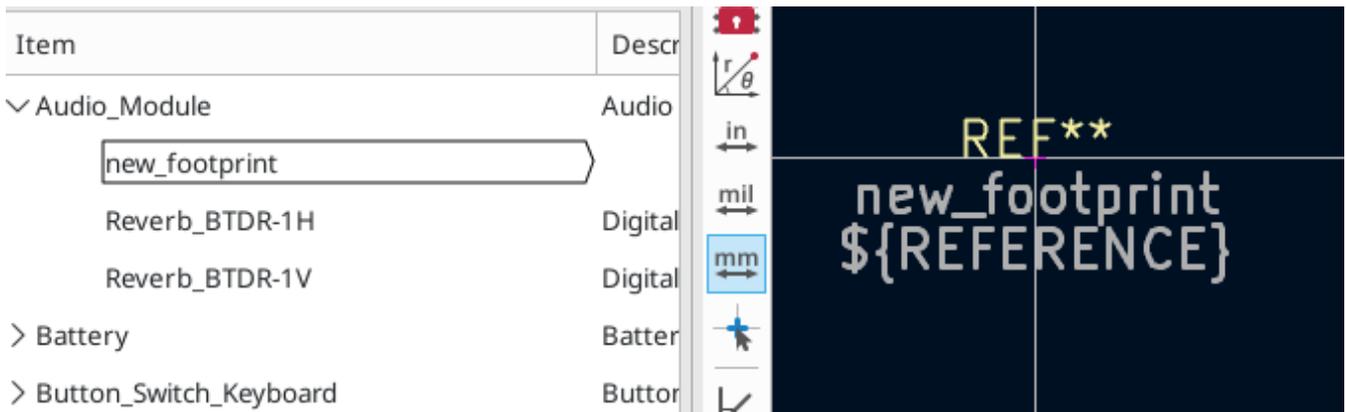
After choosing a name, location, and library table, a new, empty library is created. When you create new footprints, you can save them in this library.

## Creating a new footprint

To create a new footprint in the current footprint library, click the  button or click **File** → **New Footprint...** A new, untitled footprint will be created in the selected library. The new footprint is opened in the editor.

To set the name of the footprint, open its [properties dialog](#) ( button or **E**). The name will set the name of the footprint, which is used when assigning a footprint to a symbol, and is also used as the filename of the footprint file on disk.

The new footprint will be empty except for several default text items. The footprint contains two default (mandatory) footprint fields, `Reference` and `Value`. `Reference` contains the text `REF**`, which will be replaced with the reference designator of the footprint's corresponding symbol when the [footprint is added to the board](#). `Value` is initially set to `Untitled`, which can be changed, but this will also be updated with the contents of the corresponding symbol's `Value` field when the footprint is added to the board. Finally, there is a footprint text item containing the string `${REFERENCE}`, which is a [text variable](#) that will resolve to the value of the footprint's `Reference` field once the footprint is on a board.



These items are centered on the footprint's anchor (origin point), which is indicated with a magenta cross symbol. The anchor can be repositioned (changing the `(0, 0)` point for the footprint) by selecting the  button and clicking on the new desired anchor position.

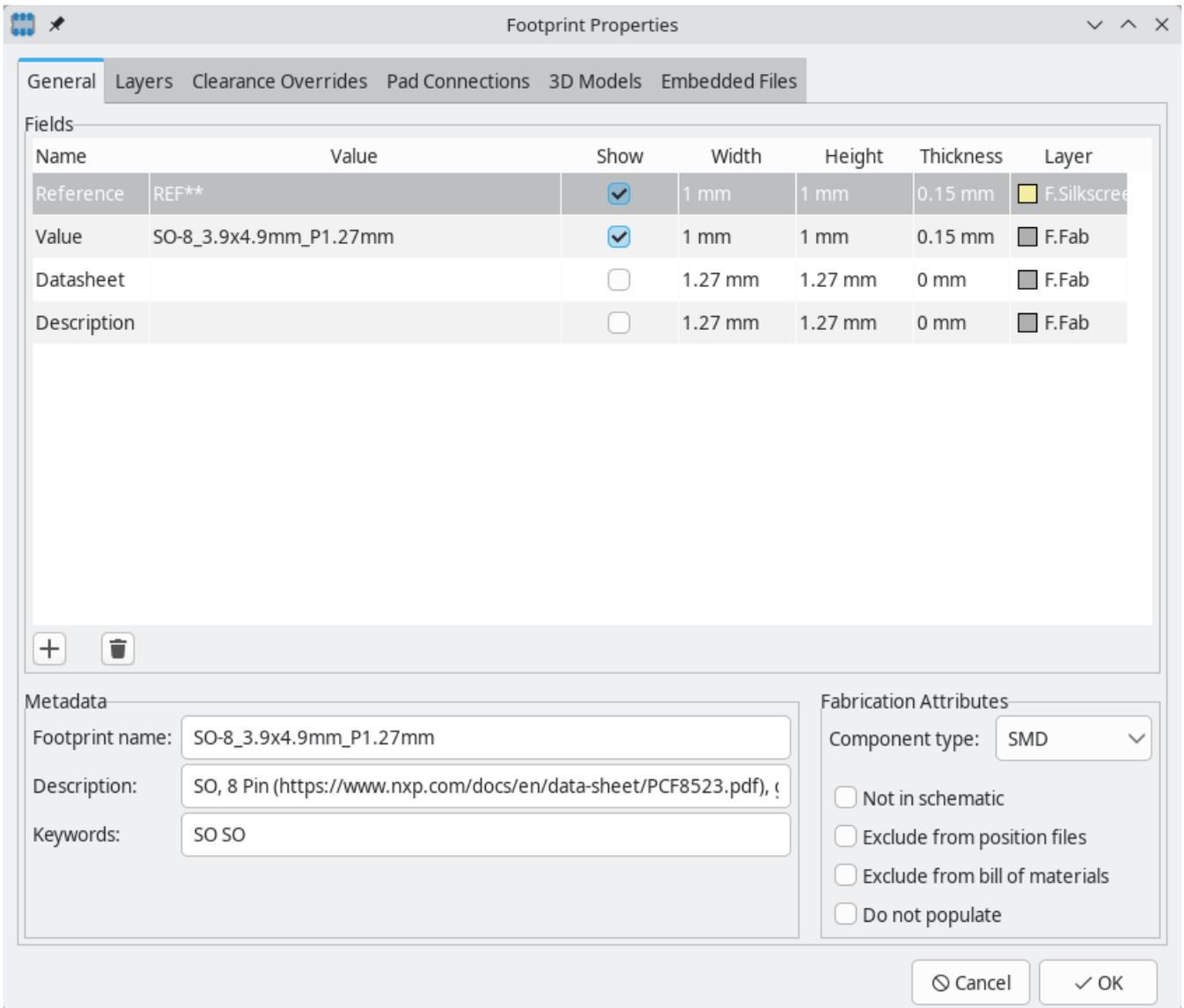
**NOTE** Rather than manually creating a footprint, for some common footprints you can use a [footprint wizard](#) to create a footprint based on a set of parameters.

## Editing footprint properties

Footprints have a number of properties and metadata items that can be defined. These include text fields, attributes that can be set or not (such as Do Not Populate), clearance and zone connection settings, and 3D model paths. These are initially defined in the library copy of the footprint, but they can be modified on a per-instance basis once a footprint is added to a board. In other words, two copies of the same footprint on a single board can have their properties edited separately.

Some properties, namely text fields and attributes, will be automatically set for each footprint in a board based on the fields and attributes in the footprint's corresponding schematic symbol. Fields and attributes are synced from symbols to footprints when you perform the [Update PCB From Schematic](#) action. They are also synced from footprints back to symbols when you perform the [Update Schematic From PCB](#) action.

To edit footprint properties, click the  button to show the Footprint Properties dialog. You can also double click an empty spot in the editing canvas.



The most important settings are in the **General** tab. Here you can edit [footprint fields](#) as well as many basic properties for the footprint. Other tabs contain options for [footprint layers](#), [clearance overrides](#), [pad connections](#) including [net ties](#) and [jumper pads](#), [3D models](#), and [embedded files](#).

Footprint fields are explained in a [separate section](#). The other settings in the General tab are:

- **Footprint name** is the footprint's name in the library. Footprints are identified by a combination of the library and footprint name. The footprint name is the same as the footprint's filename on disk, and is also initially the same as the footprint's `Value` field. However, the `Value` field can be edited in the footprint editor, and when a footprint is added to a board, its `Value` field will be updated with the value of the footprint's corresponding symbol.
- **Description** is a description of the footprint. It should be human readable, but it is also used when searching for a footprint.

**NOTE**

This description property is specifically a description of the **footprint**. This is not to be confused with the `Description` field, which will be set to the description of the footprint's corresponding symbol when the footprint is added to a board.

- The **keywords** should contain additional terms related to the footprint. Keywords are primarily used, in combination with the footprint name and the `Description` field, for searching for the footprint in the

Footprint Chooser and the Footprint Editor. Those three items are also displayed when you select a footprint in the Footprint Chooser.

- **Component type** is one of SMD, Through hole, or Unspecified. A footprint's type affects KiCad's behavior in a few ways:
  - Footprint type can be used to filter footprints from component placement files as well as other exports, such as STEP files. Additionally, the footprint type is included as metadata in IPC-2581 exports.
  - Footprint 3D models can be shown and hidden in the 3D viewer based on their type. For example, SMD models can be hidden while through hole models are still displayed.
  - Footprints of different types are reported separately in the [Board Statistics dialog](#).
  - DRC and the [footprint checker](#) will report footprints containing pads that do not match the parent footprint's type, for example through hole pads in an SMD footprint.
- **Not in schematic, Exclude from position files, Exclude from bill of materials, and Do not populate** set the default state of these attributes when the footprint is added to the board. These attributes are overridden by the value of the attributes in the footprint's linked symbol.
  - If **not in schematic** is checked, KiCad will not expect the footprint to correspond to a symbol in the schematic. When updating a PCB from the schematic, KiCad will ordinarily remove footprints that don't have corresponding symbols according to the **delete footprints with no symbols** setting. However, such footprints will not be deleted when they have **not in schematic** set.
  - If **exclude from POS files** is checked, KiCad will not include the footprint in component placement file exports.
  - If **exclude from bill of materials** is checked, the component will not be included in bill of materials exports in either the schematic or PCB editors. This attribute is synced to and from the footprint's corresponding schematic symbol.
  - The **do not populate** attribute is primarily a schematic symbol attribute, and is synced to and from the footprint's corresponding schematic symbol. Footprints with this attribute set can optionally be excluded from component placement file exports and some other types of outputs. These footprints can also be hidden in the 3D viewer.

## Footprint pads

Pads are added to a footprint by clicking the  button in the right toolbar, then clicking again in the desired location in the canvas. The tool continues adding new pads each time you click on the canvas until you cancel the tool (). Each new pad has its pad number incremented by one relative to the previous pad number.

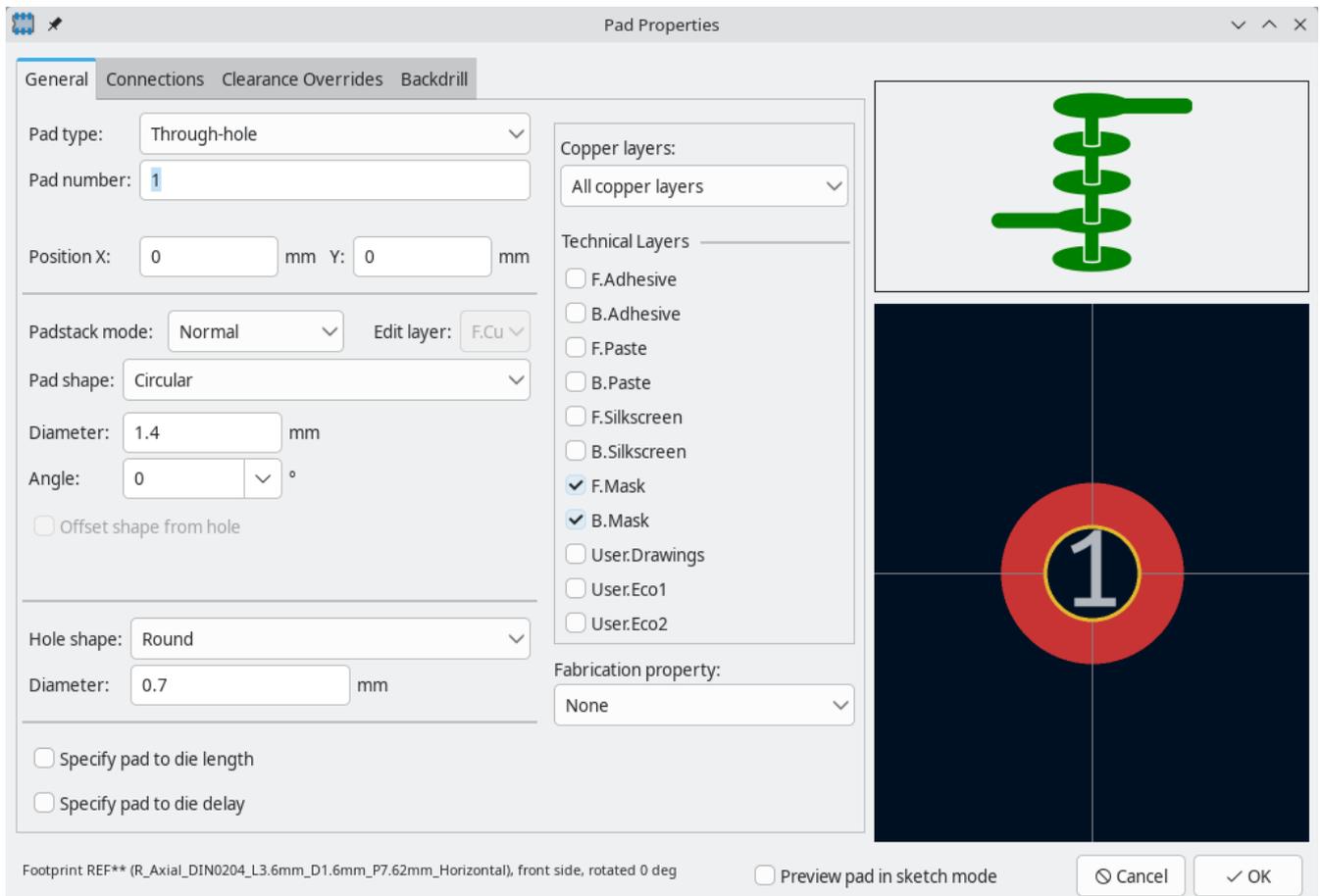
You can configure basic surface-mount or through-hole pad shapes in the [Pad Properties](#) dialog. Some pads may require designing [custom-shaped pads](#) or [postmachining/backdrilling](#).

KiCad also offers tools to help make it easier to design footprints containing many pads, including [default pad properties](#), a [pad renumbering tool](#), a [pad table](#), and an [array tool](#).

## Editing pad properties

You can edit a pad after adding it by opening the pad's properties dialog (E). These properties are also editable using the [Properties Manager](#).

The most frequently edited properties of a pad are in the **General** tab. These include the pad's position, geometry, and layer settings. Other tabs let you configure [zone](#), [thermal](#), and [teardrop connections](#) to the pad, [electrical clearance and solder mask / paste expansion](#) overrides, and [post-machining](#) and [backdrilling](#).



The pad settings in the General tab are:

**Pad type** controls which features are enabled for the pad.

- **SMD** pads are electrically-connected and have no hole. In other words, they exist on a single copper layer.
- **Through-hole** pads are electrically-connected and have a plated hole. The hole exists on every layer, and the copper pad exists on multiple layers (see **Copper layers** setting below).
- **Edge Connector** pads are SMD pads that are allowed to overlap the board outline on the `Edge.Cuts` layer.
- **NPTH**, **Mechanical** pads are non-plated through holes that do not have an electrical connection.
- **SMD Aperture** pads are pads that have no hole and no electrical connection. These can be used to add specific designs to a technical layer, for example a paste or solder mask aperture.

The **Copper layers** setting controls which copper layers will have a shape associated with the pad.

For SMD pads, the options are `F.Cu` or `B.Cu`, controlling whether the pad sits on the front or the back of the board *relative to the footprint's location*. In other words, if a pad is set to exist on `B.Cu` in its properties, and the footprint is flipped to the back of the board, *that pad will now exist on `F.Cu`, because it also has been flipped*.

- For through-hole pads, it is possible to remove the pad shape from copper layers where the pad is not electrically connected to other copper (tracks or filled zones). Setting the copper layers to **connected layers only** will remove the pad shape from any unconnected layers, and setting to `F.Cu`, `B.Cu`, and **connected layers** will remove the pad shape from any internal unconnected layers. This can be useful in dense board designs to increase the routable area on internal layers.

The **Technical layers** checkboxes control which technical layers will have an aperture added with the pad's shape. By default, pads have apertures on the paste and mask layers matching their copper layer.

- For SMD pads, the mask and paste apertures are on whichever outer layer matches the pad's copper layer.
- For through-hole pads, mask apertures are on both `F.Mask` and `B.Mask` by default, with no paste apertures. The front and back mask and paste layers can be enabled or disabled independently.

#### NOTE

The solder mask expansion and solder paste margin values in the **Clearance Overrides** tab apply to all of the pad's enabled mask and paste layers.

The **Pad number** controls what the pad will be electrically connected to in the board. A pad has the same net connection as the pin with the same number in the corresponding schematic symbol.

**Pad Position X** and **Y** are the location of the center of the pad, relative to the footprint's origin.

A pad's diameter and hole size can be defined on a per-layer basis. This is also known as defining the pad's *padstack*. The **Padstack mode** controls whether the pad shape is the same on all layers or whether individual layers are individually controlled.

- In the **Normal** padstack mode, the pad's diameter and hole size are the same on all layers.
- In the **Front/Inner/Back** padstack mode, the pad's diameter and hole size can be controlled separately for the front layer, the back layer, and the inner layers (the inner layers will all have the same settings). The **Edit layer** dropdown controls which layer (or group of layers) is currently being displayed and edited.
- In the **Custom** padstack mode, the via's diameter and hole size can be controlled completely independently on each layer. The **Edit layer** dropdown controls which layer is currently being displayed and edited.

**Pad shape** controls the basic shape of the pad. Pad shapes can be one of:

- circular
- oval
- rectangular
- trapezoidal

rounded rectangle

- chamfered rectangle
- chamfered with other corners rounded
- custom (circular base)
- custom (rectangular base)

Each pad shape has its own set of options; for example, rounded rectangles have settings for **pad size X** and **Y**, **angle**, **corner size**, and **corner radius**.

**NOTE**

The size of a pad can also be adjusted interactively in the canvas by dragging the editing handles at the pad corners.

**NOTE**

Some components may require pads with unusual shapes that cannot be configured in this dialog. You can create [custom pad shapes](#) for such components.

Through-hole and NPTH pads have a hole in addition to the pad itself. The **hole shape** can be **circular** or **oval**, with corresponding size controls. By default the pad is centered on the hole, but the pad can be offset relative to the hole if the **offset shape from hole** option is enabled (circular pads cannot be offset from the hole).

**Fabrication properties** are primarily used as metadata in Gerber X2 fabrication output, where the fabrication property is included as an aperture attribute for each pad. Some properties also affect DRC. The following fabrication properties are available:

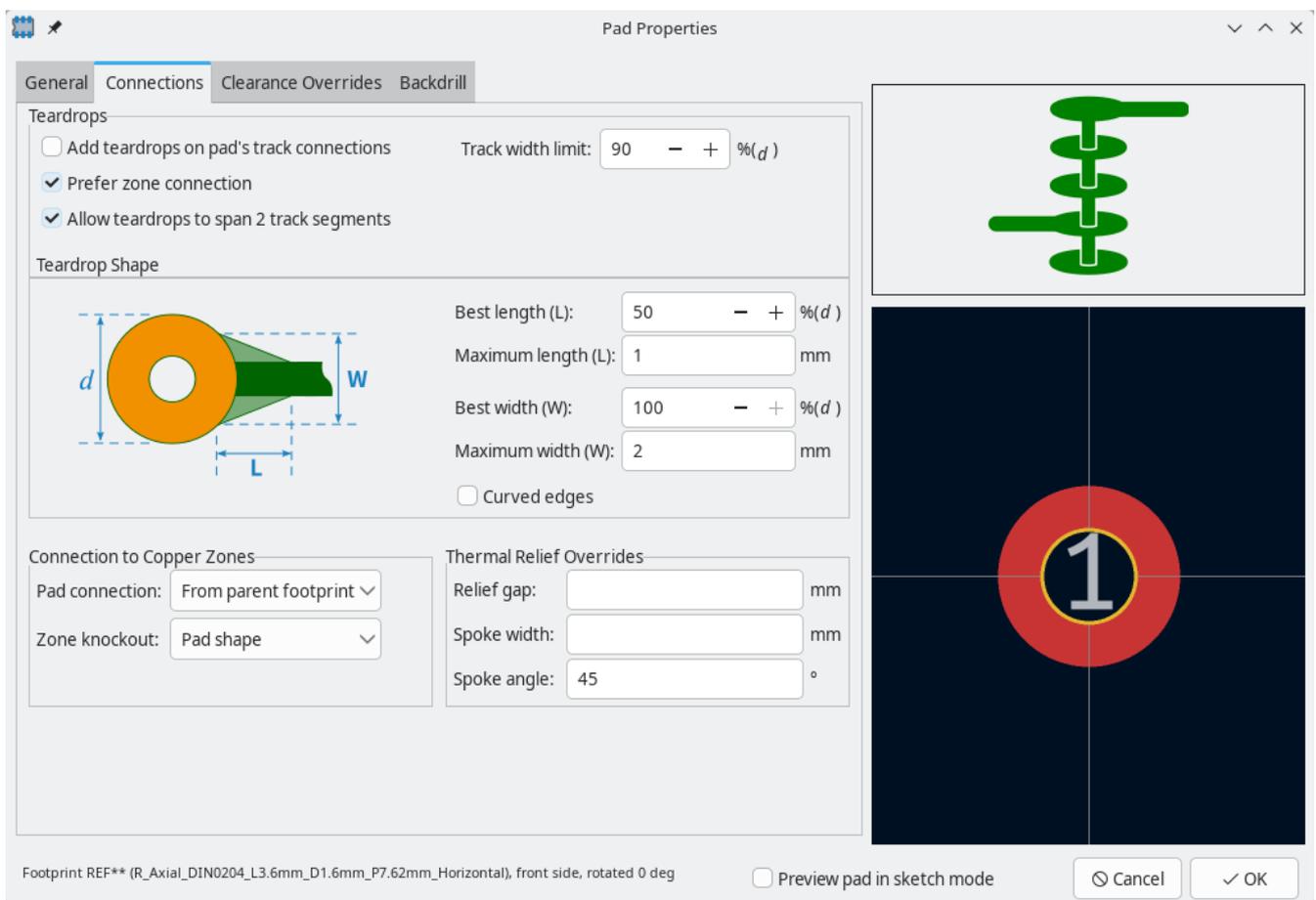
- **BGA pad** can only be applied to SMD pads, and only affects Gerber X2 output.
- **Fiducial, local to footprint** and **fiducial, global to board** only affect Gerber X2 output.
- **Test point** can only be applied to SMD or through hole pads, can only be applied to pads on outer layers, and only affects Gerber X2 output.
- Through hole pads with the **heatsink pad** property are allowed in SMD footprints (PTH pads without this property cause a DRC violation when they are used in SMD footprints). It also affects Gerber X2 output.
- The **castellated pad** property is for pads that intentionally intersect the board edge such that they will be bisected when the board is manufactured. Pads with this property are allowed to intersect the board edge and still be routed (it is otherwise a DRC error for a pad to intersect the board edge, which makes routing impossible). In STEP exports, pads with this property are clipped to the board edge. This property also affects Gerber X2 output.
- Through hole pads with the **mechanical** property can be used in SMD footprints without causing a DRC violation. This can be used for mounting pads or other mechanical through hole pads in surface mount footprints. This is similar to the **heatsink pad** property, but does not affect Gerber X2 output.
- The **press-fit pad** property is for pads designed to have component leads pressed into them without solder. This property only affects Gerber X2 output.
- **None** is for pads for which none of the other fabrication properties apply. It has no effect.

**Specify pad to die length:** This setting allows a length to be associated with this pad that will be added to the routed track length by the track length tuning tools and the Net Inspector. This can be used to specify internal bondwire lengths for more accurate length matching, or in other situations where the electrical length of a net is longer than the length of the routed tracks on the board.

**Specify pad to die delay:** This setting is similar to the pad-to-die length, but for time-domain tuning. It allows a time delay to be associated with this pad that will be added to the routed track delay by the track length tuning tools.

## Pad connections

The **Connections** tab of a pad's properties contains settings for how pads connect to other objects, including settings for teardrops, zone connections, and thermal reliefs.



The Teardrops section contains settings controlling teardrop connections between tracks and the pad, if teardrops are used. Teardrop settings are explained in the [teardrop documentation](#).

**Pad connection** controls whether the pad will have a solid, thermal relief, or no connection to the zone. Like the other overrides, this one may be set for an individual pad or for an entire footprint. The default setting for this control is **From parent footprint**, and the default footprint setting is to use the connection mode specified in the zone properties.

**Zone knockout** controls the behavior of the zone filler when the pad uses a custom shape rather than one of the default shapes. This can be used to achieve different results when using thermal reliefs and custom pad shapes.

**Relief gap** controls the length of the thermal spokes, or the gap between the pad's shape and the filled copper area of the zone. This value is normally empty which will cause the relief gap to be inherited from the connecting zone's settings.

