Schematic Editor

The KiCad Team