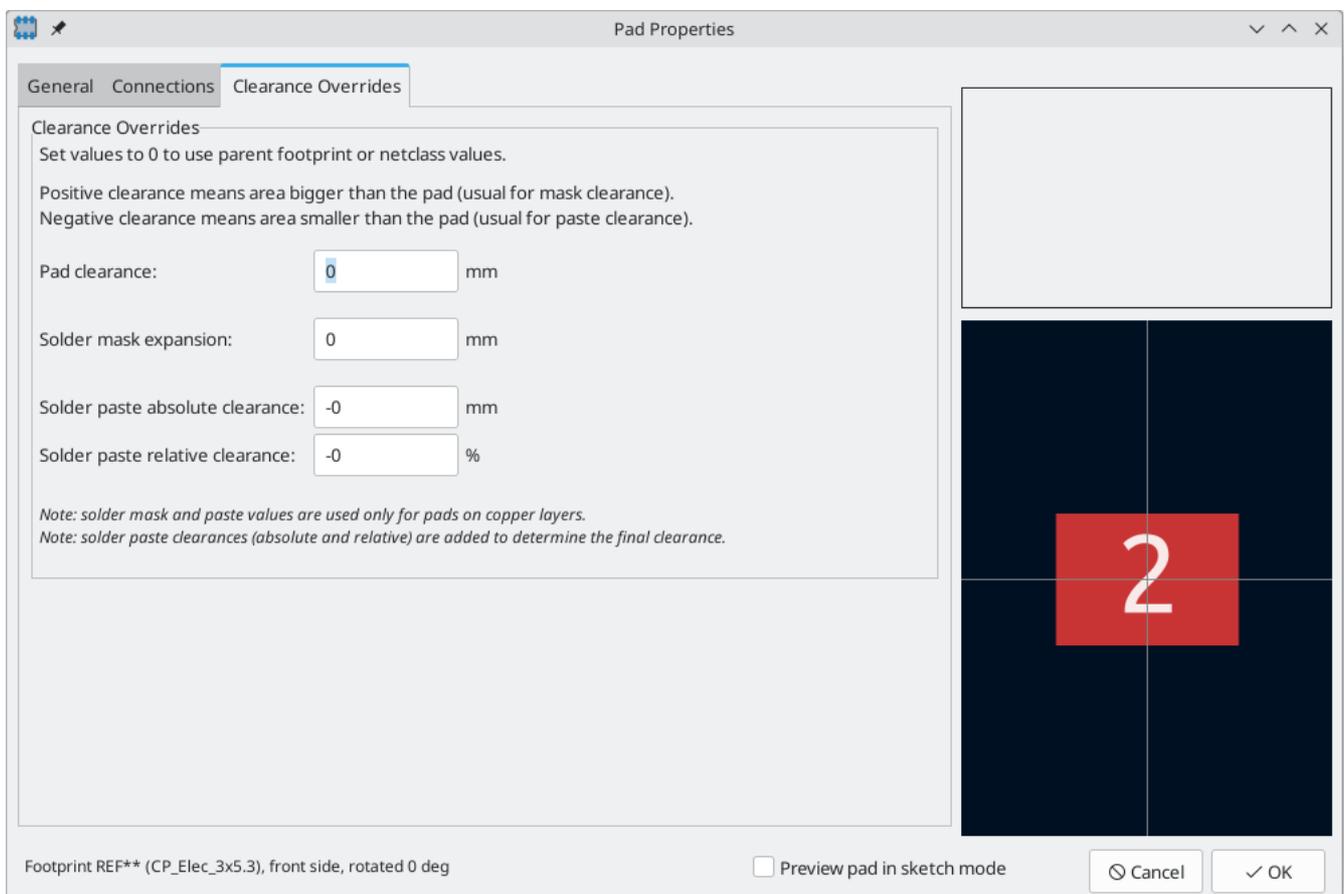
**Spoke width** controls the width of the spokes generated when the zone connection mode is **Thermal Relief**. This value is normally empty which will cause the spoke width to be inherited from the connecting zone's settings.

**NOTE**

Prior to KiCad version 9, a relief gap or spoke width of 0 caused that value to be inherited. In KiCad 9 and later, a relief gap or spoke width of 0 sets that value to 0, while a blank value causes the value to be inherited.

## Pad clearance overrides

The **Clearance Overrides** tab of a pad's properties holds settings for pad-specific overrides to board clearance and mask/paste expansion.



**Pad clearance** controls the minimum clearance between the pad and any copper shape (tracks, vias, pads, zones) on a different net. This value is normally empty which will cause the pad clearance to be inherited from any clearance override set on the footprint, or the board's design rules and netclass rules if the footprint clearance is also empty.

**NOTE**

Prior to KiCad version 9, a pad clearance of 0 caused the pad clearance value to be inherited. In KiCad 9 and later, a pad clearance of 0 sets the clearance to 0, while a blank pad clearance causes the clearance to be inherited.

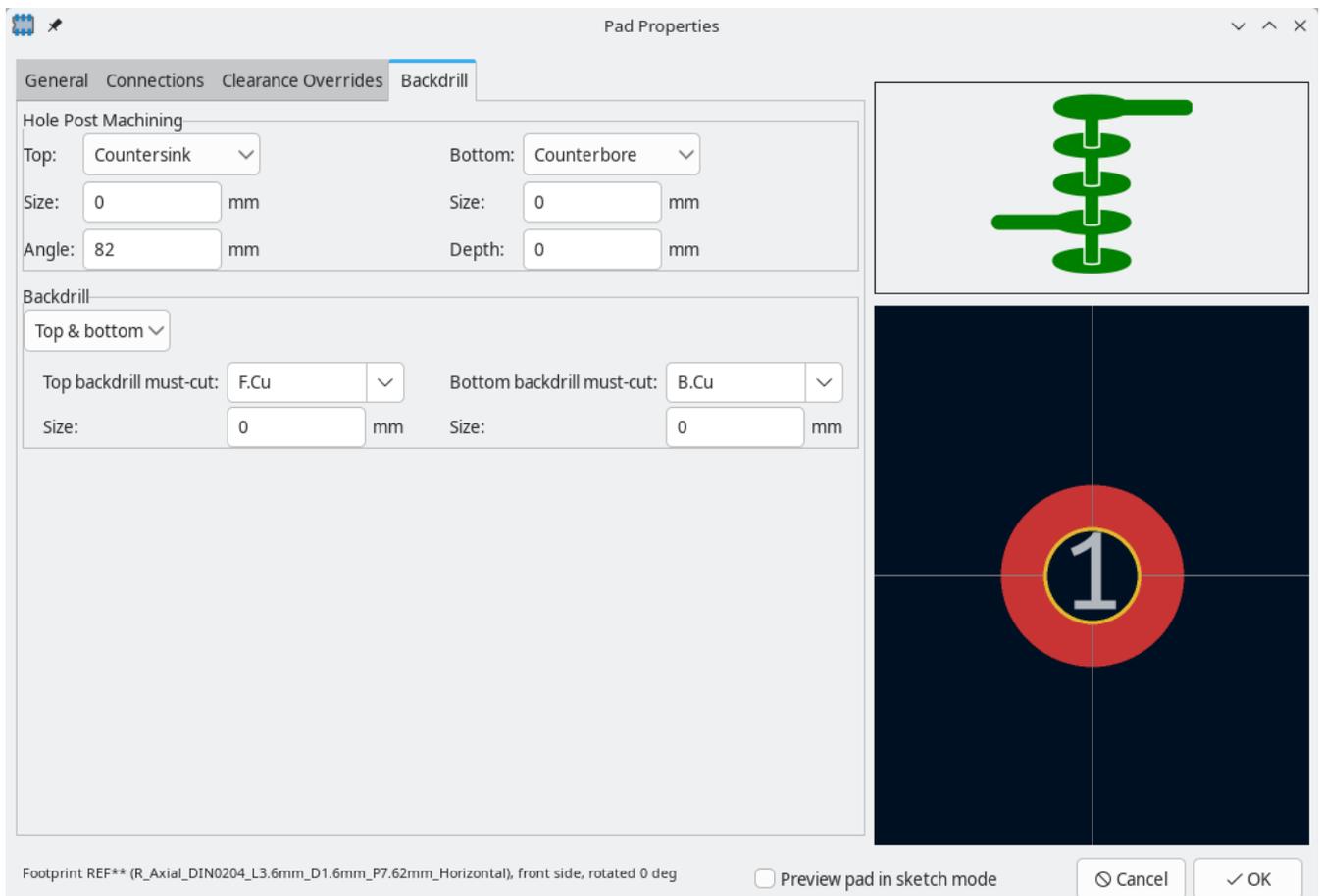
The aperture appearing on any technical layer will have the same shape and size as the pad shape on the copper layer(s). In the PCB manufacturing process, the manufacturer will often change the relative size of mask and paste apertures relative to the copper pad size, but since this size change is specific to a manufacturing process, most manufacturers expect the design data to be provided with the apertures set to the same size as the copper pads. For specific situations where you need to oversize or undersize a technical layer aperture in the design data, you can use the settings here.

**Solder mask expansion** controls the size difference between the pad shape and the aperture shape on the F.Mask and B.Mask layers. A positive number means the solder mask aperture will be larger than the copper shape. This number is an inflation applied to all directions. For example, a value of 0.1mm here will cause the solder mask aperture to be inflated by 0.1mm, meaning that there will be an 0.1mm border on all sides of the pad and the solder mask opening will be 0.2mm wider than the pad when measured along a given axis.

**Solder paste clearance** is a setting to specify the solder paste shape relative to the parent pad size (the size difference between the pad shape and the aperture shape on the F.Paste and B.Paste layers). This can be specified as an absolute offset from the pad edge (e.g. -0.1mm), a value relative to the pad dimension (e.g. -5%), or both (e.g. -0.1mm - 5%). If it is 0 or blank, the solder paste aperture will be the same size as the pad. Positive values mean solder paste aperture larger than the pad. Negative values mean solder paste aperture smaller than the pad.

## Pad post-machining and backdrilling

You can control post-machining and backdrill operations for pads in the **Backdrill** tab of the pad's properties.



[Post-machining](#) and [backdrilling](#) are explained in their own sections.

## Custom pad shapes

For some footprints, the built-in pad shapes (round, rectangular, etc.) may not be sufficient. In these cases you can construct custom pads with arbitrary shapes using **Pad Edit Mode**. This mode lets you combine a basic pad with graphic shapes to build a new pad out of the compound shape.

To build a custom pad, first add a regular pad using the pad tool ( button). This base pad will become the custom pad's anchor or snapping point, so be sure to place the pad in the exact location where you want tracks to attach to the pad. The shape and size of the pad do not matter, but the hole, if any, will remain in the final custom pad. In other words, a surface mount base pad will result in a surface mount custom pad, and a through hole base pad will result in a through hole custom pad. The custom pad's number will be inherited from the base pad.

Next, enter Pad Edit Mode by selecting the base pad, right-clicking, and selecting **Edit Pad as Graphic Shapes** ( + ). Add graphic shapes as appropriate to create the desired pad shape. Shapes touching the base pad will be unioned together with the base pad to create the final pad shape.

You can exit Pad Edit Mode by right-clicking and selecting **Finish Pad Edit**, or pressing  +  again. When you exit pad edit mode, all shapes that touch the base pad will be combined with the pad. For example, when editing a surface mount pad on `F.Cu`, any shapes that are on `F.Cu` and touch the base pad will become part of the custom pad. Any shapes that do not overlap the base pad, or that are on a different layer, will remain separate. If the base pad is a through hole pad, overlapping shapes on `F.Cu` will be combined in the custom pad. Because through hole pads have the same pad shape on all copper layers, this shape will become part of the custom pad on all copper layers, not just `F.Cu`. For convenience, Pad Edit Mode dims the color of other pads and all shapes that are not contiguous with the base pad so that you can see which shapes will be included in the custom pad and which will not.

Custom pads can only contain a single base pad. Any additional pads that touch the base pad or the contiguous graphics, whether they have the same or different pad numbers as the base pad, will remain separate pads after the shapes are combined into the custom pad.

### NOTE

If you would like to add multiple anchors (snapping points) to a custom pad, you can add additional separate pads on top of the custom pad. Create the custom pad as normal, containing the first snapping point, then add additional pads with the same number and place them overlapping the custom pad in the desired snapping locations. They will remain distinct pads and will not be combined with the custom pad, but they will act as additional pad anchors and will be electrically connected to the custom pad.

To modify an existing custom pad, select it and enter Pad Edit Mode again. You can then continue to edit the component shapes to adjust the pad shape, or change the position of the base pad to adjust the pad anchor.

KiCad automatically chooses a size and location for showing the pad number over the pad. Particularly for unusually shaped pads, the automatically determined size and location may not be optimal. In these cases, you can manually specify a region in which KiCad should draw the pad number by adding a pad **number box** primitive. To add a number box, enter Pad Edit Mode and add a rectangular shape. In the Properties Manager for the rectangle, check the **Number Box** checkbox. The rectangle will then be shown as a wireframe, and when you exit Pad Edit Mode it will be used to draw the pad number.

In the board, KiCad will automatically add thermal spokes when connecting the pad to a zone. The thermal spoke settings are determined by the pad, footprint, and zone settings, and the thermal spokes by default

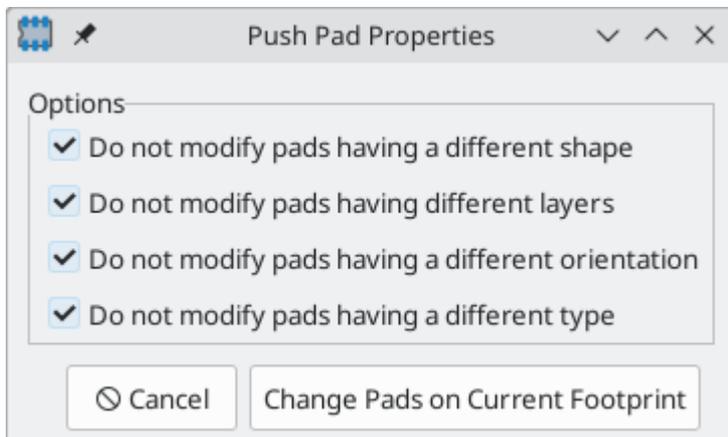
connect to the pad anchor. You can override the default thermal spoke placement by adding **thermal relief templates** to the custom pad. To add a thermal relief template, enter Pad Edit Mode and add a line shape. In the Properties Manager for the line, check the **Thermal Relief Template** checkbox. In Pad Edit Mode, the line will then be shown as a wireframe, and it will not be shown outside of pad edit mode. If any thermal relief templates are present in the pad, KiCad will not automatically add additional spokes when filling zones; spokes will only be placed where there are thermal relief templates defined in the pad. Thermal relief templates only determine the spoke location: spoke width and relief gap are still defined in the pad, footprint, and/or zone properties, as normal.

## Default pad properties

When you place a new pad, the new pad's properties are copied from the **default pad properties**. Each time any pad is edited, the pad's updated properties are stored as the default pad properties, so that new pads will match the properties of the most recently edited pad.

You can directly edit the default pad properties by selecting **Edit** → **Default Pad Properties...**, or choose an existing pad to represent the default by right clicking the pad and choosing **Copy Pad Properties to Default**. New pads will use that pad's properties as their defaults until a new default is selected, either by editing another pad, editing the default pad properties, or manually copying a different pad's properties to the default.

There are several ways to update existing pads with the properties of other pads. You can apply the default pad properties to an explicit selection of pads by selecting the desired target pads, right clicking, and choosing **Paste Default Pad Properties to Selected** from the right click context menu. You can also update other pads with a selected pad's properties using **Push Default Pad Properties to Other Pads...**, also in the right click context menu.



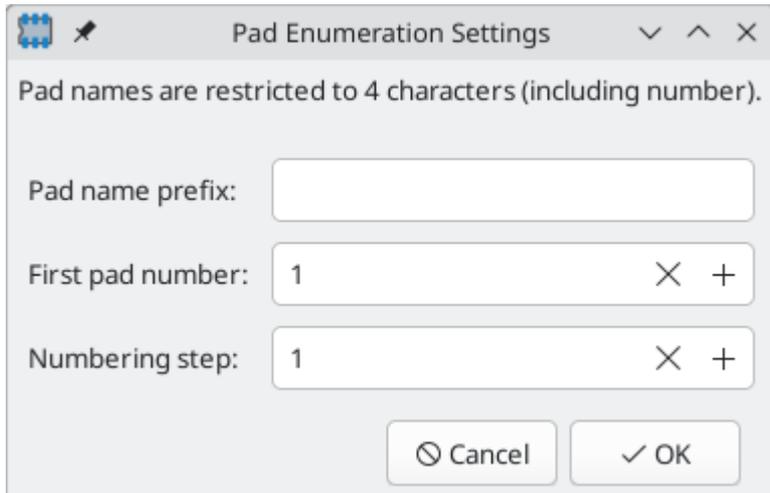
This tool has several options to filter which pads are targeted.

- If **do not modify pads having a different shape** is selected, only pads with the exact same shape properties as the selected pad will be updated.
- If **do not modify pads having different layers** is selected, only pads on the same layer(s) as the selected pad will be updated.
- If **do not modify pads having a different orientation** is selected, only pads with the same orientation as the selected pad will be updated.
- If **do not modify pads pads having a different type** is selected, only pads with the same pad type as the selected pad will be updated.

- If no options are selected, all pads in the footprint will be updated.

## Renumbering pads

You can quickly renumber existing pads using the Renumber Pads tool (**Edit** → **Renumber Pads...**).



The tool has several options. Pads will be renumbered starting at the selected **first pad number**, and each subsequent pad will have its number incremented by the **numbering step**. You can also choose an optional **pad name prefix** which will be inserted before of the incrementing part of each pad number.

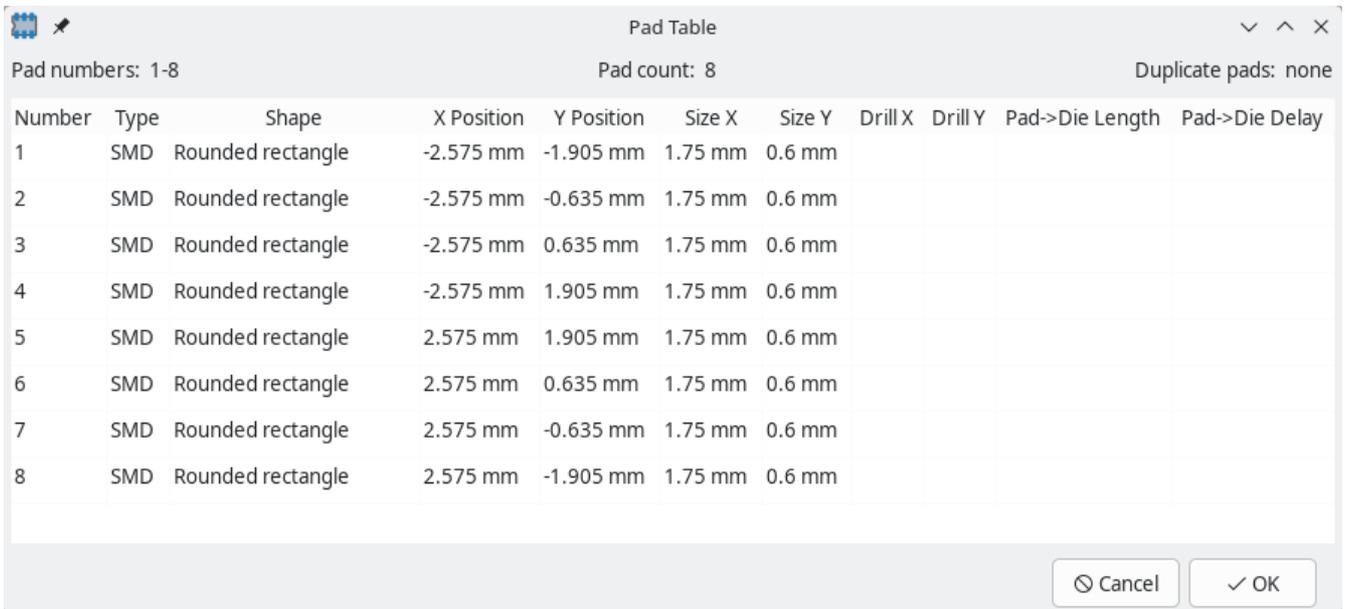
Once you click **OK**, you will be prompted to click on a pad, which will be assigned a new pad number based on the selected initial pad number and prefix. You can keep clicking on pads to assign them the next number in the sequence based on the selected numbering step. Double click on a pad to renumber that pad and end the sequence, or press Esc to discard the changes.

## Pad table

Another way to edit pads is to use the Pad Table, which is accessible via the  button. The Pad Table displays all of the pads in the footprint and their properties in a table view, so it is useful for making bulk pad changes. Any pad property can be edited by clicking on the appropriate cell.

### NOTE

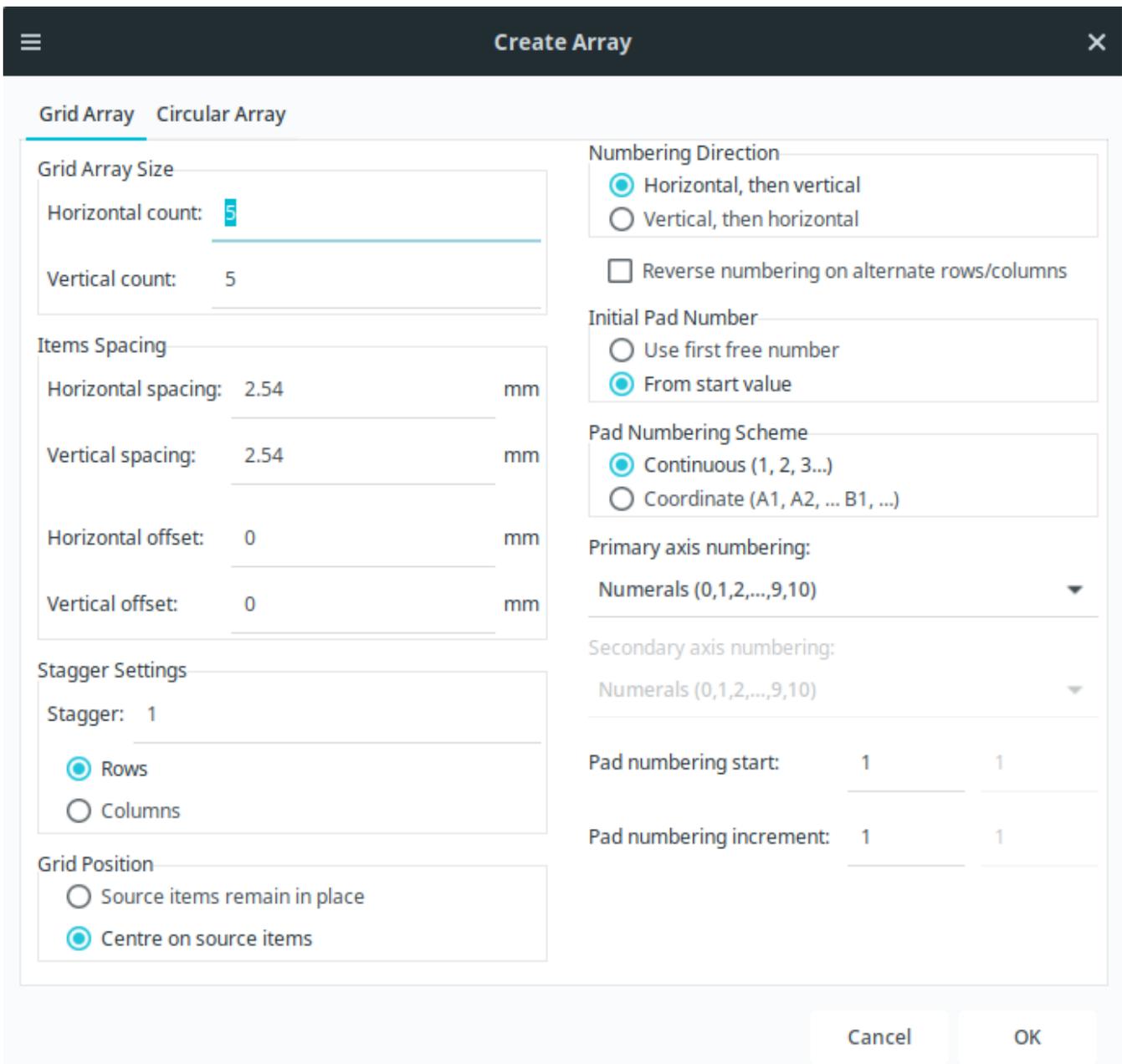
Columns of the pad table can be shown or hidden by right-clicking on the header row and checking or unchecking additional columns.



## Pad arrays

You can create an array of pads from a source pad by right clicking the source pad and selecting **Create from Selection** → **Create Array...** (**Ctrl** + **T**).

This array tool can also be used for creating arrays of other objects, as described in the [PCB Editor documentation](#). For pads, however, there are additional options for controlling pad numbering.



For grid arrays, you can select a numbering direction, either **horizontal, then vertical** or **vertical, then horizontal**. If **reverse numbering on alternate rows/columns** is selected, the direction of increasing pad numbers will alternate from one row/column to the next.

The initial pad number in the array can either be the first unused pad number in the footprint (**use first free number**) or the specified **pad numbering start** value. After the first number, the pad numbering can either be **continuous** (1, 2, 3, ...) or **coordinate** based, in other words, dependent on both the row and column (A1, A2, ..., B1, ...). In addition to the initial pad number (**pad numbering start**), you can specify a numbering step (**pad numbering skip**). For coordinate-based numbering, you can configure separate starting numbers and steps for each axis. You can select whether pad numbers use decimal digits (0-9), hexadecimal digits (0-F), the full alphabet, or the alphabet excepting certain ambiguous letters (I, O, S, Q, X, and Z).

## Footprint graphics and text

Footprints can contain graphic shapes, text, and dimensions. These objects can be placed on nonphysical layers, like `F.Fab` or `User.Drawings`, or they can be placed on layers that will be part of the manufactured

circuit board, such as `Edge.Cuts` or a silkscreen, soldermask, or copper layer. Objects on copper layers can make electrical connections.

Closed shapes on a footprint's `F.Courtyard` and `B.Courtyard` layers will form the footprint's front and back courtyard, respectively. A courtyard defines the physical extents of a footprint and limits where footprints are allowed to be placed in relation to other footprints. If a footprint's courtyard overlaps another footprint's courtyard, a DRC violation will be generated.

Shapes on a footprint's `Edge.Cuts` layer will correspond to board edges on any PCB that includes the footprint. Closed shapes will result in cutouts, while unclosed shapes will result in unclosed edges. Unclosed edges must be closed in the full board design.

The buttons on the right toolbar can be used to create:

- Lines ( , default hotkey `Ctrl + Shift + L` )
- Arcs ( , default hotkey `Ctrl + Shift + A` )
- Bezier curves ( , default hotkey `Ctrl + Shift + B` )
- Rectangles (  )
- Circles ( , default hotkey `Ctrl + Shift + C` )
- Polygons ( , default hotkey `Ctrl + Shift + P` )
- Text ( , default hotkey `Ctrl + Shift + T` )
- Textboxes (  )
- Tables (  )
- Dimensions ( , of which several types are available

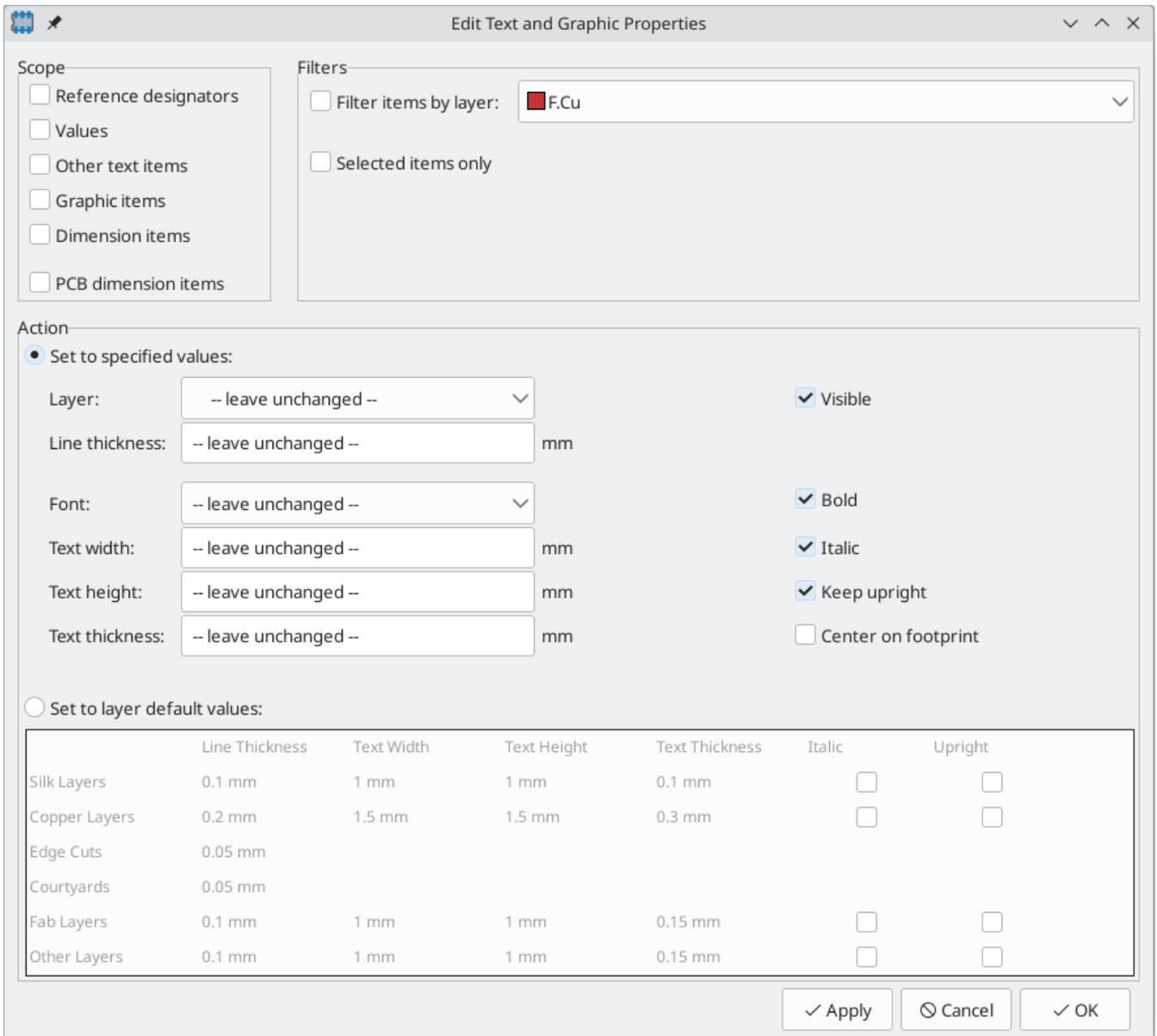
**NOTE**

You can customize the default style of newly-created text and shape objects in **Preferences** → **Footprint Editor** → **Default Values**.

Graphical objects and their properties are described in more detail in the [PCB Editor documentation](#).

## Bulk editing footprint text and graphics

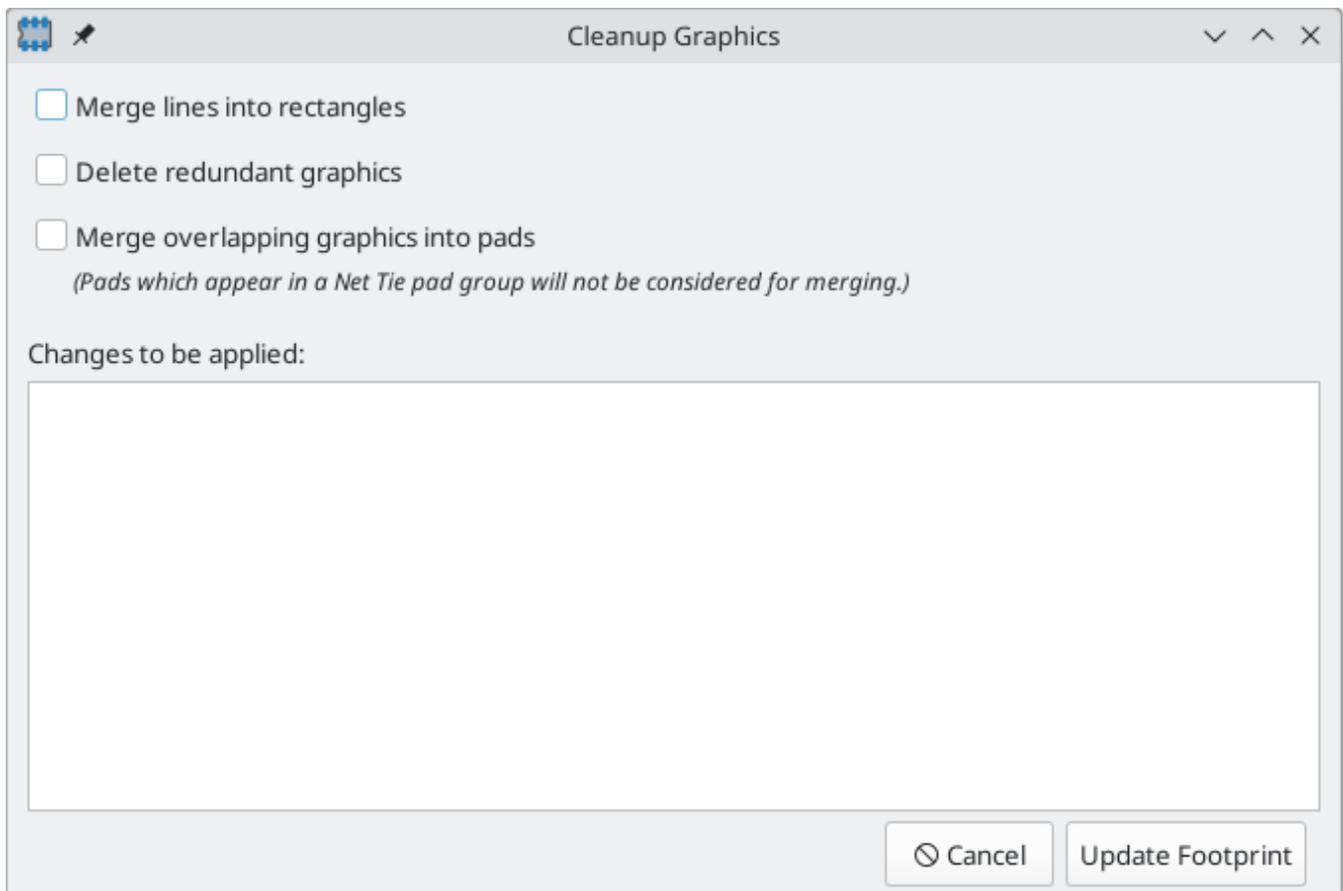
Properties of text and graphics can be edited in bulk using the **Edit Text and Graphics Properties** dialog (**Edit** → **Edit Text & Graphic Properties...**).



This dialog is described in more detail in the [PCB Editor documentation](#).

## Cleaning up footprint graphics

There is a dedicated tool for performing common cleanup operations on graphics, which is run via **Tools** → **Cleanup Graphics...**



The following cleanup actions are available and will be performed when selected:

**Merge lines into rectangles:** combines individual graphic lines that together form a rectangle into a single rectangle shape object.

**Delete redundant graphics:** deletes graphics objects that are duplicated or degenerate.

**Merge overlapping graphics into pads:** merges graphic copper shapes that overlap pads into a [custom pad](#).

Any changes that will be applied to the footprint are displayed at the bottom of the dialog. They are not applied until you press the **Update Footprint** button.

## Footprint fields

Footprints contain multiple fields, which are named values containing information related to the footprint. Fields can be visible and shown on any board layer, or they can be hidden and only shown in the footprint's properties. Some fields have special meaning to KiCad: `Reference` and `Footprint` are both used by KiCad to identify schematic symbols and PCB footprints, for example. Other fields may contain information that is important for a design but is not interpreted by KiCad, like pricing or stock information for a part.

Any fields defined in a library footprint will be included in the footprint when it is added to a board. You can also add new fields to footprints in the board. Whether they are in the library footprint or not, these fields can then be edited on a per-footprint basis in the board. Symbol fields are also transferred to the board and added as fields in the corresponding footprint.

**NOTE**

Footprint fields are different than graphic text. Fields are named, i.e. they have both a name ( `Reference` ) and a value ( `R101` ), whereas footprint text only has a value. Fields can be added to and deleted from footprints in a board in the Footprint Properties dialog, while text items can only be added to a footprint in the footprint editor. Fields are also synced between footprints and their corresponding symbols in the schematic. Before KiCad version 8.0, footprints did not have named fields, only graphic text.

All library footprints are defined with four default fields which correspond to the [default fields in library symbols](#): `Reference`, `Value`, `Datasheet`, and `Description`. These default fields cannot be deleted. The `Reference` field initially has the value `REF**`, while the `Value` field is initially set to the name of the footprint. In the board, the values of the four default fields will be set to the values of the matching fields in the footprint's corresponding symbol.

**NOTE**

The `Description` footprint field is the description of the symbol, not the footprint, and will be overwritten by the value of the corresponding symbol's description. Footprints have a separate footprint description property (not a field), which is specifically intended for a description of the footprint.

Fields each have an associated layer, which determines which board layer the field will be placed on. Fields can also be visible or hidden.

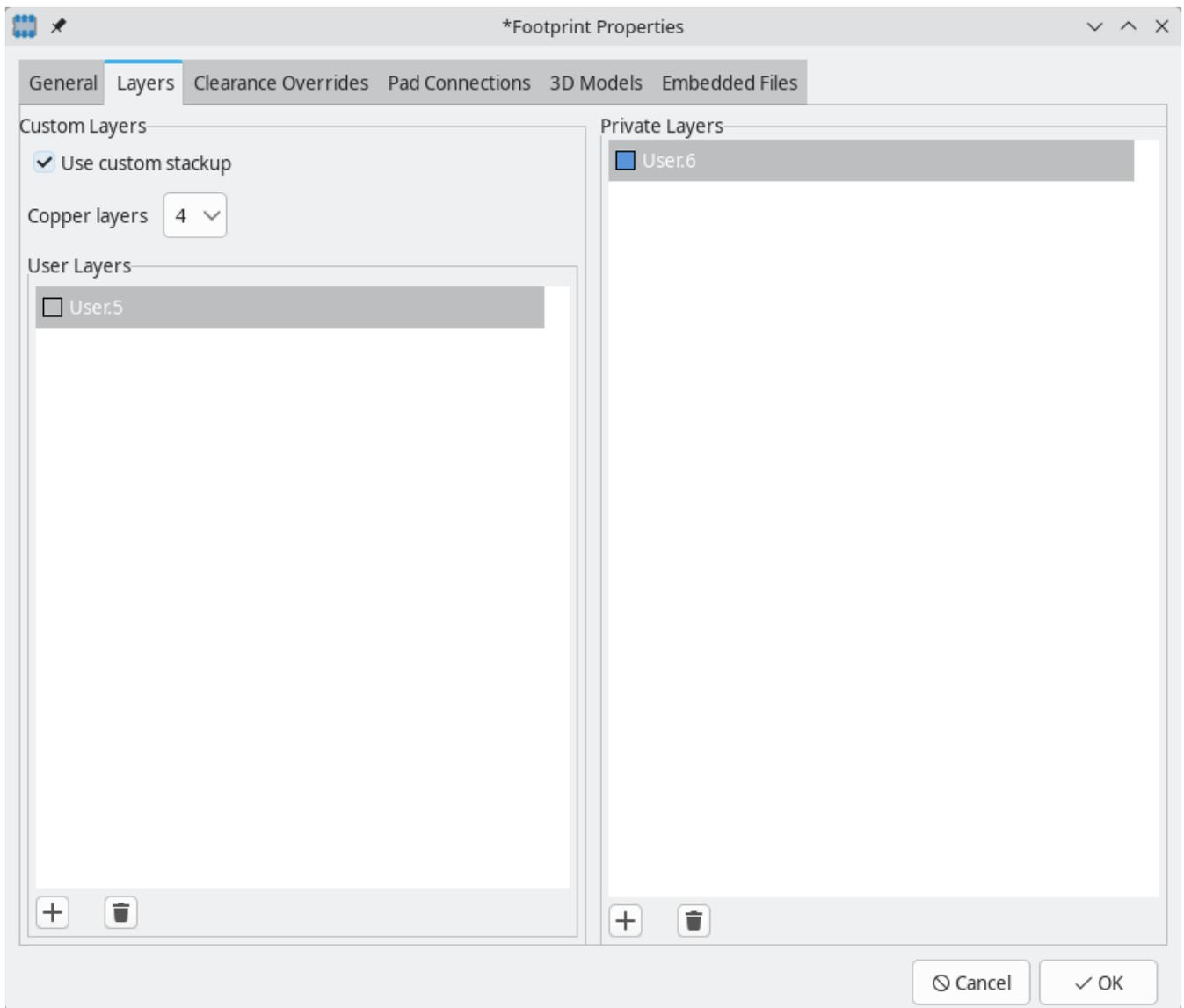
To edit an existing footprint field, double-click the field, select it or hover and press `E`, or right-click on the field text and select **Properties...**

To add new fields, delete optional fields, or edit existing fields, use the  icon on the main tool bar to open the Footprint Properties dialog. Fields can be arbitrarily named, but names starting with `ki_`, e.g. `ki_description`, are reserved by KiCad and should not be used for user fields.

Fields have a number of properties, each of which is shown as a column in the properties grid. Not all columns are shown by default; columns can be shown or hidden by right clicking on the grid header and selecting or deselecting columns from the menu.

## Footprint layers

By default, footprints have a front copper layer ( `F.Cu` ), a back copper layer ( `B.Cu` ), and a third copper layer that represents all inner copper layers on any board the footprint is added to ( `Inner layers` ). However, you can customize this stackup by enabling the **Use custom stackup** option in the Layers tab of a footprint's properties. When this setting is enabled, you must specify the exact number of copper layers contained in the footprint using the **Copper layers** dropdown. You can then customize any of these copper layers in the Footprint Editor, but if the footprint is added to a board with fewer copper layers, any items on the extra copper layers will be ignored.



You can also add additional user (non-copper) layers using the **User Layers** table. Any layers configured in this table will be added to the footprint, and you can edit these layers like any other footprint layer. When the footprint is added to a board, these layers will be visible in the board editor if the board is configured to include those layers.

**NOTE**

You can globally configure the number of user layers in footprints, as well as their names, in the **User Layer Names** section of the Footprint Editor's preferences. These user layers are shown in all footprints in addition to any layers configured in an individual footprint.

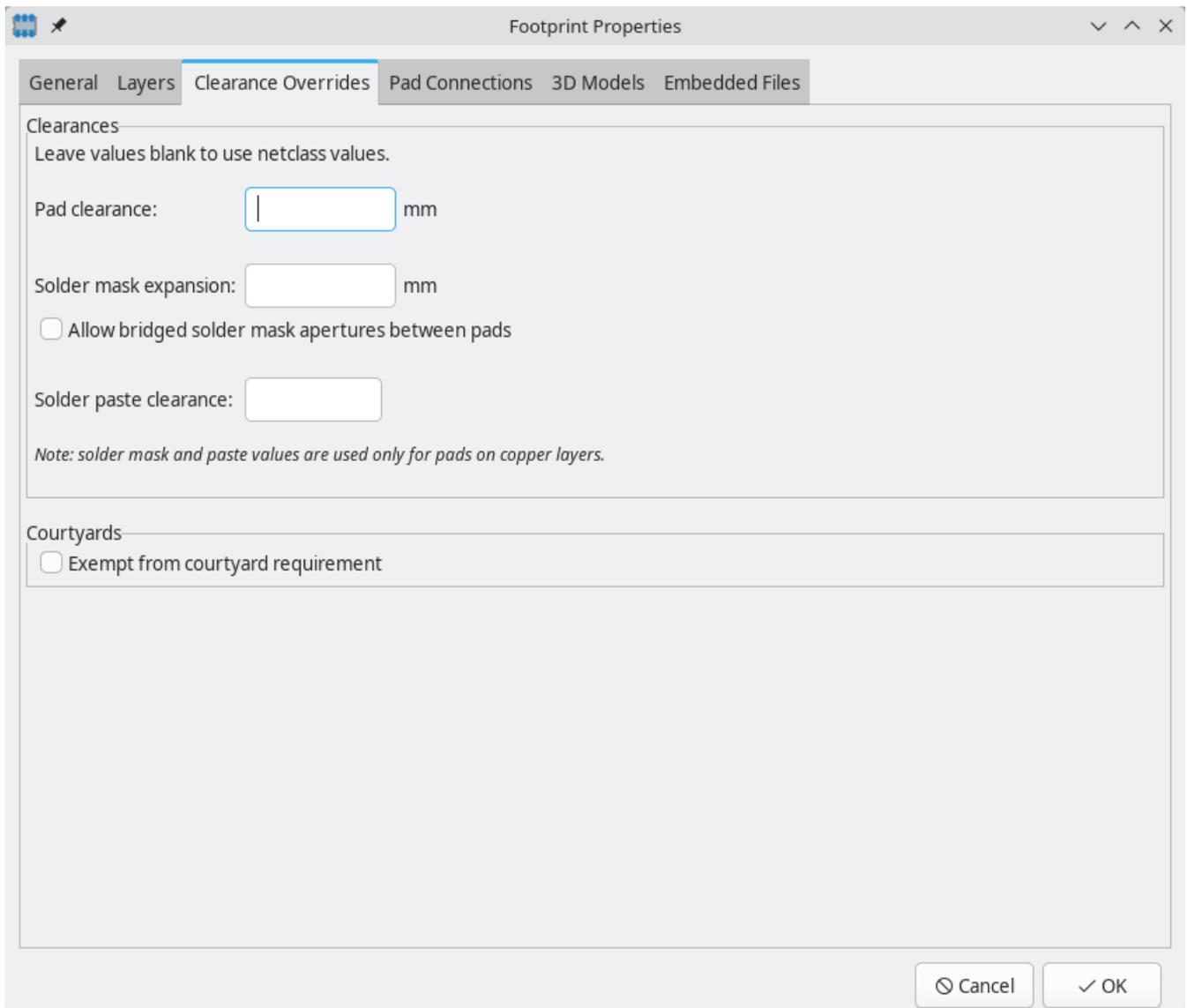
## Private footprint layers

Footprints can also have private footprint layers, which are layers that can be viewed and edited in the Footprint Editor but are never shown in the footprint when it is added to a board. Therefore any objects that are on private layers will not be visible in the PCB Editor or included in PCB fabrication outputs. This may be useful, for example, for notes or graphics that are of interest when drawing or editing a footprint but not needed at the board level.

Any of the existing `User.*` layers (`User.Drawings`, `User.Comments`, `User.Eco1`, `User.1`, etc.) can optionally be a private layer. To make a layer private, add a private layer in the **Layers** tab of the footprint properties dialog, then select the desired layer. Any objects on that layer will not be shown on the board.

## Clearance overrides

The **Clearance Overrides** tab of Footprint properties holds settings for footprint-specific overrides to board clearance and mask/paste expansion.



**Pad clearance** controls the minimum clearance between the footprint's pads and any copper shape (tracks, vias, pads, zones) on a different net. This value is normally empty, which will cause the pad clearance to be inherited from the board's design rules and netclass rules. This value can be overridden for individual pads by setting the pad's clearance to a nonblank value.

### NOTE

Prior to KiCad version 9, a pad clearance of 0 caused the pad clearance value to be inherited. In KiCad 9 and later, a pad clearance of 0 sets the clearance to 0, while a blank pad clearance causes the clearance to be inherited.

The aperture appearing on any technical layer will have the same shape and size as the pad shape on the copper layer(s). In the PCB manufacturing process, the manufacturer will often change the relative size of mask and paste apertures relative to the copper pad size, but since this size change is specific to a manufacturing process, most manufacturers expect the design data to be provided with the apertures set to the same size as the copper pads. For specific situations where you need to oversize or undersize a technical

layer aperture in the design data, you can use the settings here. These values can be overridden for individual pads by setting the pad's expansion or clearance to a nonzero value.

**Solder mask expansion** controls the size difference between the pad shape and the aperture shape on the F.Mask and B.Mask layers. A positive number means the solder mask aperture will be larger than the copper shape. This number is an inflation applied to all directions. For example, a value of 0.1mm here will cause the solder mask aperture to be inflated by 0.1mm, meaning that there will be an 0.1mm border on all sides of the pad and the solder mask opening will be 0.2mm wider than the pad when measured along a given axis.

**Solder paste clearance** is a setting to specify the solder paste shape relative to the parent pad size (the size difference between the pad shape and the aperture shape on the F.Paste and B.Paste layers). This can be specified as an absolute offset from the pad edge (e.g. -0.1mm), a value relative to the pad dimension (e.g. -5%), or both (e.g. -0.1mm - 5%). If it is 0 or blank, the solder paste aperture will be the same size as the pad. Positive values mean solder paste aperture larger than the pad. Negative values mean solder paste aperture smaller than the pad.

If **exempt from courtyard requirement** is checked, the footprint will not trigger a DRC violation if it does not contain a courtyard. Without this attribute set, a footprint without graphics on the F.Courtyard or B.Courtyard layer will cause a "Footprint has no courtyard defined" DRC violation.

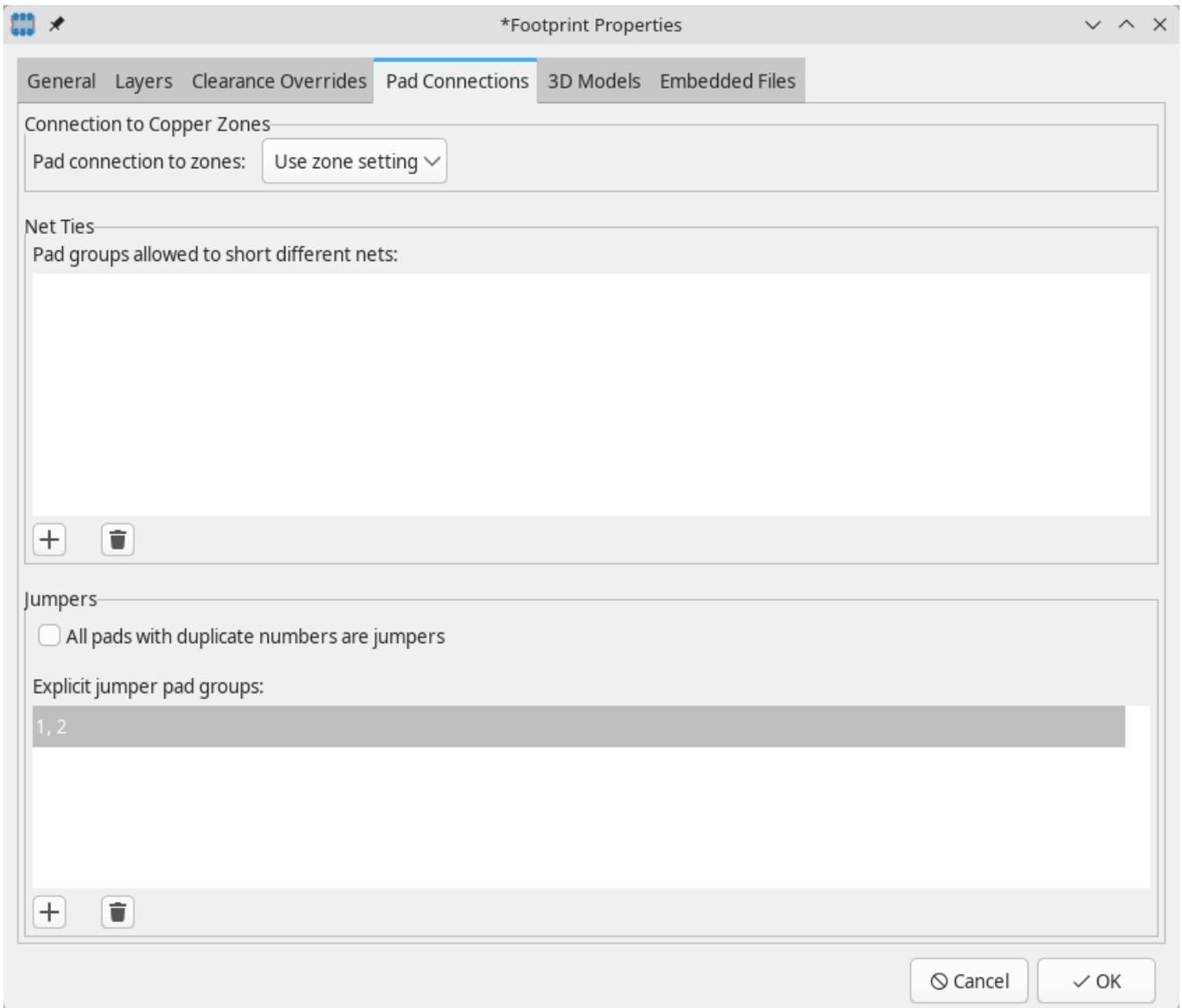
## Clearance override hierarchy

Clearance, mask expansion, and paste margin values are resolved through a hierarchy of overrides. When a value is set at multiple levels, the most specific level takes precedence:

1. **Pad-level override:** A value set directly on an individual pad. This is the highest-priority override and applies only to that specific pad.
2. **Footprint-level override:** A value set in the footprint's **Clearance Overrides** tab. This applies to all pads in the footprint that do not have their own pad-level override.
3. **Board-level default:** The global values configured in **Board Setup** → **Solder Mask/Paste**. These apply to all pads that do not have a footprint-level or pad-level override.

## Pad connections, net ties, and jumper pads

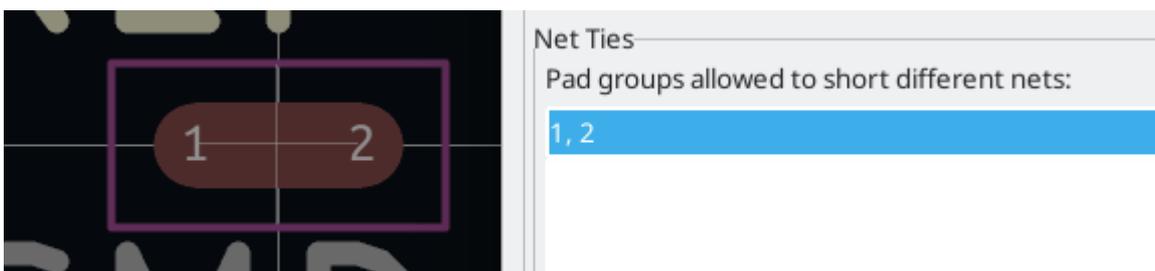
The Pad Connections tab of Footprint Properties holds settings for how pads in the footprint connect to zones, whether the footprint can short two different nets (a [net tie](#)), and whether pads in the footprint are internally connected in the attached component ([jumper pads](#)).



**Pad connection to zones** controls whether the footprint's pads will have solid, thermal relief, or no connection to zones. The default setting for this control is **From zone setting**, which uses the connection mode specified in the connection zones' properties. The control in this tab sets the override for an entire footprint, but you can also override the setting for individual pads in each pad's properties by setting the pad's connection mode to a value other than **From parent footprint**.

## Net ties

Footprints can act as net ties, where two separate nets are electrically connected by copper. Connecting nets together would normally causes a DRC error due to violating the clearance between two nets, but a footprint can be configured to short nets without causing a DRC violation. This can be used to connect multiple grounds at a specific location, to make kelvin sense connections to a component, or for other applications.



Net ties connect two or more nets in one contiguous region of copper. Each net in a net tie must have its own pad. Pads are not ordinarily allowed to short to other pads; to allow pads to be shorted in net ties, the shorting pads must be added to a **net tie group**. To create a net tie group, add the pad numbers of the shorting pads to the **Net Ties** table in the **Pad Connections** tab of the Footprint Properties dialog. For example, to allow pads 1 and 2 to short in a footprint, add a line to the table with the contents 1,2 or 1 2.

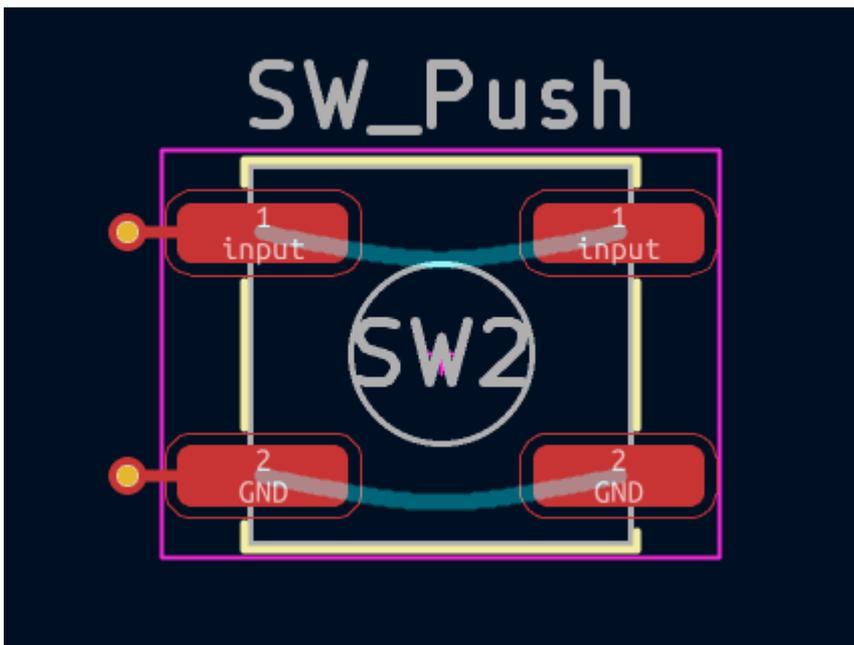
After creating a net tie group, the specified pads are allowed to be electrically shorted. Pads in net tie groups can be connected either by directly overlapping the pads or by adding a copper shape that overlaps both pads.

Footprints can contain multiple net tie groups. Each group can short two or more nets, but every group must remain electrically separate from other groups.

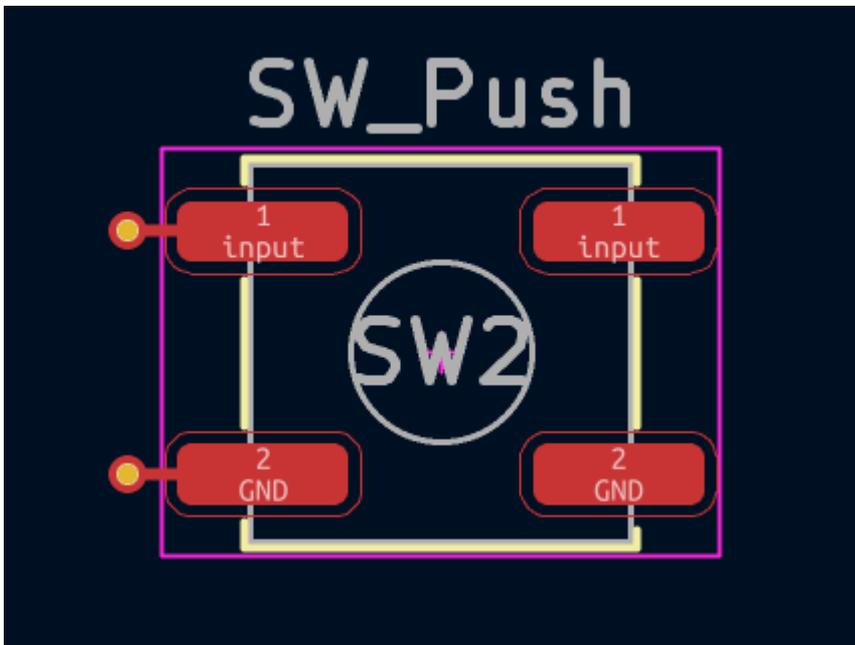
## Jumper pads

You can configure a footprint to *jumper* some or all of its pads internally. Any jumpered pads will always have the same net; they are considered shorted together, even if they are not explicitly connected together in the schematic and PCB.

Footprint with jumper pads represent components that internally short together multiple pads. Examples of such components are wire jumpers, connectors with multiple connected shield pads, and switches with multiple shorted pads on each side of the switch.



*A footprint without jumpered pads. The ratsnest shows missing connections between same-numbered pads.*



The same footprint in the same circuit with same-number pads jumpered (internally connected by the component). PCB connections between the same-numbered pads are no longer required.

Because KiCad considers the jumpered pads to be shorted by the component, you only need to connect to one of the shorted pads in the schematic and PCB.

#### NOTE

Jumper pads are different than net ties. A [net tie](#) allows copper to bridge two different nets, and each pad in the net tie has a different net. With jumper pads, each pad has the same net, and the electrical connection between jumper pads is assumed to be off-board in the attached component.

You can configure jumper pads in the **Pad Connections** tab of a footprint's Properties.

When the **All pads with duplicate numbers are jumpers** setting is enabled, all pads (and pins in the linked symbol) will be jumpered to the other pads with the same number. For example, all pads numbered 1 will be considered connected to each other.

You can also configure jumper pad groups for pads that don't have the same number by adding **Explicit pad jumper groups** with the **+** button. Enter multiple pad numbers separated by spaces or commas. A pad jumper group with the text `1 2` or `1, 2` will jumper together pads 1 and 2. A group with `3 4 5` or `3, 4, 5` will jumper together pads 3, 4, and 5.

#### NOTE

Like many other footprint properties, a footprint's jumper pad settings are transferred from the linked symbol. You can configure a symbol's jumper pins using the [Symbol Editor](#).

## 3D models

The **3D Models** tab of Footprint Properties allows you to attach external 3D model files to a footprint and view the footprint in three dimensions along with any attached models.

 footprint editor properties 3dmodels

The main part of the window is a 3D preview of the footprint and any attached models. The buttons to the right of the preview let you enable or disable an orthographic projection () , show or hide the PCB model (

, align the view to one of the six face-aligned perspectives () , and refresh the view (). The bottom button () lets you set the thickness of the PCB in the preview.

The top of the dialog lets you attach external models. Each added model will be shown in the footprint preview as well as in the full PCB 3D view when the footprint is added to a board. Footprint models can be in STEP, VRML, or **IDF** format. The models are specified as paths to the model files, which can contain [path variables](#) such as `${KIPRJMOD}` or `${KICAD9_3DMODEL_DIR}`. Click the **Configure Paths** button to configure path variables. If there is a problem loading a model file from the specified path, an icon in the leftmost column will indicate an error.

**NOTE**

KiCad will automatically resolve versioned path variables from older versions of KiCad to the value of the corresponding variable from the current KiCad version, as long as the old variable is not explicitly defined itself. For example, `${KICAD8_FOOTPRINT_DIR}` will automatically resolve to the value of `${KICAD9_FOOTPRINT_DIR}` if there is no `KICAD8_FOOTPRINT_DIR` variable defined.

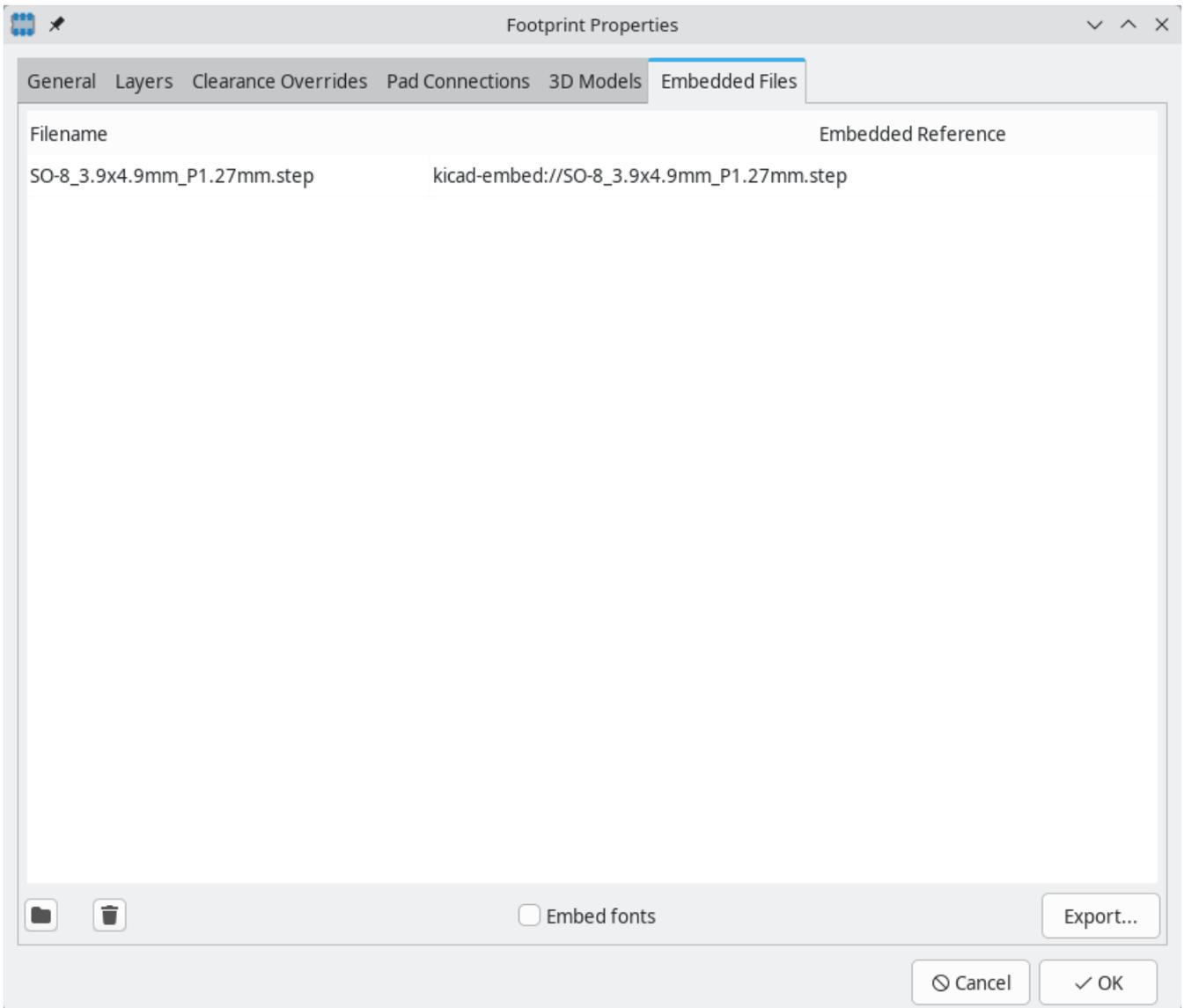
**NOTE**

Many footprints in KiCad's standard library do not yet have model files created for them. However, these footprints may contain a path to a 3D model that does not yet exist, in anticipation of the 3D model being created in the future.

By default, models are added with their origin placed at the footprint's origin, with no offset, scaling, or rotation. Offset, scaling, and rotation can be applied to a model using the controls to the left of the preview canvas. The model's opacity can be adjusted using the **opacity** slider, and the model can be completely hidden by deselecting the **show** checkbox in the rightmost column of the model table.

## Embedding files

External files can be embedded within a footprint. Embedding a file stores a copy of the file inside the footprint. The footprint can then refer to the embedded copy of the file instead of the external file, which makes the footprint more portable as it doesn't rely on an external file, although the footprint's filesize is increased as a result. In footprints this is especially useful for embedding 3D models. Files embedded in a footprint are deduplicated when the footprint is added to a board: if a file is embedded in a footprint, and multiple instances of that footprint are added to the board, only one copy of the file will be embedded, and all of the footprint instances will refer to the same embedded file. Files embedded in a footprint cannot be referred to in the parent board. File embedding is explained in more detail in the [Board Setup documentation](#).



#### NOTE

You can add a 3D model to a footprint and embed it in one step. To do so, add a 3D model in the **3D Models**, and enable the **Embed File** checkbox in the file browser while choosing a model. This embeds the file and automatically uses the embedded reference as the file path instead of the path to the external file.

## Rule areas

Rule areas, also known as keepouts, are footprint regions that can have specific DRC rules defined for them. Some basic rules are available that will raise DRC errors if certain types of objects are within the bounds of the rule area, but rule areas can also be used together with [custom DRC rules](#) to define complex DRC behavior that only applies within the rule area. A rule area in a footprint takes effect when the footprint is added to the board.

You can add a rule area by clicking the  button on the right toolbar (**Ctrl** + **Shift** + **K**). Click on the canvas to place the first corner, which will show the Rule Area Properties dialog. After configuring the rule area appropriately, press **OK** to continue placing corners of the rule area. The rule area shape can be an arbitrary polygon; click on the starting corner or double click to finish placing the rule area.

Rule areas are described in more detail in the [PCB editor](#) documentation.

## Reference images

Just like in the PCB Editor, you can use reference images in the Footprint Editor to assist with your footprint designs. Footprint reference images are only shown in the Footprint Editor: they are not shown in the PCB Editor when a footprint is added to a board, and they do not appear in any fabrication outputs.

To add a reference image, use the  button on the right toolbar and select the desired reference image file.

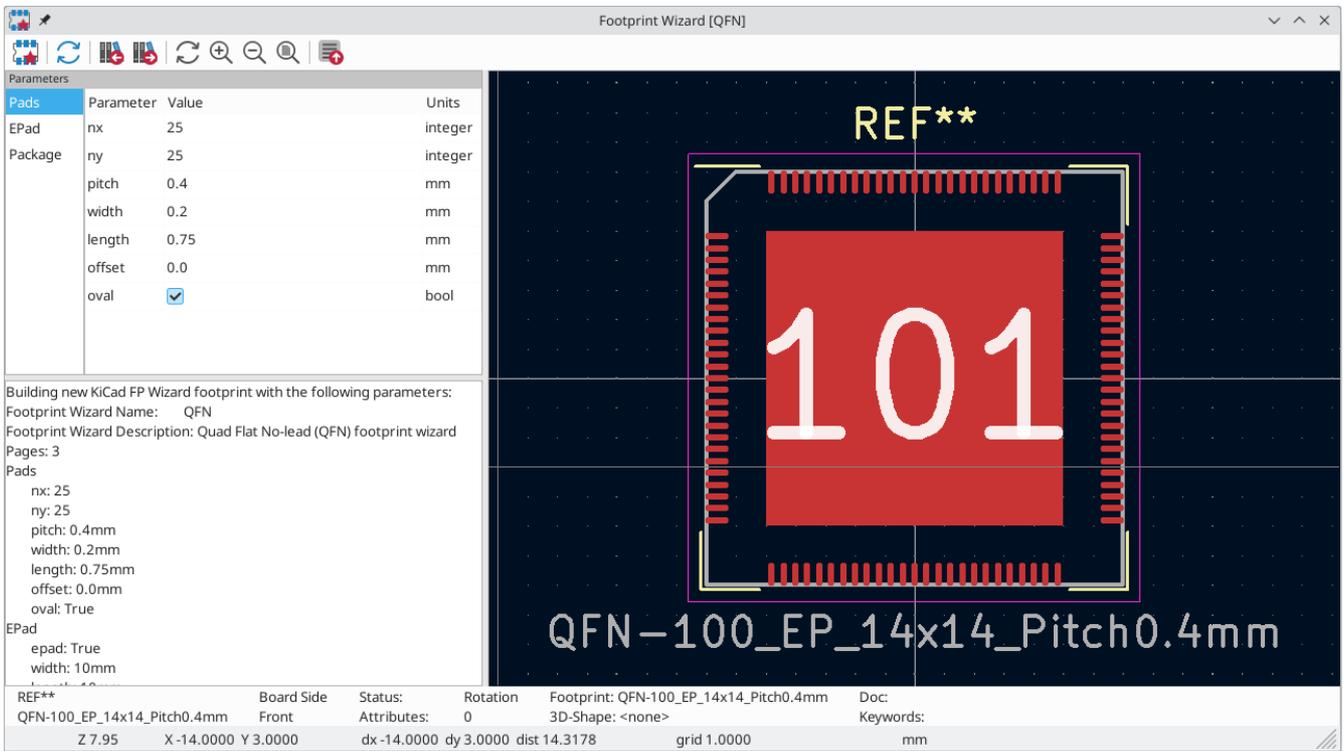
Reference images are described in more detail in the [PCB Editor documentation](#).

## Footprint wizards

KiCad provides a set of footprint wizards that can be used to create some common kinds of footprints based on a set of parameters. Wizards for the following types of footprints are provided:

- BGA packages
- QFN packages
- QFP packages
- SOIC, MSOP, SSOP, TSSOP, etc. packages
- SIP and DIP packages
- ZIP packages
- ZOIC packages
- FPC connectors
- Micromatch SMD connectors
- Circular pad arrays
- Touch sliders
- Mutual capacitance touch buttons
- USS-39 barcodes
- QR codes

To create a footprint using a footprint wizard, click the  button and choose a footprint type from the list that appears.

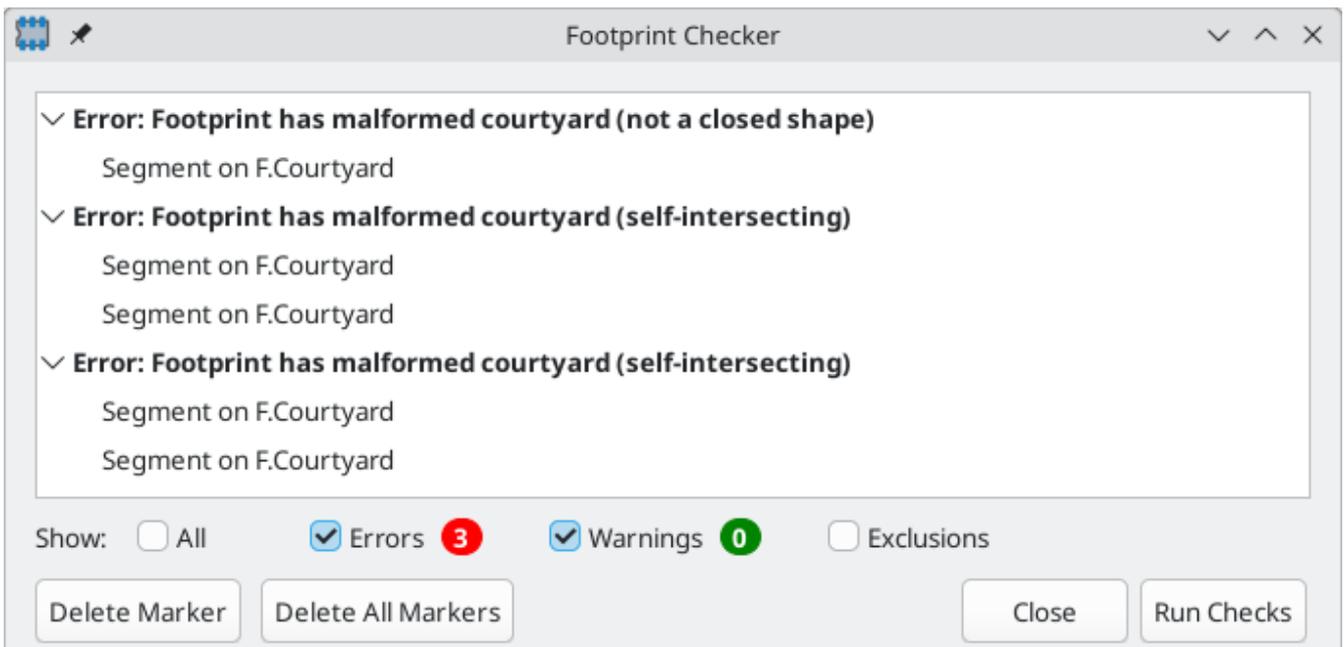


In the window that appears, fill out the parameters as appropriate. When the parameters are correctly filled out, press the  button to transfer the generated footprint back into the footprint editor. Then you can make additional manual modifications and save the footprint as normal.

In addition to the set of footprint wizards that KiCad provides, you can also create your own. For more information about creating new footprint wizards, see the [Scripting section](#) of the Advanced Topics chapter.

## Checking footprints

The Footprint Editor can check for common issues in your footprints. Run the footprint checker using the  button in the top toolbar.



Any footprint issues that are detected are listed in the dialog and displayed with arrow indicators in the editing canvas. Clicking on an issue in the dialog will focus on the issue in the canvas.

The footprint checker checks for:

- Malformed or missing courtyards
- Pads that don't match the footprint's type: footprints without any through hole pads should be set to the surface mount footprint type
- Through hole pads without a hole
- Plated through hole pads not on any copper layers
- Plated through hole pads without a copper annulus
- Plated through hole pads with a hole that isn't fully within the pad
- Surface mount pads on both the front and back
- Surface mount pads with mismatched copper and paste/mask layers (front copper and back paste/mask, or vice versa)
- Pads co-located with or too close to other holes
- Collisions between silkscreen and solder mask openings
- Pads that short to other pads outside of net tie groups
- Nonexistent pads in net tie groups
- Pads in that appear in multiple net tie groups

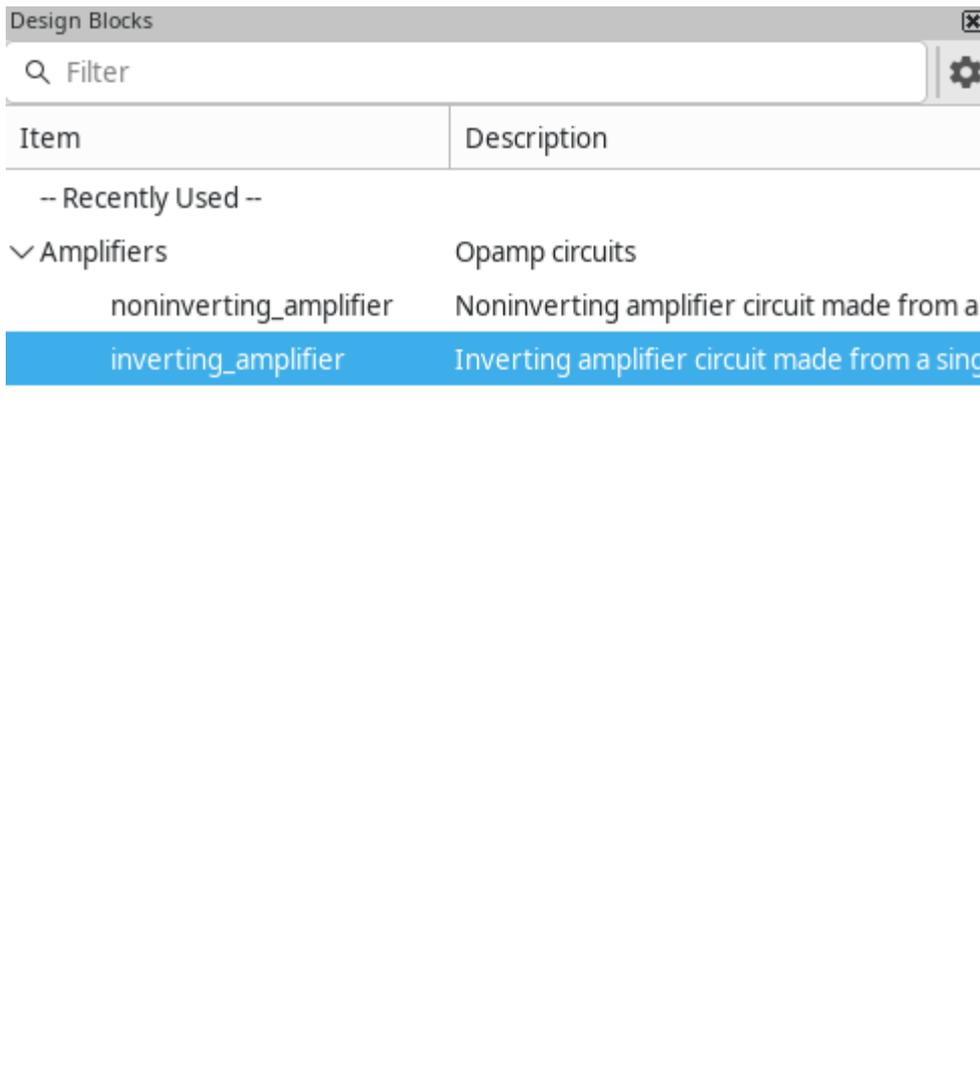
# Design blocks

Design blocks allow you to save a portion of a schematic and/or layout and reuse it later. You can reuse design blocks within the same project or between different projects. Design blocks are saved and organized in design block libraries, much like symbols and footprints. A single design block can contain a schematic fragment, a layout fragment, or both.

When you instantiate a schematic design block in a schematic, the saved schematic fragment is inserted into the current schematic, either in the current sheet or in a new subsheet. Instantiating a layout design block directly adds saved board fragment to the board. If a design block contains both schematic and layout fragments, you can apply the saved layout to the footprints that correspond to the symbols in the schematic fragment.

Typically, design blocks are created first as a schematic fragment, with layout added later. Reusing schematic fragments in design blocks is covered in the [Schematic Editor manual](#).

To use layout design blocks, first show the Design Blocks panel by clicking **View** → **Panels** → **Design Blocks**. This opens a docked panel on the right side of the schematic editor. To close the panel, use the same menu entry again, click the close button in the top right of the panel, or right click in the panel and choose **Hide Library Tree**.



**inverting amplifier**

The Design Blocks panel lets you create design block libraries and create, edit, and place design blocks in the PCB.

## Creating and managing design blocks

Design blocks are saved in design block libraries, so you need to add a library before you can save any design blocks.

### Creating a new design block library

To create a new library, right click in the library tree and select **New Library...** This opens a file browser for you to choose a location and name for the new library. Design block libraries are saved as a folder; the folder name is the library name plus the suffix `.kicad_blocks`.

The file browser also lets you choose whether the new library should be added to the global design block library table or the project design block library table. Libraries in the global library table will be available to all projects, while libraries in the project library table will only be available in the current project.

#### NOTE

The global and project design block library tables are managed using **Preferences** → **Manage Design Block Libraries...** This includes deleting and renaming design block libraries. The design block library tables behave in the same way as the symbol and footprint library tables. For more information about managing library tables, see the [footprint library table documentation](#).

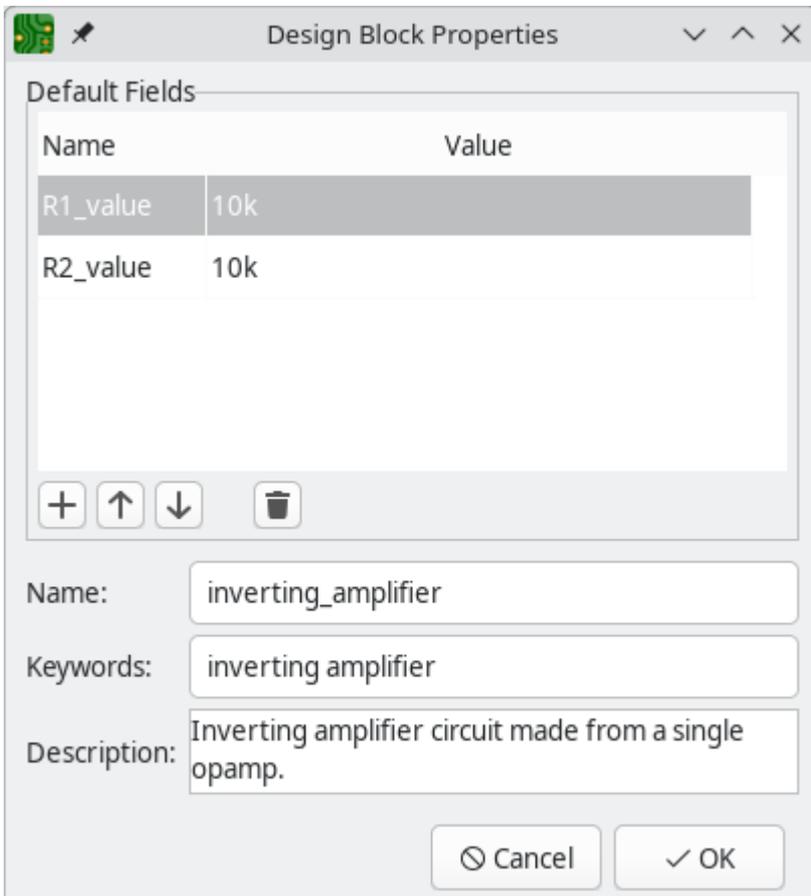
After choosing a name, location, and library table, a new, empty library is created. When you create new design blocks, you can save them in this library.

### Creating a new design block

In the PCB Editor, design blocks can be created either from the entire contents of a board or from a selection of board objects. To create and save a new design block, select the desired source objects, either by opening the desired sheet or selecting the objects in the editing canvas. Then right click the design block library that will contain the block and select **Save Board as Design Block...** or **Save Selection as Design Block...** as appropriate.

#### NOTE

To save a layout fragment to an existing design block, including a design block that contains a schematic but no layout fragment, don't create a new design block. Instead, [update the existing design block](#).



This brings up the Design Block Properties dialog, where you can edit the properties of the new design block.

- **Name:** this is the name of the new design block, which is shown in the library tree and the preview pane. It is also used when filtering design blocks with the filter textbox. When design blocks are added as a group or as a schematic sheet, this is the default name of the new group or sheet.
- **Keywords:** these are space-separated keywords describing the design block. They are displayed in the design block preview pane and used when filtering design blocks with the filter textbox.
- **Description:** this is a description of the design block, which is shown in the library tree and the preview pane. It is also used when filtering design blocks with the filter textbox.
- **Default Fields:** these are key/value pairs which are included as [hierarchical sheet fields](#) when the design block's schematic is placed as a sheet. Fields are ignored when the design block is not placed as a sheet.

You can edit a design block's properties after creating it by right clicking the design block in the design block library tree and selecting **Properties...**

## Updating an existing design block

To update the layout of an existing design block, first [place the design block](#) you want to modify, then edit the layout as desired. If the design block exists but does not contain a layout portion, place the design block in the schematic, [update the pcb from the schematic](#) to add the linked footprints to the board, then edit the layout as desired.

When the layout changes are complete, save the modified layout to the original design block: select the board elements you want to save as the design block, then right click the name of the design block you want

to modify and click **Update Design Block from Selection**. The existing contents of the design block are replaced by the selection.

Depending on how the design block was placed in the schematic, there are several other possible ways to save the modified schematic into the design block. If the current board only contains what you want to save into the design block, you can right click the name of the design block in the Design Block panel and click **Update Design Block from Board**. The existing contents of the design block are replaced by everything in the board. If the design block was placed as a group, you can right click the group in the canvas and choose **Save to Linked Design Block**.

## Using design blocks in a board

There are two ways to use layout design blocks in a board, which are useful in different situations.

- Design blocks can be directly added to a board, in the same way that schematic design blocks are directly added to a schematic. When a design block is added this way, its contents are not associated with any symbols from the schematic. This can be useful to add footprints or other board items in a predetermined arrangement when those items do not correspond to anything in the schematic. For example, mounting holes in a particular pattern could be added this way.
- If a design block was used in the schematic, you can go through the normal [Update PCB from Schematic](#) flow to add the schematic design block's symbols' assigned footprints to the board, then apply the design block's saved layout to those footprints. This lets you reuse both a schematic fragment and its linked layout.

Both ways are explained in more detail below.

## Directly placing design blocks in a board

To directly place a layout design block in a board, without any schematic symbols that correspond to the contents of the layout design block, you can place the layout design block using the Design Blocks panel.

The Design Blocks panel contains a library tree that lists your design block libraries and the design blocks contained in each library. Each library can be expanded or collapsed to show or hide the design blocks in that library. There is a **Recently Used** pseudo-library at the top of the tree that contains any design blocks that you have recently placed. You can pin any libraries to the top of the list by right clicking the library and selecting **Pin Library**.

You can filter design blocks by their name, description, and keywords using the filter textbox at the top of the Design Blocks panel. By default, matches are sorted by best match, but you can change to sorting alphabetically by selecting **Sort Alphabetically** under the  menu.

When you select a design block in the library tree, the design block's name and metadata are displayed below the library tree along with a graphical preview of the design block. The metadata includes the block's description and keywords.

To add a design block to the schematic, double click it in the library tree or right click a design block and select **Place Design Block**.

There are several options controlling how the design block is placed:

-

If the **Place repeated copies** checkbox is enabled, KiCad will begin placing the design block again when you finish placing the previous block. To cancel placing the next block, press  or right click and select **Cancel**.

- If the **Place as group** checkbox is enabled, the contents of the design block will be added to a [group](#) in the board, and the group's library link will be set to point to the source design block and library. It is not required to place design blocks as a group, but some design block functionality is not available unless the design block is grouped. See the [Grouped design blocks](#) section for more information.
- If the **Keep annotations** checkbox is enabled, KiCad will insert the design block without changing the footprint annotations as defined in the saved design block. If it is not enabled, KiCad will clear the footprint annotations while inserting the design block.

Once placed in a schematic, the contents of a design block behave the same as any other schematic objects and can be edited, moved, deleted, etc. exactly as if they were added to the schematic normally.

## Applying layouts to design blocks from a schematic

If you want to reuse a layout on some footprints that are linked to symbols in your schematic, you can use design blocks to achieve this. You will need a design block that contains both schematic and layout fragments. After you add the design block in the schematic, you can apply the saved layout to the corresponding footprints.

### NOTE

Also consider the [multichannel tool](#) as an alternative workflow for reusing layout within a project.

The process for applying a design block's saved layout is as follows:

1. [Create a design block](#) that contains both schematic and layout components.
2. [Add the design block to the schematic](#). Make sure to place the schematic design block as a group.
3. [Update the PCB from the schematic](#). Make sure to enable the **Group footprints based on symbol group** option. This adds footprints to the board corresponding to the symbols in the schematic design block and groups them together. The group is set up with a [library link](#) that points to the correct design block.
4. In the PCB Editor, select the group of footprints that corresponds to the schematic design block. Right click it and choose **Apply Design Block Layout**. This rearranges the footprints according to the design block layout, draws tracks between pads as appropriate, and adds any other contents from the design block layout such as graphics or text. The new items are also added to the group.

## Grouped design blocks

When you place a design block in a board, you have the option to place the block as a group. You also have the option to group the design block's board contents when transferring a design block from the schematic, as long as the design block was also grouped in the schematic. In addition to keeping the design block together as a single unit, some design block features can only be used if the design block's contents are grouped. In particular, design blocks must be grouped in both the schematic and PCB if you plan to apply a layout to the design block in the PCB.

The following design block features can be used when a design block is grouped:

-

Place additional instances of a design block by right clicking the grouped design block and clicking **Place Linked Design Block**.

- Update the library version of a design block to match a design block in the PCB by right clicking the grouped design block and clicking **Save to Linked Design Block**.
- Copy the schematic group to the PCB, grouping the footprints corresponding to the design block's symbols, by [updating the PCB from the schematic](#).
- Apply a design block layout to the footprints in a design block by right clicking the grouped footprints in the PCB and clicking **Apply Design Block Layout**.
- As an alternative workflow to applying the layout saved in a design block, use the [multichannel tool](#) to repeat a layout from one group to another. With this workflow, you can place a design block as a group in the schematic, transfer the group to the PCB, manually perform the layout for one design block's group, and then use the multichannel tool to replicate the layout of the first group to the other groups.

When starting from a schematic design block, the easiest way to group a design block is to enable the **Place as group** checkbox when you place the design block in the schematic. Then transfer the group to the PCB while [updating the PCB from the schematic](#) by enabling the **Group footprints based on symbol groups** option.

When directly placing a layout design block, you can enable the **Place as group** checkbox when you place the design block in the PCB.

If you didn't place a design block as a group, but want to group it later, that's also possible. First, select all of the contents of the design block, then [group](#) them. It is important that the group includes exactly the contents of the design block, with no missing or extra items. In the group's properties, set the **Library link** so that it matches the design block's name and library, in the format `<library>:<block>`. For example, the library link for the `inverting_amplifier` design block in the `amplifiers` design block library would be `amplifiers:inverting_amplifier`.

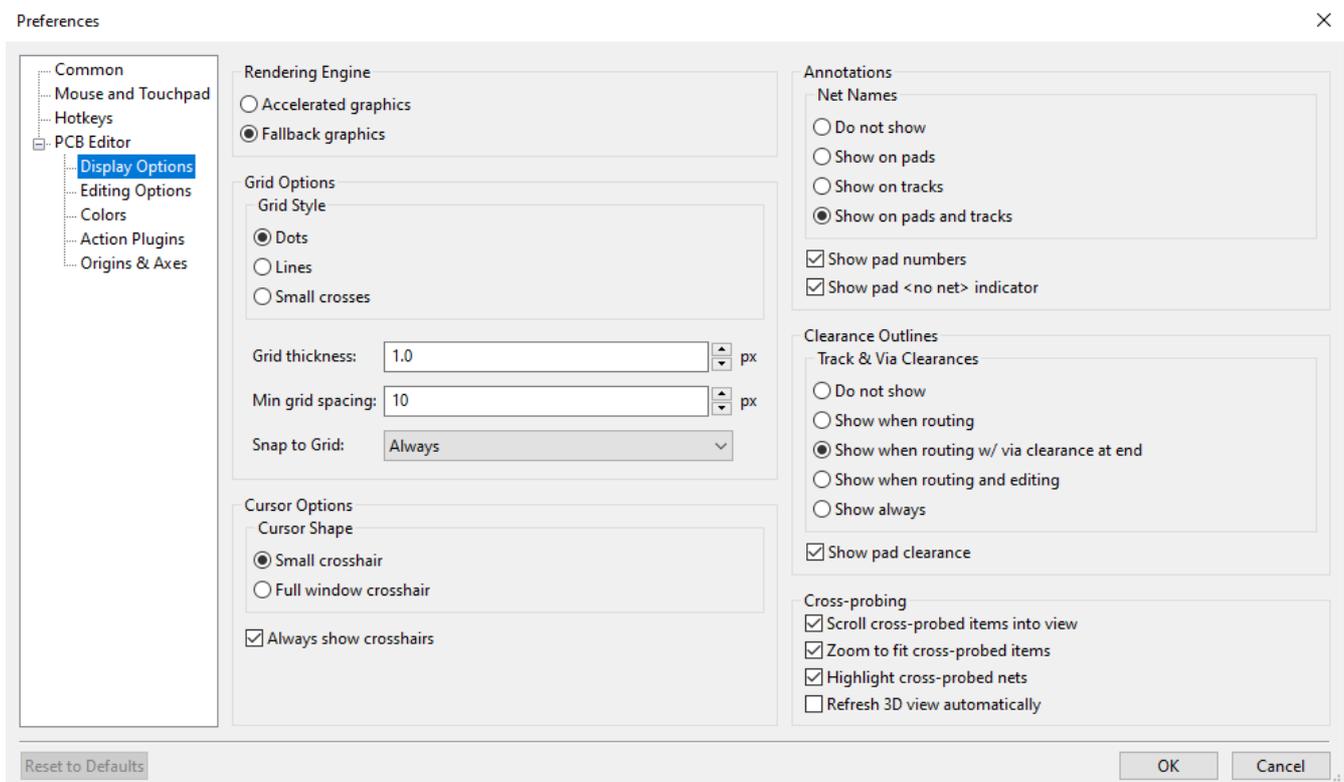
# Advanced topics

## Configuration and Customization

The KiCad PCB Editor has a variety of preferences that can be configured through the Preferences dialog. Like all parts of KiCad, the preferences for the PCB Editor are stored in the user configuration directory and are independent between KiCad minor versions to allow multiple versions to run side-by-side with independent preferences.

The first sections of the Preferences dialog (Common, Mouse and Touchpad, and Hotkeys) are shared between all KiCad programs. These sections are described in detail in the KiCad manual under the "Common preferences" section.

## Display options



**Rendering Engine:** Controls if Accelerated graphics or Fallback graphics are used.

**Grid style:** Controls how the alignment grid is drawn.

**Grid thickness:** Controls how thick grid lines or dots are drawn.

**Min grid spacing:** Controls the minimum distance, in pixels, between two grid lines. Grid lines that violate this minimum spacing will not be drawn, regardless of the current grid setting.

**Snap to grid:** Controls when drawing and editing operations will be snapped to coordinates on the active grid. "Always" will enable snapping even when the grid is hidden; "When grid shown" will enable snapping only when the grid is visible.

### NOTE

Grid snapping can be temporarily disabled by holding down Ctrl.

**Cursor shape:** Controls whether the editing cursor is drawn as a small crosshair or a full-screen crosshair (a set of lines covering the entire drawing canvas). The editing cursor shows where the next drawing or editing action will occur and will be snapped to a grid location if snapping is enabled.

**Always show crosshairs:** Controls whether the editing cursor is shown all the time or only when an editing or drawing tool is active.

**Net names:** Controls whether or not net name labels are drawn on copper objects. These labels are guides for editing only and do not appear in fabrication outputs.

**Show pad numbers:** Controls whether or not pad number labels are drawn on footprint pads.

**Show pad <no net> indicator:** Controls whether or not pads with no net are indicated with a special marker.

**Track clearance:** Controls whether or not clearance outlines around tracks and vias are shown. Clearance outlines are shown as thin shapes around objects that indicate the minimum clearance to other objects, as defined by constraints and design rules.

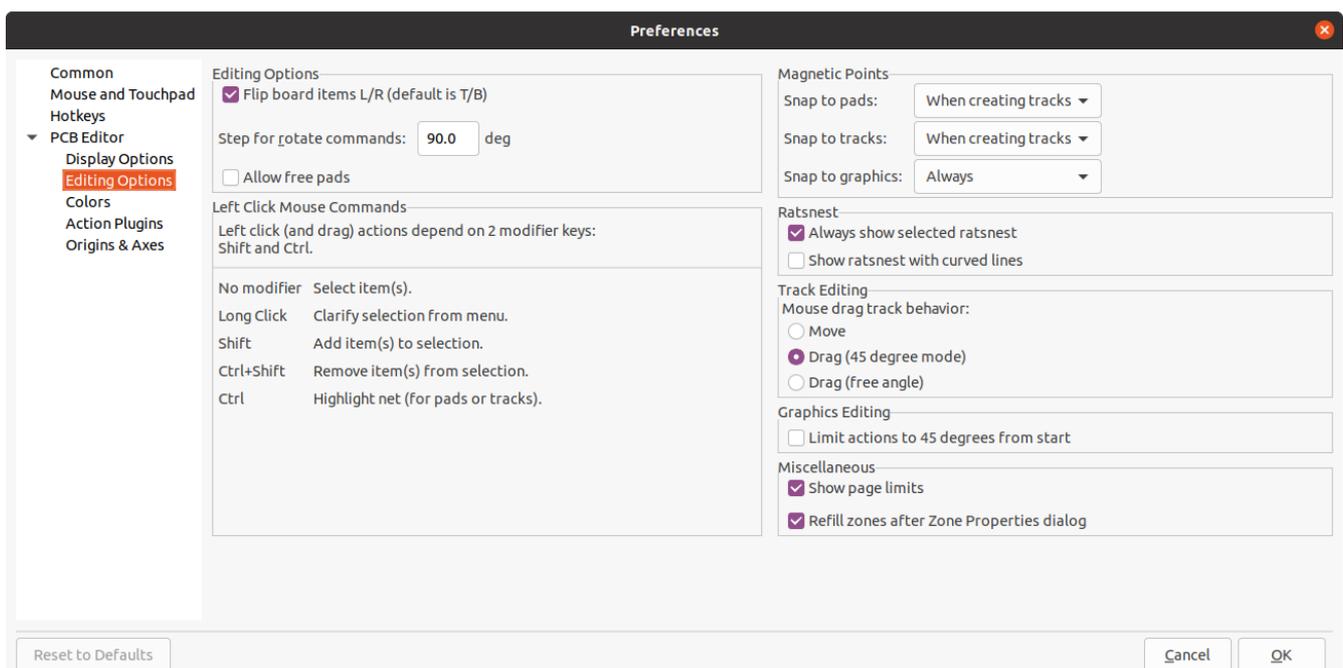
**Show pad clearance:** Controls whether or not clearance outlines around pads are shown.

**Center view on cross-probed items:** When the Schematic and PCB Editors are both running, controls whether clicking a component or pin in Eeschema will center the PCB Editor view on the corresponding footprint or pad.

**Zoom to fit cross-probed items:** Controls whether the view will be zoomed to show a cross-probed footprint or pad.

**Highlight cross-probed nets:** Controls whether or not nets highlighted in Eeschema will be highlighted in the PCB Editor when the highlight tool is activated in both tools.

## Editing options



**Flip board items L/R:** Controls the direction board items will be flipped when moving them between the top and bottom layers. When checked, items are flipped Left-to-Right (about the Vertical axis); when unchecked, items are flipped Top-to-Bottom (about the Horizontal axis).

**Step for rotate commands:** Controls how far the selected object(s) will be rotated each time the Rotate command is used.

**Allow free pads:** Controls whether or not the pads of footprints can be unlocked and edited or moved separately from the footprint.

**Magnetic points:** This section controls object snapping, also called magnetic points. Object snapping takes precedence over grid snapping when it is enabled. Object snapping only works to objects on the active layer. Hold  to temporarily disable object snapping.

**Snap to pads:** Controls when the editing cursor will snap to pad origins.

**Snap to tracks:** Controls when the editing cursor will snap to track segment endpoints.

**Snap to graphics:** Controls when the editing cursor will snap to graphic shape points.

**Always show selected ratsnest:** When enabled, the ratsnest for a selected footprint will always be shown even if the global ratsnest is hidden.

**Show ratsnest with curved lines:** Controls whether ratsnest lines are drawn straight or curved.

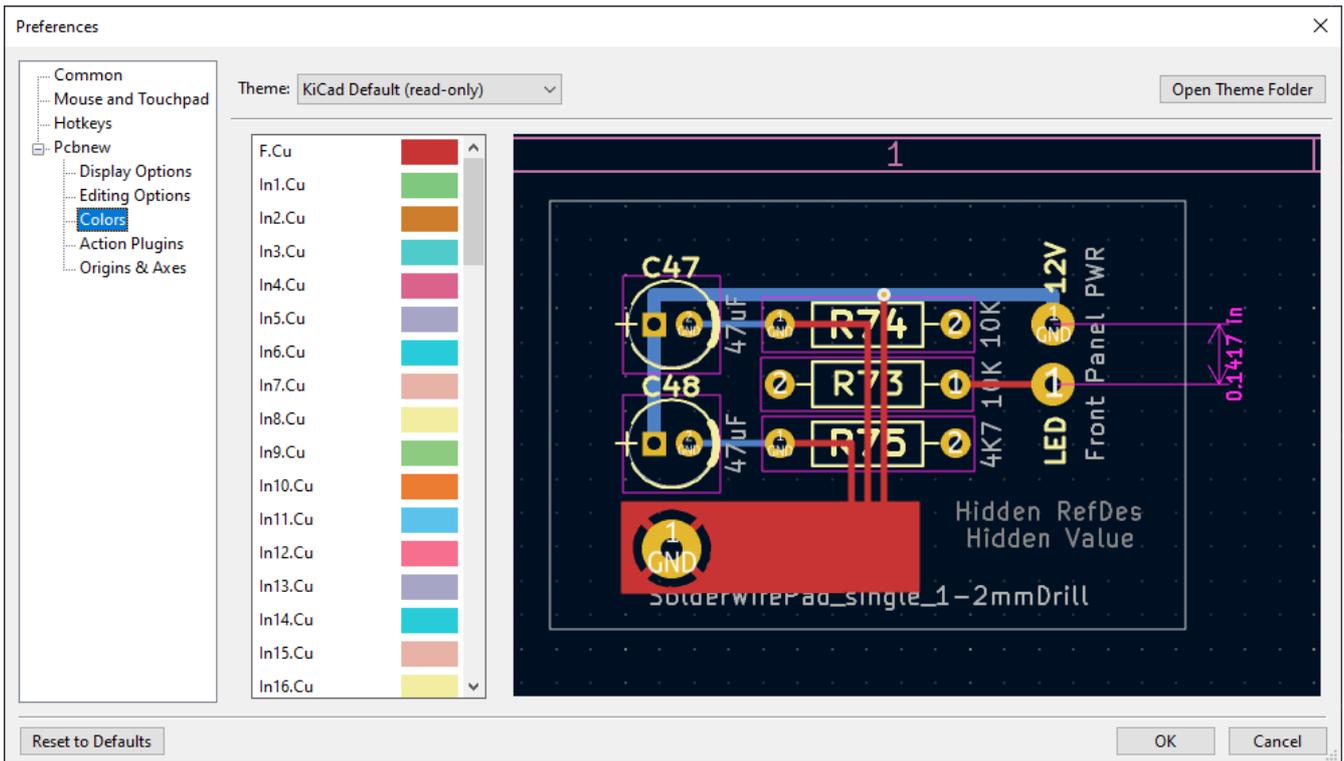
**Mouse drag track behavior:** Controls the action that will occur when you drag a track segment with the mouse: "Move" will move the track segment independent of any others. "Drag (45 degree mode)" will invoke the push-and-shove router to drag the track, respecting design rules and keeping other track segments attached. "Drag (free angle)" will move the nearest corner of the track segment, highlighting collisions with other objects but not moving them out of the way.

**Limit actions to 45 degrees from start** Controls whether lines drawn with the graphic drawing tools can take on any angle. Note that this only affects drawing new lines: lines can be edited to take on any angle.

**Show page limits:** Controls whether or not the page boundary is drawn as a rectangle.

**Refill zones after Zone Properties dialog:** Controls whether or not zones are automatically refilled after editing the properties of any zone. This may be disabled on complicated designs or slower computers to improve responsiveness.

# Colors



KiCad supports switching between different color themes to match your preferences. KiCad 10.0 comes with two built-in color themes: "KiCad Default" is a new theme designed to have good contrast and balance for most cases and is the default for new installations. "KiCad Classic" is the default theme from KiCad 5.1 and earlier versions. Neither of these built-in themes can be modified, but you can create new themes to customize the look of KiCad as well as install themes made by other users.

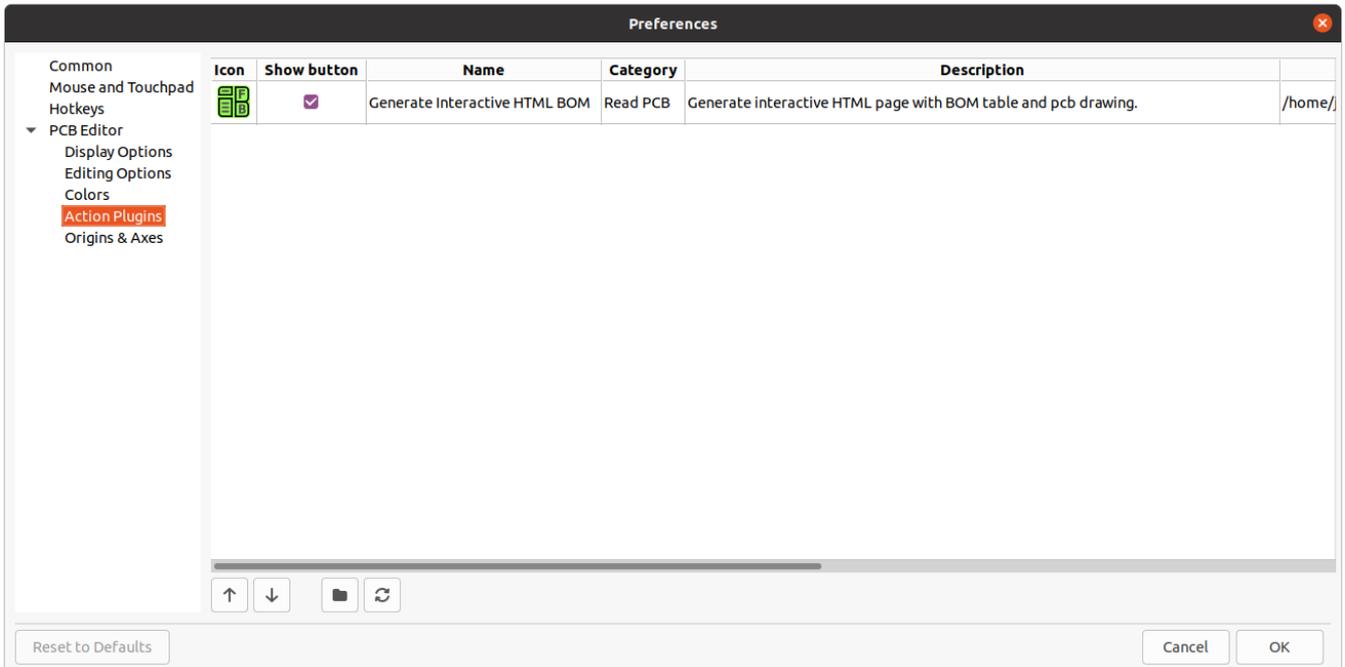
Color themes are stored in JSON files located in the `colors` subdirectory of the KiCad configuration directory. The "Open Theme Folder" button will open this location in your system file manager, making it easy to manage your installed themes. To install a new theme, place it in this folder and restart KiCad. The new theme will be available from the drop-down list of color themes if the file is a valid color theme file.

To create a new color theme, choose New Theme... from the drop-down list of color themes. Enter a name for your theme and then begin editing colors. The colors in the new theme will be copied from whatever theme was selected before you created the new theme.

To change a color, double-click or middle-click the color swatch in the list. The "Reset to Default" button will reset that color to its corresponding entry in the "KiCad Default" color theme.

Color themes are saved automatically; all changes are reflected immediately when you close the Preferences dialog. The window on the right side of the dialog shows a preview of how the selected theme will look.

# Action plugins

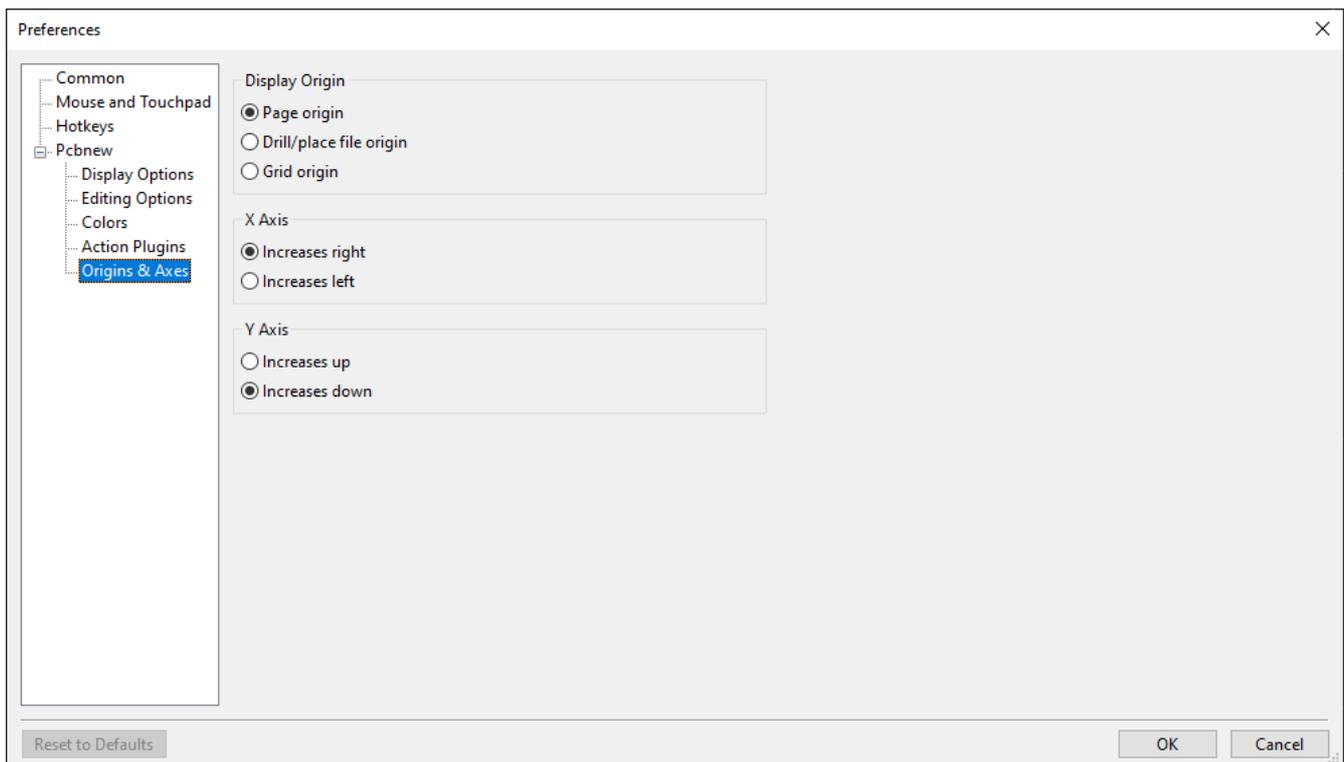


The KiCad PCB editor supports plugins written in Python that can perform actions on the board being edited. These plugins can be installed using the built-in Plugin and Content Manager (see the KiCad chapter for details) or by placing the plugin files inside the user plugins directory. See the Scripting section below for details.

Each plugin that is detected will be shown in a row on this preferences page. Plugins may show a button on the top toolbar of the PCB editor. If the "Show button" control is unchecked for a plugin, it may still be accessed from the Tools > External Plugins menu.

The arrow controls at the bottom of the list allow changing the order that the plugins appear in the toolbar and menu. The folder button will launch a file explorer to the plugin folder, to make installing new plugins easier. The refresh button will scan the plugin folder for any new or removed plugins and update the list.

## Origin & axes



**Display origin:** Determines which coordinate origin is used for coordinate display in the editing canvas. The page origin is fixed at the corner of the page. The drill/place file origin and the grid origin can be moved by the user.

**X axis:** Controls whether X-coordinates increase to the right or to the left.

**Y axis:** Controls whether Y-coordinates increase upwards or downwards.

## Text variables

KiCad supports text variables, which allow you to reference predefined variables by name in many kinds of text. KiCad will substitute the variable name with the text string assigned to the variable. Text variables can be used in schematic and PCB graphic text, symbol and footprint text, symbol and footprint fields, drawing sheet fields, custom DRC rules, and other text contexts. This substitution happens anywhere the variable name is used inside the variable replacement syntax of `${VARIABLENAME}`.

For example, you could create a variable named `VERSION` and set the text substitution to `1.0`. Now, in any text object on the PCB, you can enter `${VERSION}` and KiCad will display this as `1.0`. If you change the value to `2.0`, every text object that includes `${VERSION}` will be updated automatically. You can also mix regular text and variables: a text object with the text `Version: ${VERSION}` will be displayed as `Version: 1.0`.

You can define text variables in the [schematic](#) or [board setup](#) dialogs. These are referred to as project text variables because they are defined for the whole project, so a project text variable defined in the Schematic Editor can also be used in the Board Editor, and vice versa.

The PCB Editor also has a number of built-in system text variables, which are listed in the table below. System text variables may be available in some contexts and not others. There is an equivalent list of variables for the [Schematic Editor](#).

Variable name	Description
ADDR	Column (letter) and row (zero-indexed number) address of current table cell. This variable is only available in <a href="#">table</a> cells.
CELL(<row>, <col>)	Contents of the table cell indexed by <row> and <col> (zero-indexed). This variable is only available in <a href="#">table</a> cells.
COL	Column number (zero-indexed) of current table cell. This variable is only available in <a href="#">table</a> cells.
COMMENT1 - COMMENT9	Contents of drawing sheet's Comment<n> field.
COMPANY	Contents of drawing sheet's Company field.
CURRENT_DATE	Today's date, in ISO format.
FILENAME	Filename of the board, with a file extension.
FILEPATH	Full file path of the board, with a file extension.
ISSUE_DATE	Contents of drawing sheet's Issue Date field.
KICAD_VERSION	Current version of KiCad. This variable is only available in drawing sheet fields.
LAYER	Layer of the object. In footprint fields and text objects in footprints, this is the layer of the field/text object, not the layer of the parent footprint. In drawing sheet fields, this resolves to the plotted layer, for example F.Fab in a plot of the F.Fab layer and F.Cu in a plot of the F.Cu layer. This variable is only available in objects that have a layer, like footprint fields and PCB text. It is not available if there is no applicable layer, such as in DRC rules.
PAPER	Current sheet's paper size. This variable is only available in drawing sheet fields.
PROJECTNAME	Project name, without a file extension.
REVISION	Contents of drawing sheet's Revision field.
ROW	Row number (zero-indexed) of current table cell. This variable is only available in <a href="#">table</a> cells.
TITLE	Contents of drawing sheet's Title field.
VARIANT	Name of the currently selected variant.
VARIANT_DESC	Description of the currently selected variant.
VCSHASH	Full hash of the project's current Git commit. If the project is not part of a Git repository, resolves to "no hash".
VCSSHORTHASH	Short (8 character) hash of the project's current Git commit. If the project is not part of a Git repository, resolves to "no hash".
<variablename>	Contents of <a href="#">project text variable</a> <variablename> .

Variable name	Description
<fieldname>	<p>Contents of footprint field &lt;fieldname&gt;. Fields can only be accessed from within their parent object, so footprint fields can be accessed from other fields or text within the footprint.</p> <p>Both built-in footprint fields and user-defined fields from the corresponding symbol are available. Built-in footprint fields use all uppercase letters: for example, to access a footprint's value, use <code>\${VALUE}</code>.</p> <p>Built-in footprint fields are <code>FOOTPRINT_LIBRARY</code>, <code>FOOTPRINT_NAME</code>, <code>LAYER</code>, <code>NET_CLASS(&lt;pad_number&gt;)</code>, <code>NET_NAME(&lt;pad_number&gt;)</code>, <code>PIN_NAME(&lt;pad_number&gt;)</code>, <code>REFERENCE</code>, <code>SHORT_NET_NAME(&lt;pad_number&gt;)</code>, and <code>VALUE</code>.</p>
<refdes>: <fieldname>	<p>Contents of field &lt;fieldname&gt; in footprint &lt;refdes&gt;.</p> <p>Both built-in footprint fields and user-defined fields from the corresponding symbol are available. Built-in footprint fields use all uppercase letters: for example, to access the value of <code>U1</code>, use <code>\${U1:VALUE}</code>.</p> <p>Built-in footprint fields are <code>FOOTPRINT_LIBRARY</code>, <code>FOOTPRINT_NAME</code>, <code>LAYER</code>, <code>NET_CLASS(&lt;pad_number&gt;)</code>, <code>NET_NAME(&lt;pad_number&gt;)</code>, <code>PIN_NAME(&lt;pad_number&gt;)</code>, <code>REFERENCE</code>, <code>SHORT_NET_NAME(&lt;pad_number&gt;)</code>, and <code>VALUE</code>.</p> <p><b>Note:</b> If the footprint referenced by &lt;refdes&gt; is reannotated, i.e. receives a new reference designator, &lt;refdes&gt; in the text variable will be automatically updated to the new reference designator so that the text variable continues to reference the same footprint.</p>
DRC_ERROR <errorname>	<p>Generates a <b>DRC error</b> named &lt;errorname&gt;. Everything inside the braces resolves to an empty string, while everything after the braces is included in the descriptive text for the DRC violation. The text variable must be at the beginning of the text item.</p> <p>For example, a text item containing <code>\${DRC_ERROR TODO}Length match tracks</code> will display as the text "Length match tracks" and generate a DRC error named "TODO" with the description "Length match tracks".</p>
DRC_WARNING <warningname>	<p>Generates a DRC warning named &lt;warningname&gt;. This behaves the same as <code>DRC_ERROR</code>, except a warning is generated rather than an error.</p>

## Text expressions

KiCad has a system for evaluating text expressions, which allows you to manipulate text, evaluate mathematical expressions in strings, and dynamically create strings based on other information. Text expressions can be used in schematic and PCB graphic text, symbol and footprint text, symbol and footprint fields, drawing sheet fields, and other text contexts.

Text expressions are evaluated when the `@{expression}` syntax is used. Everything inside the braces is evaluated, and the entire expression, including the surrounding `@{}`, is replaced by the evaluated results. If the expression cannot be evaluated, the expression is not replaced and the original text is retained.

As a basic example, a text object containing `1 plus 1 is @{1 + 1}` will read `1 plus 1 is 2`.

Many expression functions are provided to perform more complex operations. The available expression functions are detailed in the tables below. For example, the absolute value of a number can be calculated using the `abs` function: `@{abs(-1)}` evaluates to `1`.

A single text item can contain multiple text evaluations, and evaluations can be nested. You can also combine text evaluations with [text variables](#). For example, `@{max(1, 2, 3)}`, `@{max(4, 5, 6)}` evaluates to `3`, `6`. If the text variable `var` is defined as `-10`, a text item containing `@{max(abs($var), 5)}` evaluates to `10`.

Numbers can be written in scientific notation, so `@{2e3 + 1}` evaluates to `2001`.

Text expressions support units. Results are always printed in SI units, and the unit is not printed. For example, `@{1mm + 1in}` evaluates to `26.4`.

In addition to numbers, some expression functions operate on strings. A string is text that is enclosed with single or double quotes, like `'string 1'` or `"string 2"`. An example of an expression function that operates on strings is `upper`, which converts a string to uppercase. `@{upper("string")}` evaluates to `STRING`.

You can concatenate multiple strings or strings and numbers using the `concat` expression function or using the `+` operator: `@{concat("hello ", "world")}` and `@{"hello " + "world"}` both evaluate to `hello world`. `@{concat("revision ", 1)}` and `@{"revision " + 1}` both evaluate to `revision 1`.

## Mathematical and logical expressions

The following arithmetic operations are available for operations between two numbers. The operators are all infix operators and obey order of operations. Parentheses can be used for grouping. For example, `@{3 + 4 * 3}` evaluates to `15`, and `@{(3 + 4) * 3}` evaluates to `21`.

Arithmetic operator	Description	Example
<code>+</code>	Addition  <b>Note:</b> <code>+</code> can also be used to concatenate two strings, or to concatenate a string and a number.	<code>@{1 + 1} = 2</code>
<code>-</code>	Subtraction	<code>@{1 - 1} = 0</code>
<code>*</code>	Multiplication	<code>@{2 * 2} = 4</code>
<code>/</code>	Division (the expression is not evaluated if the divisor is 0)	<code>@{3 / 2} = 1.5</code> <code>@{1 / 0} = @{1 / 0}</code>
<code>%</code>	Modulo (the expression is not evaluated if the divisor is 0)	<code>@{3 % 2} = 1</code> <code>@{3 % -2} = 1</code> <code>@{1 % 0} = @{1 % 0}</code>
<code>^</code>	Exponentiation (right-associative)	<code>@{2 ^ 3} = 8</code> <code>@{2 ^ 3 ^ 2} = 512</code>

The following expression functions are available for performing mathematical functions. For example, `@{abs(-1)}` evaluates to 1.

Expression function	Description	Example
<code>abs(x)</code>	Evaluates to the absolute value of <code>x</code> .	<code>@{abs(-1)}</code> = 1
<code>sqrt(x)</code>	Evaluates to the square root of <code>x</code> . The expression is not evaluated if <code>x</code> is negative.	<code>@{sqrt(4)}</code> = 2 <code>@{sqrt(-1)}</code> = <code>@{sqrt(-1)}</code>
<code>pow(x, y)</code>	Evaluates to <code>x</code> to the power of <code>y</code> .	<code>@{pow(2, 3)}</code> = 8
<code>floor(x)</code>	Evaluates to the greatest integer less than or equal to <code>x</code> .	<code>@{floor(3.7)}</code> = 3
<code>ceil(x)</code>	Evaluates to the smallest integer greater than or equal to <code>x</code> .	<code>@{ceil(3.2)}</code> = 4
<code>round(x, [precision])</code>	Evaluates to <code>x</code> rounded to the number of decimal places given by <code>precision</code> . <code>precision</code> is optional; if it is not given, <code>x</code> is rounded to the nearest integer.	<code>@{round(3.7)}</code> = 4 <code>@{round(3.2)}</code> = 3 <code>@{round(3.14159, 2)}</code> = 3.14
<code>min(...)</code>	Evaluates to the minimum of the given comma-separated values.	<code>@{min(5, 3, 8, 1)}</code> = 1
<code>max(...)</code>	Evaluates to the maximum of the given comma-separated values.	<code>@{max(5, 3, 8, 1)}</code> = 8
<code>sum(...)</code>	Evaluates to the sum of the given comma-separated values.	<code>@{sum(1, 2, 3, 4)}</code> = 10
<code>avg(...)</code>	Evaluates to the average of the given comma-separated values.	<code>@{avg(2, 4, 6)}</code> = 4
<code>random()</code>	Evaluates to a random number between 0 (inclusive) and 1 (exclusive).	$0 \leq \text{@{random()}} < 1$
<code>shunt(r1, r2)</code>	Evaluates to the equivalent resistance of the two parallel resistances <code>r1</code> and <code>r2</code> : $(r1*r2) / (r1+r2)$ .	<code>@{shunt(10k, 10k)}</code> = 5000
<code>db(ratio)</code>	Converts the power ratio <code>ratio</code> to decibels: $10*\log_{10}(\text{ratio})$ .	<code>@{db(2)}</code> $\approx$ 3
<code>dbv(ratio)</code>	Converts the voltage or current ratio <code>ratio</code> to decibels: $20*\log_{10}(\text{ratio})$ .	<code>@{dbv(2)}</code> $\approx$ 6
<code>fromdb(db)</code>	Converts the quantity <code>db</code> , in decibels, to a power ratio: $10^{(db/10)}$ .	<code>@{fromdb(3)}</code> $\approx$ 2

Expression function	Description	Example
fromdbv(db)	Converts the quantity db, in decibels, to a voltage or current ratio: $10^{(db/20)}$ .	<code>@{fromdbv(6)}</code> $\approx 2$
enearest(value, [series])	Evaluates to the closest E-series standard value to value from the specified series. Valid values for series are "E3", "E6", "E12", "E24", "E48", "E96", and "E192". series is optional; if it is not given, "E24" is used.	<code>@{enearest(5)}</code> = 5.1
eup(value, [series])	Evaluates to the closest E-series standard value greater than value from the specified series. Valid values for series are "E3", "E6", "E12", "E24", "E48", "E96", and "E192". series is optional; if it is not given, "E24" is used.	<code>@{eup(5)}</code> = 5.1
edown(value, [series])	Evaluates to the closest E-series standard value less than value from the specified series. Valid values for series are "E3", "E6", "E12", "E24", "E48", "E96", and "E192". series is optional; if it is not given, "E24" is used.	<code>@{edown(5)}</code> = 4.7

## Boolean expressions

The following boolean operations are available. These operations evaluate to 1 if the expression is true or 0 if it is false.

Arithmetic operator	Description	Example
<	Less than	<code>@{1 &lt; 2}</code> = 1
<=	Less than or equal to	<code>@{1 &lt;= 2}</code> = 1
>	Greater than	<code>@{1 &gt; 2}</code> = 0
>=	Greater than or equal to	<code>@{1 &gt;= 2}</code> = 0
==	Equal to	<code>@{1 == 2}</code> = 0
!=	Not equal to	<code>@{1 != 2}</code> = 1

The following expression functions are available for operating on the result of a boolean or mathematical expression.

**NOTE**

Strings are text that is surrounded by single or double quotes. For example, "this is a string" and 'this is also a string'.

Expression function	Description	Example
upper(str)	Evaluates to str converted to uppercase.	<code>@{upper("hello world")} = HELLO WORLD</code>
lower(str)	Evaluates to str converted to lowercase.	<code>@{lower("HELLO WORLD")} = hello world</code>
concat(...)	Evaluates to the given comma-separated strings or numbers concatenated together.  <b>Note:</b> strings and numbers can also be concatenated using the + operator.	<code>@{concat("Hello", " ", "World")} = Hello World</code>  <code>@{concat("Count: ", 42, " items")} = Count: 42 items</code>  <code>@{"Count: " + 42 + " items"} = Count: 42 items</code>
beforefirst(str, c)	Evaluates to the portion of str before the first occurrence of the character c in str.	<code>@{beforefirst("hello.world.txt", ".")} = hello</code>
beforelast(str, c)	Evaluates to the portion of str before the last occurrence of the character c in str.	<code>@{beforelast("hello.world.txt", ".")} = hello.world</code>
afterfirst(str, c)	Evaluates to the portion of str after the first occurrence of the character c in str.	<code>@{afterfirst("hello.world.txt", ".")} = world.txt</code>
afterlast(str, c)	Evaluates to the portion of str after the last occurrence of the character c in str.	<code>@{afterlast("hello.world.txt", ".")} = txt</code>
format(num, [precision])	Formats num as a string containing num rounded to precision decimal places. precision is optional; if it is not given, num is rounded to 2 decimal places.  <b>Note:</b> format() behaves exactly the same as fixed().	<code>@{format(3.14159)} = 3.14</code>  <code>@{format(3.14159, 3)} = 3.142</code>  <code>@{format(1234.5)} = 1234.50</code>
fixed(num, [precision])	Formats num as a string containing num rounded to precision decimal places. precision is optional; if it is not given, num is rounded to 2 decimal places.  <b>Note:</b> fixed() behaves exactly the same as format().	<code>@{fixed(3.14159)} = 3.14</code>  <code>@{fixed(3.14159, 3)} = 3.142</code>  <code>@{fixed(1234.5)} = 1234.50</code>

Expression function	Description	Example
<code>currency(amount, [symbol])</code>	Formats <code>amount</code> as a string containing the currency symbol specified by <code>symbol</code> and a value with two decimal places. <code>symbol</code> is optional; if it is not given, "\$" is used.	<code>@{currency(1234.56)}</code> = \$1234.56  <code>@{currency(999.99, "€")}</code> = €999.99

## Date and time formatting

The following expression functions are available for manipulating dates and times.

Expression function	Description	Example
<code>today()</code>	Evaluates to the current date, represented as the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01).  <b>Note:</b> this value can be formatted in various ways using the <code>dateformat()</code> function. You can add or subtract numbers to this value to get dates relative to the current date.	
<code>now()</code>	Evaluates to the current timestamp, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1970-01-01).  <b>Note:</b> you can add or subtract numbers to this value to get timestamps relative to the current timestamp.	
<code>dateformat(days, [format])</code>	Formats <code>days</code> as a string depending on the specified date <code>format</code> . <code>days</code> is a number representing the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01). Valid values for <code>format</code> are "ISO", "US", "EU", "Chinese", "Japanese", "long", and "short". <code>format</code> is optional; if it is not given, "ISO" is used.	<code>@{dateformat(0)}</code> = 1970-01-01  <code>@{dateformat(0, "ISO")}</code> = 1970-01-01  <code>@{dateformat(0, "short")}</code> = Jan 1, 1970
<code>weekdayname(days)</code>	Evaluates to the day of the week of the date represented by <code>days</code> . <code>days</code> is a number representing the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01).	<code>@{weekdayname(0)}</code> = Thursday

## Version control expressions

The following expression functions are available for extracting version control system information about the project. If the project is not under version control, each expression function evaluates to `<unknown>`.

Git is the only supported version control system for these expression functions.

Expression function	Description
<code>vcsidentifier([length])</code>	Evaluates to the current HEAD commit identifier, truncated to <code>length</code> characters. <code>length</code> is optional; if it is not given, the commit identifier is truncated to 40 characters. The minimum <code>length</code> is 4 characters.
<code>vcsbranch()</code>	Evaluates to the current branch name.
<code>vcsauthor()</code>	Evaluates to the current HEAD commit's author's name.
<code>vcsauthoremail()</code>	Evaluates to the current HEAD commit's author's email address.
<code>vcscommitter()</code>	Evaluates to the current HEAD commit's committer's name.
<code>vcscommitteremail()</code>	Evaluates to the current HEAD commit's committer's email address.
<code>vcscommitdate([format])</code>	Evaluates to the current HEAD commit's commit date. Valid values for <code>format</code> are "ISO", "US", "EU", "Chinese", "Japanese", "long", and "short". <code>format</code> is optional; if it is not given, "ISO" is used.
<code>vcsnearestlabel([match], [anyTags])</code>	Evaluates to the name of the matching tag closest to the HEAD commit. <code>match</code> is an optional pattern (glob-style) to match tag names against; if it is given, only tag names that match the pattern will be considered. If it is not given, this is equivalent to the pattern <code>*</code> . If <code>anyTags</code> is 0 or not given, only annotated tags will be considered. If <code>anyTags</code> is 1, all tags will be considered.
<code>vcslabeldistance([match], [anyTags])</code>	Evaluates to the number of commits between HEAD and the matching tag closest to the HEAD commit. <code>match</code> is an optional pattern (glob-style) to match tag names against; if it is given, only tag names that match the pattern will be considered. If it is not given, this is equivalent to the pattern <code>*</code> . If <code>anyTags</code> is 0 or not given, only annotated tags will be considered. If <code>anyTags</code> is 1, all tags will be considered.
<code>vcsdirty([includeUntracked])</code>	Evaluates to 1 if the repository is dirty, or 0 if it is not. If <code>includeUntracked</code> is 0 or not given, untracked files will not contribute to the repository's dirty status. If <code>includeUntracked</code> is 1, untracked files will contribute.
<code>vcsdirtysuffix([suffix], [includeUntracked])</code>	Evaluates to the given <code>suffix</code> if the repository is dirty or an empty string if it is not. <code>suffix</code> is optional; if it is not given, <code>-dirty</code> is used. If <code>includeUntracked</code> is 0 or not given, untracked files will not contribute to the repository's dirty status. If <code>includeUntracked</code> is 1, untracked files will contribute.

Expression function	Description
<code>vcsfileidentifier(file, [length])</code>	Evaluates to the commit identifier of the last commit that modified the specified file, truncated to length characters. length is optional; if it is not given, the commit identifier is truncated to 40 characters. The minimum length is 4 characters.
<code>vcsfileauthor(file)</code>	Evaluates to the name of the author of the last commit that modified the specified file.
<code>vcsfileauthoremail(file)</code>	Evaluates to the email address of the author of the last commit that modified the specified file.
<code>vcsfilecommitter(file)</code>	Evaluates to the name of the committer of the last commit that modified the specified file.
<code>vcsfilecommitteremail(file)</code>	Evaluates to the email address of the committer of the last commit that modified the specified file.
<code>vcsfilecommitdate(file, [format])</code>	Evaluates to the commit date of the last commit that modified the specified file. Valid values for format are "ISO", "US", "EU", "Chinese", "Japanese", "long", and "short". format is optional; if it is not given, "ISO" is used.

## Custom design rules

KiCad's custom design rule system allows creating design rules that are more specific than the generic rules available in the Constraints page of the Board Setup dialog. Custom design rules have many applications, but in general they are used to apply certain rules to a portion of the board, such as a specific net or net class, a specific area, or a specific footprint.

Custom design rules are stored in a separate file with the extension `kicad_dru`. This file is created automatically when you start adding custom rules to a project. If you are using custom rules in your project, make sure to save the `kicad_dru` file along with the `kicad_pcb` and `kicad_pro` files when making backups or committing to a version control system.

### NOTE

The `kicad_dru` file is managed automatically by KiCad and should not be edited with an external text editor. Always use the Custom Rules page of the Board Setup dialog to edit custom design rules.

## The Custom Rules editor

The custom rules editor is located in the Board Setup dialog and provides a text editor for entering custom rules, a syntax checker that will test your custom rules and note any errors, and a syntax help dialog that contains a quick reference to the custom rules language and some example rules.

The custom rules editor also provides context-sensitive autocomplete to suggest valid keywords and properties. The autocomplete suggestion menu appears automatically, but it can also be opened manually by pressing `Ctrl + Space`.

It is a good idea to use the **Check rule syntax** button after editing custom rules to make sure there are no syntax errors. Any errors in the custom rules will prevent the design rule checker from running.

## Custom rule syntax

The custom design rule language is based on s-expressions and allows you to create design constraints that are not possible with the built-in constraints. Each design rule generally contains a **condition** defining what objects to match and a **constraint** defining the rule to be applied to the matched objects.

The language uses parentheses ( ( and ) ) to define clauses of related keywords and values. Parentheses must always be matched: for every ( there must be a matching ). Inside a clause, keywords and values are separated by whitespace (spaces, tabs, and newlines). By convention, a single space is used, but any number of whitespace characters between keywords and values is acceptable. In places where text strings are valid, strings without any whitespace may be quoted with " or ', or unquoted. Strings that contain whitespace must always be quoted. Newlines cannot be used within a quoted string. Where nested quotes are required, a single level of nesting is possible by using " for the outer quote character and ' for the inner (or vice versa). Newlines between clauses are not required, but are typically used in examples for clarity.

In the syntax descriptions below, items in <angle brackets> represent keywords or values that must be present and items in [square brackets] represent keywords or values that are optional or only sometimes required.

The Custom Rules file must start with a version header defining the version of the rules language. As of KiCad 10.0, the version is 1. The syntax of the version header is (version <number>). So in KiCad 10.0 the header should read:

```
(version 1)
```

After the version header, you can enter any number of rules. Rules are evaluated in reverse order, meaning the last rule in the file is checked first. Once a matching rule is found for a given set objects being tested, no further rules will be checked. In practice, this means that more specific rules should be later in the file, so that they are evaluated before more general rules.

For example, if you create one rule that limits the minimum clearance between tracks in the net HV and tracks in any other net and a second rule that limits the minimum clearance for all objects inside a certain rule area, make sure the first rule appears later in the custom rules file than the second rule. Otherwise tracks in the HV net could have the wrong clearance if they fall inside the rule area.

Each rule must have a name and one or more **constraint** clauses. The name can be any string and is used to refer to the rule in DRC reports. The **constraint** defines the behavior of the rule. Rules may also have a **condition** clause that determines which objects should have the rule applied, an optional **layer** clause which specifies which board layers the rule applies to, and an optional **severity** clause which specifies the severity of the resulting DRC violation.

```
(rule <name>
  [(severity <severity>)]
  [(layer <layer_name>)]
  [(condition <expression>)]
  (constraint <constraint_type> [constraint_arguments]))
```

The custom rules file may also include comments to describe rules. Comments are denoted by any line that begins with the # character (not including whitespace). You can press `Ctrl` + `/` to comment or uncomment lines automatically.

```
# Clearance for 400V nets to anything else
(rule HV
  (condition "A.hasNetclass('HV')")
  (constraint clearance (min 1.5mm)))
```

Custom rules can reference [text variables](#). These are not technically part of the custom rules syntax, but are resolved in a preprocessing step before evaluating the rules file. Because text variables are resolved as a simple text replacement, they can be used anywhere in the DRC rules, as long as all rules are syntactically valid after the variables have been resolved.

Text variables could be used, for example, to define a project-wide value that is used in one or more constraints, like `(constraint clearance (min ${hv_clearance}))`, where `${hv_clearance}` is a text variable defined in the project as `${3 mm}`.

Not all text variables can be resolved in custom rules. Some text variables resolve to different values depending on the object they are a part of, such as `${LAYER}` or `${<fieldname>}`. Because text variables in custom rules are resolved before evaluating any rules or applying rules to specific objects in the board, such text variables cannot be resolved in custom rules. Any text variables that cannot be resolved will remain in their unresolved state and cause a syntax error when the rules file is evaluated.

## Layer Clause

The `layer` clause determines which layers the rule will work on. While the layer of objects can be tested in the `condition` clause as described below, using the `layer` clause is more efficient.

The value in the `layer` clause can be any board layer name, or the shortcut keywords `outer` to match the front and back copper layers ( `F.Cu` and `B.Cu`) and `inner` to match any internal copper layers.

If the `layer` clause is omitted, the rule will apply to all layers.

Some examples:

```
# Do not allow footprints on back layer (no condition clause means this rule always
applies)
(rule "Top side footprints only"
  (layer B.Cu)
  (constraint disallow footprint))

# This rule does the same thing, but is less efficient
(rule "Top side footprints only"
  (condition "A.Layer == 'B.Cu'")
  (constraint disallow footprint))

# Larger clearance on outer layers (inner layer clearance set by board minimum clearance)
(rule "clearance_outer"
  (layer outer)
  (constraint clearance (min 0.25mm)))
```

## Severity Clause

The `severity` clause sets the DRC violation severity whenever the rule is violated.

Possible values are `error`, `warning`, `ignore`, and `exclusion`. Ignored rules are not observed by the interactive router and violations are not shown in the DRC dialog. However, ignored rules are evaluated for matching and therefore can still override earlier rules. Errors, warnings, and excluded rules are all observed by the interactive router, and violations are displayed in the DRC dialog when the appropriate filters are selected.

### WARNING

Setting a rule's severity to `ignore` does not disable the rule; only the effects of the rule are disabled. The rule is still evaluated and can still override previous rules.

## Condition Clauses

The `condition` clause determines which objects which objects the rule applies to. If a rule has a condition clause, the rule will apply to any objects that match the condition. If a rule does not have any condition clauses, it will apply unconditionally.

The rule **condition** is an expression contained inside a text string (and therefore usually surrounded by quotes in order to allow whitespace for clarity). The expression is evaluated against each pair of objects that is being tested by the design rule checker. For example, when checking for clearance between copper objects, each copper object (track segment, pad, via, etc.) on each net is checked against other copper objects on other nets. If a custom rule exists where the expression matches the two given copper objects and the constraint defines a copper clearance, this custom rule could be used to determine the required clearance between the two objects.

The objects being tested are referred to as `A` and `B` in the expression language. The order of the two objects is not important because the design rule checker will test both possible orderings. For example, you can write a rule that assumes that `A` is a track and `B` is a via. There are some expression functions that test both objects together; these use `AB` as the object name.

The expression in a condition must resolve to a boolean value (true or false). If the expression resolves to true, the rule is applied to the given objects.

Each object being tested has **properties** that can be compared, as well as **functions** that can be used to perform certain tests. The syntax for using properties and functions is `<object>.<property>` and `<object>.<function>([arguments])` respectively.

**NOTE**

When you type `<object>.` in the text editor (A., B., or AB.), an autocomplete list will open that contains all the object properties that can be used.

The object properties and functions are compared using **boolean** and **relational operators** to result in a boolean expression. The following operators are supported:

<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code>&gt;</code> , <code>&gt;=</code>	Greater than, greater than or equal to
<code>&lt;</code> , <code>&lt;=</code>	Less than, less than or equal to
<code>&amp;&amp;</code>	And
<code>  </code>	Or
<code>!</code>	Not (unary)

For example, `A.NetName == 'VDD'` will apply to any objects that are part of the "VDD" net and `A.NetName != B.NetName` will apply to any objects that have different net names. Parentheses can be used to clarify the order of operations in complex expressions but they are not required. All the boolean operators have the same precedence and are evaluated in order from left to right.

To test a boolean property, evaluate the property itself, without comparing it to a boolean literal like `true` or `false` (which don't exist in the DRC rules language). For example, to test if a footprint's boolean `Do_not_populate` property is set, the boolean expression `A.Do_not_populate` by itself is sufficient. It will resolve to a true value if the footprint's DNP attribute is set, and a false value otherwise. To check if a boolean is false, use the `!` operator (unary not): `!A.Do_not_populate` will resolve to a true value if the DNP attribute is unset, and a false value otherwise.

Some properties represent a physical measurement, such as a size, angle, length, position, etc. On these properties, **unit suffixes** can be used in the custom rules language to specify what units are being used. If no unit suffix is used, the internal representation of the property will be used instead (nanometers for distances and degrees for most angles). The following suffixes are supported:

<code>mm</code>	Millimeters
<code>mil</code> , <code>th</code>	Thousandths of an inch (mils)
<code>in</code> , <code>"</code>	Inches
<code>deg</code>	Degrees
<code>rad</code>	Radians

**NOTE**

The units used in custom design rules are independent of the display units in the PCB editor.

Numeric conditions can use simple math expressions, for example `(condition "A.Hole_Size_X == 1.0mm + 0.1mm")`.

## Constraint Clauses

The `constraint` clause of the rule defines the behavior of the rule on the objects that are matched by the condition. Each constraint clause has a **constraint type** and one or more arguments that set the behavior of the constraint. A single rule may have multiple constraint clauses, in order to set multiple constraints (for example, `clearance` and `track_width`) for objects that match the same rule conditions.

Many constraints take arguments that specify a physical measurement or quantity. These constraints support minimum, optimal, and maximum value specification (abbreviated "min/opt/max"). The **minimum** and **maximum** values are used for design rule checking: if the actual value is less than the minimum or is greater than the maximum value in the constraint, a DRC error is created. The **optimal** value is only used for some constraints, and informs KiCad of a "best" value to use by default. For example, the optimal `diff_pair_gap` is used by the router when placing new differential pairs. No errors will be created if the differential pair is later modified such that the gap between the pair is different from the optimal value, as long as the gap is between the minimum and maximum values (if these are specified). In all cases where a min/opt/max value is accepted, any or all of the minimum, optimal, and maximum value can be specified.

Min/opt/max values are specified as `(min <value>)`, `(opt <value>)`, and `(max <value>)`. For example, a track width constraint may be written as `(constraint track_width (min 0.5mm) (opt 0.5mm) (max 1.0mm))` or simply `(constraint track_width (min 0.5mm))` if only the minimum width is to be constrained.

Numeric constraint values can use simple math expressions, for example `(constraint clearance (min 0.5mm + 0.1mm))`.

Constraint type	Argument type	Description
<code>annular_width</code>	min/max	Checks the width of annular rings on vias and pads.
<code>assertion</code>	boolean expression	Checks that the boolean expression is true. If the expression is false, a DRC error will be created. The expression can use any of the properties listed in the Object Properties section.
<code>bridged_mask</code>		Checks for solder mask bridges between copper items. This constraint does not take a min/opt/max value. In combination with a severity clause, this constraint can be used to allow or disallow solder mask bridging in various conditions.
<code>clearance</code>	min	Specifies the <b>electrical</b> clearance between copper objects of different nets. (See <code>physical_clearance</code> if you wish to specify clearance between objects regardless of net.)  To allow copper objects to overlap (collide), create a <code>clearance</code> constraint with the <code>min</code> value less than zero (for example, <code>-1</code> ).

Constraint type	Argument type	Description
connection_width	min	Checks the width of connections between pads and zones. An error will be generated for each pad connection that is narrower than the min value.
courtyard_clearance	min	Checks the clearance between footprint courtyards and generates an error if any two courtyards are closer than the min distance. If a footprint does not have a courtyard shape, no errors will be generated from this constraint.  To allow courtyard objects to overlap (collide), create a courtyard_clearance constraint with the min value less than zero (for example, -1 ).
creepage	min	Specifies the creepage between copper objects of different nets.
diff_pair_gap	min/opt/max	Checks the gap between parallel tracks in a differential pair. The opt setting is used by the interactive router for placing new differential pairs. An error will be generated if the spacing between tracks in a differential pair is outside of the min and max settings. Differential pair gap is not tested on non-parallel portions of a differential pair (for example, the fanout from a component).
diff_pair_uncoupled	max	Checks the distance that a differential pair track is routed uncoupled from the other polarity track in the pair (for example, where the pair fans out from a component, or becomes uncoupled to pass around another object such as a via). An error will be generated for each differential pair with an uncoupled distance that is greater than the max value. Differential pair tracks are considered uncoupled if they are not parallel or if they are outside the range set by a diff_pair_gap constraint.
disallow	track via micro_via buried_via pad zone text graphic hole footprint	Specify one or more object types to disallow, separated by spaces. For example, (constraint disallow track) or (constraint disallow track via pad) . If an object of this type matches the rule condition, a DRC error will be created. This constraint is essentially the same as a keepout rule area, but can be used to create more specific keepout restrictions.

Constraint type	Argument type	Description
edge_clearance	min	<p>Checks the clearance between objects and the board edge.</p> <p>This can also be thought of as the "milling tolerance" as the board edge will include all graphical items on the <code>Edge.Cuts</code> layer as well as any <b>oval</b> pad holes. (See <code>physical_hole_clearance</code> for the drilling tolerance.)</p> <p>To allow objects to overlap (collide) with the board edge, create an <code>edge_clearance</code> constraint with the <code>min</code> value less than zero (for example, <code>-1</code>).</p>
hole_clearance	min	Checks the clearance between a drilled hole in a pad or via and copper objects on a different net. The clearance is measured from the diameter of the hole, not its center.
hole_size	min/opt/max	Checks the size (diameter) of a drilled hole in a pad or via. For oval holes, the smaller (minor) diameter will be tested against the <code>min</code> value (if specified) and the larger (major) diameter will be tested against the <code>max</code> value (if specified). The <code>opt</code> value is used by the interactive router as the default via hole size for nets that match the rule condition.
hole_to_hole	min	<p>Checks the clearance between mechanically-drilled holes in pads and vias. The clearance is measured between the diameters of the holes, not between their centers.</p> <p>This constraint is solely for the protection of drill bits. The clearance between <b>laser-drilled</b> (microvias) and other non-mechanically-drilled holes is not checked, nor is the clearance between <b>milled</b> (oval-shaped) and other non-mechanically-drilled holes.</p>
length	min/opt/max	Checks the total routed length for the nets that match the rule condition and generates an error for each net that is below the <code>min</code> value (if specified) or above the <code>max</code> value (if specified) of the constraint. The <code>opt</code> value sets a target length that is used by the <a href="#">length tuning tool</a> for any nets that match the rule condition.
min_resolved_spokes	0 1 2 3 4	Checks the total number of connections (spokes) to a pad. An error will be raised for each pad that has fewer than the specified number of spokes.

Constraint type	Argument type	Description
physical_clearance	min	<p>Checks the clearance between two objects, regardless of their nets. This includes objects with the same net and objects on non-copper layers. Only objects on physical layers and courtyard layers are checked: this means copper, adhesive, paste, silkscreen, mask, courtyard, and edge cut layers. Physical clearance is only checked between objects on the same layer, except for objects on <code>Edge.Cuts</code>, which are treated as if they are on all layers. In other words, physical clearance can be checked between objects on <code>Edge.Cuts</code> and objects on any of the other physical layers.</p> <p>While this can perform more general-purpose checks than <code>clearance</code>, it is much slower. Use <code>clearance</code> where possible.</p>
physical_hole_clearance	min	<p>Checks the clearance between a drilled hole in a pad or via and another object, regardless of net. The clearance is measured from the diameter of the hole, not its center.</p> <p>This can also be thought of as the "drilling tolerance" as it only includes <b>round</b> holes (see <code>edge_clearance</code> for the milling tolerance).</p>
silk_clearance	min	<p>Checks the clearance between objects on silkscreen layers and other objects.</p> <p>To allow silkscreen objects to overlap (collide) with other objects, create a <code>silk_clearance</code> constraint with the <code>min</code> value less than zero (for example, <code>-1</code>).</p>

Constraint type	Argument type	Description
skew	min/opt/max/within_diff_pairs	Checks the total skew for the nets that match the rule condition, that is, the difference between the length of each net and the longest net that is matched by the rule. If the difference between the longest net and the length of any one net is above the constraint <code>max</code> value, an error will be generated. This constraint also sets a target skew that is used by the <a href="#">skew tuning tool</a> for any nets that match the rule condition. The target skew is the <code>opt</code> value, if specified, or the <code>min</code> value if not. If neither <code>min</code> nor <code>opt</code> is specified, the target skew is 0. If the option <code>within_diff_pairs</code> is specified, the skew will be tested separately for every valid differential pair in the nets matching the rule. If <code>within_diff_pairs</code> is not specified, the skew will be tested across all matching nets (e.g. for skew tuning a bus).
solder_mask_expansion	opt	Specifies the solder mask expansion for pads, shapes and tracks.
solder_paste_abs_margin	opt	Specifies the absolute solder paste clearance for pads. Usually negative to inset the paste. The final solder paste clearance will be the absolute clearance plus the relative clearance.
solder_paste_rel_margin	opt	Specifies the relative solder paste clearance for pads. Usually negative to inset the paste. The final solder paste clearance will be the absolute clearance plus the relative clearance.
text_height	min/max	Checks the height of text, including text boxes. An error will be generated for each text item that has a height below the <code>min</code> value (if specified) or above the <code>max</code> value (if specified).
text_thickness	min/max	Checks the thickness of text, including text boxes. An error will be generated for each text item that has a thickness below the <code>min</code> value (if specified) or above the <code>max</code> value (if specified).

Constraint type	Argument type	Description
thermal_relief_gap	min	Specifies the width of the gap between a pad and a zone with a thermal-relief connection.
thermal_spoke_width	opt	Specifies the width of the spokes connecting a pad to a zone with a thermal-relief connection.
track_angle	min/max	Checks the angle between two connected track segments. An error will be generated for each connected pair with an angle below the min value (if specified) or above the max value (if specified).
track_segment_length	min/max	Checks the length of track and arc segments. An error will be generated for each segment that has a length below the min value (if specified) or above the max value (if specified).
track_width	min/opt/max	Checks the width of track and arc segments. An error will be generated for each segment that has a width below the min value (if specified) or above the max value (if specified). The opt value is used by the interactive router as the default track width for nets that match the rule condition.
via_count	min/max	Counts the number of vias on every net matched by the rule condition. An error will be generated for each net that has fewer vias than the min value (if specified) or more than the max value (if specified).
via_dangling		Checks for vias that are unconnected or connected on only one layer. This constraint does not take a min/opt/max value. In combination with a severity clause, this constraint can be used to allow or disallow dangling vias in various conditions.
via_diameter	min/opt/max	Checks the diameter of vias. An error will be generated for each via that has a diameter below the min value (if specified) or above the max value (if specified). The opt value is used by the interactive router as the default via diameter for nets that match the rule condition.
zone_connection	solid thermal_reliefs none	Specifies the connection to be made between a zone and a pad.

## Object property and function reference

The following properties can be tested in custom rule expressions:

### Common Properties

These properties apply to all PCB objects.

Property	Data type	Description
Layer	string	The board layer on which the object exists. For objects that exist on more than one layer, this property will return the first layer (for example, F.Cu for most through-hole pads/vias).
Locked	boolean	True if the object is locked.
Parent	string	Returns the unique identifier of the parent object of this object.
Position_X	dimension	The position of the object's origin in the X-axis. Note that the origin of an object is not always the same as the center of the object's bounding box. For example, the origin of a footprint is the location of the (0, 0) coordinate of that footprint in the footprint editor, but the footprint may have been designed such that this location is not in the center of the courtyard shape.
Position_Y	dimension	The position of the object's origin in the Y-axis. Note that KiCad always uses Y-coordinates that increase from the top to bottom of the screen internally, even if you have configured your settings to show the Y-coordinates increasing from bottom to top.
Type	string	One of "Bitmap", "Dimension", "Footprint", "Graphic", "Group", "Leader", "Pad", "Target", "Text", "Text Box", "Track", "Via", or "Zone".

## Connected Object Properties

These properties apply to copper objects that can have a net assigned (pads, vias, zones, tracks).

Property	Data type	Description
Net	integer	<p>The net code of the copper object.</p> <p>Note that net codes should not be relied upon to remain constant: if you need to refer to a specific net in a rule, use <code>NetName</code> instead. <code>Net</code> can be used to compare the nets of two objects with better performance, for example <code>A.Net == B.Net</code> is faster than <code>A.NetName == B.NetName</code>.</p>

Property	Data type	Description
NetClass	string	<p>The list of all net classes for the copper object. This is a priority ordered, comma delimited list where a net has multiple net classes assigned.</p> <p>Note that this list may include the <code>Default</code> net class, even if other net classes have been explicitly assigned to the net, because the <code>Default</code> net class provides fallback properties and design rules for any properties not defined by explicit net classes. See the <a href="#">net class documentation</a> for more details.</p> <p>In an expression, an object's <code>NetClass</code> property and a net class string are equal to each other if the string matches any of the net classes in the list, or if the string matches the full ordered list. For example, if an object belongs to the <code>HV</code> and <code>Default</code> net classes, all of the following expressions are true:</p> <ul style="list-style-type: none"> <li><code>A.NetClass == 'HV'</code></li> <li><code>A.NetClass == 'Default'</code></li> <li><code>A.NetClass == 'HV,Default'</code></li> </ul> <p>The following expressions are false, however:</p> <ul style="list-style-type: none"> <li><code>A.NetClass == 'LV'</code></li> <li><code>A.NetClass == 'LV,Default'</code></li> <li><code>A.NetClass == 'Default,HV'</code></li> </ul> <p>You can also check if a copper object is a member of a particular net class, regardless of any other net classes it may be a part of, using <code>hasNetclass(&lt;netclass&gt;)</code>. You can check if a copper object's net classes exactly match a given list of net classes using <code>hasExactNetclass(&lt;netclass list&gt;)</code>.</p>
NetName	string	<p>The name of the net for the copper object.</p> <p>Note that <code>Net</code> can be used instead in some situations for better performance; see the notes under <code>Net</code>.</p>
Curved_Edges	boolean	<p>True if curved edges are enabled for teardrops connected to the object.</p>
Enable_Teardrops	boolean	<p>True if teardrops are enabled for the object.</p>

Property	Data type	Description
Prefer_Zone_Connections	boolean	True if the "Prefer zone connections" property is set for the object.
Allow_Teardrops_To_Span_Two_Tracks	boolean	True if the "Allow teardrops to span two tracks" property is set for the object.
Best_Length_Ratio	double	Best ratio of teardrop length to object size for teardrops connected to the object.
Best_Width_Ratio	double	Best ratio of teardrop width to object size for teardrops connected to the object.
Max_Length	dimension	Maximum length dimension for teardrops connected to the object.
Max_Width	dimension	Maximum width dimension for teardrops connected to the object.
Max_Width_Ratio	double	Maximum allowable ratio of object size to track width for teardrops connected to the object.

## Footprint Properties

These properties apply to footprints.

Property	Data type	Description
Clearance_Override	dimension	The copper clearance override set for the footprint.

Property	Data type	Description
Component_Class	string	<p>The name of the component class set for the footprint. This is an alphabetically ordered, comma delimited list where a footprint has multiple component classes assigned.</p> <p>In an expression, a footprint's <code>Component_Class</code> property and a component class string are equal to each other if the string matches any of the component classes in the list, or if the string matches the full ordered list. For example, if a footprint belongs to the <code>Connector</code> and <code>HV</code> component classes in that order, all of the following expressions are true:</p> <ul style="list-style-type: none"> <li>• <code>A.Component_Class == 'Connector'</code></li> <li>• <code>A.Component_Class == 'HV'</code></li> <li>• <code>A.Component_Class == 'Connector,HV'</code></li> </ul> <p>The following expressions are false, however:</p> <ul style="list-style-type: none"> <li>• <code>A.Component_Class == 'LV'</code></li> <li>• <code>A.Component_Class == 'Connector,LV'</code></li> <li>• <code>A.Component_Class == 'HV,Connector'</code></li> </ul> <p>Note that while <code>Component_Class</code> is a footprint property, footprint children, such as pads or graphics, are considered to be members of any component class that their parent footprint is a member of. For example, if a footprint is has the component class <code>HV</code>, the condition <code>A.Component_Class == 'HV'</code> is true both for the footprint as well as for its pads and other children.</p> <p>You can also check if an object is part of a footprint with a specific component class using the <code>memberOfFootprint('\${Class:x}')</code> function.</p>
Do_not_Populate	boolean	True if the footprint's "Do not populate" attribute is set.
Exclude_From_Position_Files	boolean	True if the footprint's "Exclude from position files" attribute is set.
Exclude_From_Bill_of_Materials	boolean	True if the footprint's "Exclude from bill of materials" attribute is set.

Property	Data type	Description
Keywords	string	The "Keywords" from the library footprint.
Library_Description	string	The footprint's description in the footprint library. This is the footprint's description property, not the contents of the footprint field named Description.
Library_Link	string	The link to the library footprint in library_name: footprint_name format.
Not_in_Schematic	boolean	True if the footprint's "Not in schematic" attribute is set.
Orientation	double	The orientation (rotation) of the footprint in degrees.
Reference	string	The reference designator of the footprint.  Note that while footprints have a Reference property, footprint child objects (such as pads) do not. To check if an object belongs to a footprint with a specific reference, use the <code>memberOfFootprint('x')</code> function.
Solderpaste_Margin_Override	dimension	The solder paste margin override set for the footprint.
Solderpaste_Margin_Ratio_Override	dimension	The solder paste margin ratio override set for the footprint.
Thermal_Relief_Gap	dimension	The thermal relief gap set for the footprint.
Thermal_Relief_Width	dimension	The thermal relief connection width set for the footprint.
Value	string	The contents of the "Value" field of the footprint.
Zone_Connection_Style	string	One of "Inherited", "None", "Thermal reliefs" or "Solid".

## Pad Properties

These properties apply to footprint pads.

Property	Data type	Description
Clearance_Override	dimension	The copper clearance override set for the pad.
Fabrication_Property	string	One of "None", "BGA pad", "Fiducial, global to board", "Fiducial, local to footprint", "Test point pad", "Heatsink pad", "Castellated pad".

Property	Data type	Description
Orientation	double	The orientation (rotation) of the pad in degrees.
Pad_Number	string	The "number" of a pad, which can be a string (for example "A1" in a BGA).
Pad_Shape	string	One of "Circle", "Rectangle", "Oval", "Trapezoid", "Rounded rectangle", "Chamfered rectangle", or "Custom".
Pad_To_Die_Length	dimension	The value of the "pad to die length" property of a pad, which is additional length added to the pad's net when calculating net length.
Pad_Type	string	One of "Through-hole", "SMD", "Edge connector", or "NPTH, mechanical".
Pin_Name	string	The name of the pad (usually the name of the corresponding pin in the schematic).
Pin_Type	string	<p>The electrical type of the pad (usually taken from the corresponding pin in the schematic). One of "Input", "Output", "Bidirectional", "Tri-state", "Passive", "Free", "Unspecified", "Power input", "Power output", "Open collector", "Open emitter", or "Unconnected".</p> <p>Pins with a no-connection flag on them will have a "+no_connect" suffix added to the pin type string. For example, "passive+no_connect" will match a passive pin with a no-connection flag. To match a pin type whether or not the pin has a no-connection flag, use a wildcard: "passive*" will match passive pins with or without a no-connection flag.</p>
Corner_Radius_Ratio	double	For rounded rectangle pads, the ratio of radius to rectangle size.
Size_X	dimension	The size of the pad in the X-axis.
Size_Y	dimension	The size of the pad in the Y-axis.
Soldermask_Margin_Override	dimension	The solder mask margin override set for the pad.
Solderpaste_Margin_Override	dimension	The solder paste margin override set for the pad.
Solderpaste_Margin_Ratio_Override	dimension	The solder paste margin ratio override set for the pad.
Thermal_Relief_Gap	dimension	The thermal relief gap set for the pad.

Property	Data type	Description
Zone_Connection_Style	string	One of "Inherited", "None", "Thermal reliefs" or "Solid".

## Track and Arc Properties

These properties apply to tracks and arc tracks.

Property	Data type	Description
Origin_X	dimension	The x-coordinate of the start point.
Origin_Y	dimension	The y-coordinate of the start point.
End_X	dimension	The x-coordinate of the end point.
End_Y	dimension	The y-coordinate of the end point.
Width	dimension	The width of the track or arc.

## Via Properties

These properties apply to vias.

Property	Data type	Description
Diameter	dimension	The diameter of the via's pad.
Hole	dimension	The diameter of the via's finished hole.
Layer_Bottom	string	The last layer in the via stackup.
Layer_Top	string	The first layer in the via stackup.
Via_Type	string	One of "Through", "Blind/buried", or "Micro".

## Tuning Pattern Properties

These properties apply to tuning patterns.

Property	Data type	Description
End_X	dimension	The x-coordinate of the end point.
End_Y	dimension	The y-coordinate of the end point.
Min_Amplitude	dimension	The minimum amplitude of the tuning pattern.
Max_Amplitude	dimension	The maximum amplitude of the tuning pattern.
Tuning_Mode	string	One of "Single track", "Differential pair", or "Diff pair skew".
Initial_Side	string	One of "Left", "Right", or "Default".
Min_Spacing	dimension	The minimum spacing of the tuning pattern..
Corner_Radius_%	integer	The corner radius percentage of the tuning pattern.
Target_Length	dimension	The target length for the tuning pattern.
Target_Skew	dimension	The target skew for the tuning pattern.
Override_Custom_Rules	boolean	True if the tuning pattern overrides custom DRC rules.
Single-sided	boolean	True if the tuning pattern is single-sided.
Rounded	boolean	True if the tuning pattern uses rounded meanders.

## Zone and Rule Area Properties

These properties apply to copper and non-copper zones, and rule areas (formerly called keepouts).

Property	Data type	Description
Clearance_Override	dimension	The copper clearance override set for the zone.
Hatch_Gap	dimension	The distance between hatched lines in the zone.
Hatch_Minimum_Hole_Ratio	float	The minimum allowed hatching hole size, expressed as a fraction of the nominal hatching hole size.
Hatch_Orientation	integer	The angle (in degrees) of the hatched lines in the zone.
Hatch_Width	dimension	The width of hatched lines in the zone.
Min_Width	dimension	The minimum allowed width of filled areas in the zone.
Name	string	The user-specified name (blank by default).
Pad_Connections	string	One of "Inherited", "None", "Thermal reliefs", "Solid", or "Thermal Reliefs for PTH".
Priority	integer	The priority level of the zone.
Thermal_Relief_Gap	dimension	The thermal relief gap set for the zone.
Thermal_Relief_Width	dimension	The thermal relief connection width set for the zone.

## Graphic Shape Properties

These properties apply to graphic lines, arcs, circles, rectangles, and polygons.

Property	Data type	Description
Angle	dimension	The angle of an arc.
End_X	dimension	The x-coordinate of the end point.
End_Y	dimension	The y-coordinate of the end point.
Filled	boolean	True if the shape is filled.
Line_Width	dimension	Thickness of the strokes of the shape.
Line_Style	string	One of "Solid", "Dashed", "Dotted", "Dash-Dot", "Dash-Dot-Dot".
Shape	string	One of "Segment", "Rectangle", "Arc", "Circle", "Polygon", or "Bezier".
Start_X	dimension	The x-coordinate of the start point.
Start_Y	dimension	The y-coordinate of the start point.

## Text Properties

These properties apply to text objects (footprint fields, free text labels, etc).

Property	Data type	Description
Bold	boolean	True if the text is bold.
Height	dimension	Height of a character in the font.
Horizontal_Justification	string	Horizontal text justification (alignment): one of "Left", "Center", or "Right".
Italic	boolean	True if the text is italic.
Knockout	boolean	True if the text has the knockout property set.
Mirrored	boolean	True if the text is mirrored.
Name	string	The name of a footprint field. For text objects that are not footprint fields, this is an empty string.
Text	string	The contents of the text object.
Thickness	dimension	Thickness of the stroke of the font.
Width	dimension	Width of a character in the font.
Vertical_Justification	string	Vertical text alignment: one of "Top", "Center", or "Bottom".
Visible	boolean	True if the text object is visible (displayed).

## Expression functions

The following functions can be called on objects in custom rule expressions:

Function	Objects	Description
<code>enclosedByArea('x')</code>	A or B	Returns true if all of the object is inside the named rule area or zone. Note that <code>enclosedByArea()</code> is slower than <code>intersectsArea()</code> . Use <code>intersectsArea()</code> where possible.
<code>existsOnLayer('layer_id')</code>	A or B	Returns true if the object exists on the given board layer. <code>layer_id</code> is a string containing the name of a board layer.
<code>fromTo('x', 'y')</code>	A or B	Returns true if the object exists on the copper path between the given pads. <code>x</code> and <code>y</code> are pad identifiers in the format <code>'RefDes-PadNumber'</code> (for example, <code>'U1-A5'</code> ), or a reference designator alone (for example, <code>'U1'</code> ) to match all pads of a footprint. Wildcards <code>\*</code> and <code>?</code> are supported in both <code>x</code> and <code>y</code> . See <a href="#">From-To signal path matching</a> for details.
<code>getField('x')</code>	A or B	Returns the value of field <code>x</code> in the object. Note that only footprints have fields, so no field will be returned unless the object is a footprint.

Function	Objects	Description
<code>hasComponentClass('x')</code>	A or B	Returns true if the set of component classes assigned to the object contains the named component class <code>x</code> . Note that only footprints have component classes, so this function will only return true if the object is a footprint. In most cases, you will want to check if an object is part of a footprint that has a specific component class. For this query, see the <code>memberOfFootprint()</code> expression function.
<code>hasExactNetclass('x')</code>	A or B	Returns true if the set of net classes assigned to the object exactly matches the named set of net classes <code>x</code> .
<code>hasNetclass('x')</code>	A or B	Returns true if the set of net classes assigned to the object contains the named net class <code>x</code> .
<code>inDiffPair('x')</code>	A or B	Returns true if the object is part of a differential pair and the base name of the pair matches the given argument <code>x</code> . For example, <code>inDiffPair('/USB_')</code> or <code>inDiffPair('/USB')</code> both return true for objects in the nets <code>/USB_P</code> and <code>/USB_N</code> . <code>*</code> and <code>?</code> can be used as wildcards, so <code>inDiffPair('/USB*')</code> matches <code>/USB1_P</code> and <code>/USB1_N</code> as well as <code>/USB2_P</code> and <code>/USB2_N</code> . Note this will always return false if the given net is not a diff pair, meaning that there isn't a matching net of the opposite polarity. So, on a board with a net named <code>/USB_P</code> but no net named <code>/USB_N</code> , this function returns false.
<code>insideArea('x')</code>	A or B	Returns true if any part of the object is inside the named rule area or zone. Rule area and zone names can be set in their respective properties dialogs. If the given area is a filled copper zone, the function tests if the given object is inside any of the filled copper regions of the zone, not if the object is inside the zone's outline.  <b>Deprecated;</b> use <code>intersectsArea()</code> instead.

Function	Objects	Description
<p>insideCourtyard('x')</p> <p>insideFrontCourtyard('x')</p> <p>insideBackCourtyard('x')</p>	A or B	<p>Returns true if the any part of the object is inside the courtyard of the given footprint. The first variant checks both the front or back courtyard and returns true if the object is inside either one; the second and third variants check a courtyard on a specific layer. The named footprint <code>x</code> can be one of the following:</p> <ul style="list-style-type: none"> <li>• A reference designator, possibly containing wildcards <code>*</code> and <code>?</code>. <code>insideCourtyard('R?')</code> will check all footprints with references that contain <code>R</code> followed by a single character, while <code>insideCourtyard('R*')</code> will check all footprints with reference designators starting with <code>R</code>.</li> <li>• A footprint library identifier in <code>&lt;footprint_library&gt;:&lt;footprint_name&gt;</code> format, possibly containing wildcards <code>*</code> and <code>?</code>. <code>insideCourtyard('Resistor_SMD:*')</code> will check all footprints in the <code>Resistor_SMD</code> library.</li> <li>• A component class, in the form <code>\$(Class:ClassName)</code>. The <code>Class</code> keyword is not case-sensitive, but component class names are case-sensitive. The function will return true if the object is inside the courtyard of a footprint with the named component class.</li> </ul> <p><b>Deprecated;</b> use <code>intersectsCourtyard()</code>, <code>intersectsFrontCourtyard()</code>, and <code>intersectsBackCourtyard()</code> instead.</p>
intersectsArea('x')	A or B	<p>Returns true if any part of the object is inside the named rule area or zone. Rule area and zone names can be set in their respective properties dialogs. If the given area is a filled copper zone, the function tests if the given object is inside any of the filled copper regions of the zone, not if the object is inside the zone's outline.</p>

Function	Objects	Description
intersectsCourtyard('x') intersectsFrontCourtyard('x') intersectsBackCourtyard('x')	A or B	Returns true if any part of the object is inside the courtyard of the given footprint. The first variant checks both the front or back courtyard and returns true if the object is inside either one; the second and third variants check a courtyard on a specific layer. The named footprint x can be one of the following: <ul style="list-style-type: none"> <li>• A reference designator, possibly containing wildcards * and ?. intersectsCourtyard('R?') will check all footprints with references that contain R followed by a single character, while intersectsCourtyard('R*') will check all footprints with reference designators starting with R.</li> <li>• A footprint library identifier in &lt;footprint_library&gt;: &lt;footprint_name&gt; format, possibly containing wildcards * and ?. intersectsCourtyard('Resistor_SMD:*') will check all footprints in the Resistor_SMD library.</li> <li>• A component class, in the form \${Class:ClassName}. The Class keyword is not case-sensitive, but component class names are case-sensitive. The function will return true if the object intersects the courtyard of a footprint with the named component class.</li> </ul>
isBlindBuriedVia()	A or B	Returns true if the object is a blind/buried via.
isCoupledDiffPair()	AB	Returns true if the two objects being tested are part of the same differential pair but are opposite polarities. For example, returns true if A is in net /USB+ and B is in net /USB- .
isMicroVia()	A or B	Returns true if the object is a microvia.
isPlated()	A or B	Returns true if the object is a plated hole (in a pad or via).
memberOf('x')	A or B	Returns true if the object is a member of the named group x.  <b>Deprecated;</b> use <code>memberOfGroup()</code> instead.
memberOfGroup('x')	A or B	Returns true if the object is a member of a group named x.

Function	Objects	Description
<code>memberOfFootprint('x')</code>	A or B	<p>Returns true if the object is a member of the given footprint. The named footprint <code>x</code> can be one of the following:</p> <ul style="list-style-type: none"> <li>A reference designator, possibly containing wildcards <code>*</code> and <code>?</code>. <code>memberOfFootprint('R?')</code> will match all footprints with references that contain <code>R</code> followed by a single character, while <code>memberOfFootprint('R*')</code> will match all footprints with reference designators starting with <code>R</code>.</li> <li>A footprint library identifier in <code>&lt;footprint_library&gt;:&lt;footprint_name&gt;</code> format, possibly containing wildcards <code>*</code> and <code>?</code>. <code>memberOfFootprint('Resistor_SMD:*')</code> will match all footprints in the <code>Resistor_SMD</code> library.</li> <li>A component class, in the form <code>#{Class:ClassName}</code>. The <code>Class</code> keyword is not case-sensitive, but component class names are case-sensitive. The function will return true if the object is a member of a footprint with the named component class.</li> </ul>
<code>memberOfSheet('x')</code>	A or B	<p>Returns true if the object is a member of a schematic sheet named <code>x</code>. The sheet path can contain wildcards <code>\*</code> and <code>?</code>. This does not check subsheets: objects in child hierarchical sheets of <code>x</code> are not considered members of <code>x</code>. To check if an object is in a sheet or any of that sheet's child sheets, use <code>memberOfSheetOrChildren()</code>.</p>
<code>memberOfSheetOrChildren('x')</code>	A or B	<p>Returns true if the object is a member of a schematic sheet named <code>x</code> or any of its child hierarchical sheets. The sheet path can contain wildcards <code>\*</code> and <code>?</code>.</p>

## From-To signal path matching

In high-speed PCB design, constraining an entire net is often too coarse. A single net may connect many pads, but only a specific segment of that net — the path between two particular pads — carries the timing-critical signal. For example, a DDR data net may run from a memory controller pad to a DRAM pad, but the same net also fans out to termination resistors or test points. Applying a length constraint to the whole net would include copper that is irrelevant to the high-speed signaling path.

From-To paths solve this problem. A From-To path is a designer-defined signal path between two specific pads on a board. By identifying the copper objects (tracks, vias, and pads) that form the unique electrical path between a source pad and a destination pad, From-To paths allow custom DRC rules to target exactly the portion of a net that matters for signal integrity.

This concept is analogous to Altium Designer's xSignals feature, where designers define specific signal paths to scope length-matching and other high-speed design rules to individual point-to-point connections rather

## How From-To paths work

When a custom DRC rule uses the `fromTo()` function, KiCad evaluates From-To paths for the board. The process works as follows:

- 1. Endpoint discovery:** KiCad scans every footprint on the board and builds a list of pad endpoints. Each endpoint is identified by a name in the format `RefDes-PadNumber` (for example, `U1-A5` or `R3-1`). Each pad is also identified by just the parent footprint's reference designator (for example, `U1`), allowing rules to match all pads of a footprint.
- 2. Pad matching:** The `from` and `to` arguments are matched against endpoint names using wildcard comparison. The wildcards `\*` (match any sequence of characters) and `?` (match a single character) are supported. Matching is case-insensitive.
- 3. Path tracing:** For each pair of matching endpoints that share the same net, KiCad traces the copper connectivity to find the path between them. The path includes all tracks, vias, and pads that lie on the electrical route between the two endpoints.
- 4. Uniqueness check:** KiCad determines whether the path between the two pads is *unique*—that is, whether there is exactly one route between the pads. If there are multiple paths (for example, because of a ground plane or copper pour connecting the pads through more than one route), the path is still found but is marked as non-unique.

### NOTE

If the `from` pattern matches a pad that has more than one connected pad matching the `to` pattern *on the same net*, the path cannot be unambiguously determined and will not be created. Make your `from` and `to` patterns specific enough to identify exactly one pad at each end.

## Pad name format

Pad endpoints used in `fromTo()` follow a specific naming convention:

Format	Description
<code>RefDes-PadNumber</code>	Matches a specific pad on a specific footprint. For example, <code>'U1-A5'</code> matches pad A5 on U1.
<code>RefDes</code>	Matches <i>all</i> pads on the given footprint. For example, <code>'U1'</code> matches every pad on U1. This is useful when combined with a specific pad at the other end.
Wildcards	Both <code>*</code> and <code>?</code> wildcards are supported. For example, <code>'U1-A\*'</code> matches all pads on U1 whose pad number starts with A, and <code>'R?-1'</code> matches pad 1 on all single-character reference designators starting with R (such as R1, R2, etc.).

## Using `fromTo()` in custom DRC rules

The `fromTo()` function is used in the `condition` clause of a custom DRC rule. It takes two string arguments: the `from` pad identifier and the `to` pad identifier. The function returns true if the object being tested (either A or B) lies on the copper path between the specified pads.

The basic syntax is:

```
(rule <name>
  (condition "A.fromTo('<from_pad>', '<to_pad>')")
  (constraint <constraint_type> <arguments>))
```

The `fromTo()` function is most commonly combined with the following constraint types:

Constraint	Use with <code>fromTo()</code>
<code>length</code>	Constrain the total routed length of a specific signal path, rather than the entire net.
<code>skew</code>	Match lengths across a group of signal paths (for example, a DDR data bus) by constraining the skew between paths.
<code>track_width</code>	Enforce specific trace widths on high-speed signal paths.
<code>clearance</code>	Apply tighter or looser clearance rules to specific signal paths.
<code>diff_pair_gap</code>	Control differential pair spacing on specific segments.

## From-To path examples

### Length-constrain a specific signal path

To constrain the routed length of a specific signal path between a memory controller and a DRAM chip, rather than constraining the entire net:

```
# Constrain the DDR clock path from the controller to the DRAM
(rule "DDR_CLK length"
  (condition "A.fromTo('U1-C5', 'U2-D3')")
  (constraint length (min 20mm) (max 25mm)))
```

### Match lengths across a DDR data bus

When designing a DDR interface, all data signals must be length-matched to each other. Using `fromTo()`, you can target only the relevant path segments and combine them with a `skew` constraint:

```

# Match data signal lengths from memory controller to DRAM
# Each data line has its own fromTo path; the skew constraint
# checks that all matching paths are within tolerance
(rule "DDR_DQ0 length"
  (condition "A.fromTo('U1-A1', 'U2-B1')")
  (constraint length (max 50mm)))

(rule "DDR_DQ1 length"
  (condition "A.fromTo('U1-A2', 'U2-B2')")
  (constraint length (max 50mm)))

(rule "DDR_DQ2 length"
  (condition "A.fromTo('U1-A3', 'U2-B3')")
  (constraint length (max 50mm)))

(rule "DDR_DQ3 length"
  (condition "A.fromTo('U1-A4', 'U2-B4')")
  (constraint length (max 50mm)))

# Constrain skew across all DQ paths
(rule "DDR_DQ skew"
  (condition "A.fromTo('U1-A*', 'U2-B*')")
  (constraint skew (max 0.2mm)))

```

## USB signal path constraints

For a USB interface, you might constrain the differential pair paths from a connector to a controller, including both length and track width:

```

# USB data path track width
(rule "USB track width"
  (condition "A.fromTo('J1-D*', 'U1-H*')")
  (constraint track_width (min 0.09mm) (opt 0.09mm) (max 0.09mm)))

# USB data path length constraint
(rule "USB path length"
  (condition "A.fromTo('J1-D*', 'U1-H*')")
  (constraint length (max 80mm)))

```

## PCIe lane length matching

For PCIe, each transmit and receive lane must be length-matched. Using wildcards in `fromTo()`, you can write rules that cover multiple lanes:

```
# PCIe TX lane length matching from connector to controller
(rule "PCIe TX length"
  (condition "A.fromTo('J1-TX*', 'U1-TX*')")
  (constraint length (min 40mm) (max 60mm)))

# PCIe RX lane length matching from connector to controller
(rule "PCIe RX length"
  (condition "A.fromTo('J1-RX*', 'U1-RX*')")
  (constraint length (min 40mm) (max 60mm)))
```

## Series termination resistor paths

A common high-speed topology places a series termination resistor between a driver and a receiver. In this case, you need separate `fromTo()` rules for each segment of the path:

```
# Segment from driver to series resistor (keep short)
(rule "CLK driver to resistor"
  (condition "A.fromTo('U1-C5', 'R1-1')")
  (constraint length (max 10mm)))

# Segment from series resistor to receiver (length-controlled)
(rule "CLK resistor to receiver"
  (condition "A.fromTo('R1-2', 'U2-D3')")
  (constraint length (min 20mm) (max 30mm)))
```

### TIP

When a net passes through a series component such as a termination resistor, the net changes at the component pads. Define separate `fromTo()` paths for each segment (driver to resistor and resistor to receiver) to control each portion independently.

## Clearance between high-speed paths and other nets

You can combine `fromTo()` with clearance constraints to create isolation rules for sensitive signal paths:

```
# Extra clearance around high-speed clock path
(rule "CLK clearance"
  (condition "A.fromTo('U1-C5', 'U2-D3')")
  (constraint clearance (min 0.3mm)))
```

## Custom design rule examples

## Basic examples

```
(rule RF_width
  (layer outer)
  (condition "A.hasNetclass('RF')")
  (constraint track_width (min 0.35mm) (max 0.35mm)))

(rule "BGA neckdown"
  (constraint track_width (min 0.2mm) (opt 0.25mm))
  (constraint clearance (min 0.05mm) (opt 0.08mm))
  (condition "A.intersectsCourtyard('U3')"))

# Specify an optimal gap for a particular differential pair
(rule "Clock gap"
  (condition "A.inDiffPair('/CLK')")
  (constraint diff_pair_gap (opt 0.8mm)))

# Specify a larger clearance between differential pairs and anything else
(rule "Differential pair clearance"
  (condition "A.inDiffPair('*') && !AB.isCoupledDiffPair()")
  (constraint clearance (min 1.5mm)))

(rule "copper keepout"
  (constraint disallow track via zone)
  (condition "A.intersectsArea('zone3')"))

(rule "minimum creepage distance for high voltage nets"
  (condition "A.hasNetclass('HV')")
  (constraint creepage (min 5mm)))
```

## Various clearances

```
(rule "Clearance between Pads of Different Nets"
  (constraint clearance (min 3.0mm))
  (condition "A.Type == 'Pad' && B.Type == 'Pad' && A.Net != B.Net"))

(rule "Pad to Track Clearance"
  (constraint clearance (min 0.2mm))
  (condition "A.Type == 'Pad' && B.Type == 'Track'"))

# Enforce a clearance around pads (and other copper objects) in a specific footprint
(rule "Pad clearance in R1"
  (constraint clearance (min 1mm))
  (condition "A.memberOfFootprint('TP1')"))

# Enforce a mechanical clearance between components and board edge
(rule front_mechanical_board_edge_clearance
  (layer "F.Courtyard")
  (constraint physical_clearance (min 3mm))
  (condition "B.Layer == 'Edge.Cuts'"))

# Prevent copper pours under capacitors
(rule "No copper pours under capacitors"
  (constraint physical_clearance (min 0.1mm))
  (condition "A.Type == 'Zone' && B.Reference == 'C*'))
)

# This assumes that there is a cutout with 1mm thick lines
(rule "Clearance to cutout"
  (constraint edge_clearance (min 0.8mm))
  (condition "A.Layer=='Edge.Cuts' && A.Line_Width == 1.0mm"))

# prevent silk over tented vias
(rule silk_over_via
  (constraint silk_clearance (min 0.2mm))
  (condition "A.Type == '*Text' && B.Type == 'Via'"))

(rule "Allow connector silk to intersect board edge"
  (constraint silk_clearance)
  (severity ignore)
  (condition "A.memberOfFootprint('J*') && B.Layer=='Edge.Cuts'"))

(rule "Distance between Vias of Different Nets"
  (constraint hole_to_hole (min 0.254mm))
  (condition "A.Type == 'Via' && B.Type == 'Via' && A.Net != B.Net"))

(rule "Via Hole to Track Clearance"
  (constraint hole_clearance (min 0.254mm))
  (condition "A.Type == 'Via' && B.Type == 'Track'"))

(rule "Distance between test points"
  (constraint courtyard_clearance (min 1.5mm))
  (condition "A.Reference == 'TP*' && B.Reference == 'TP*'))
```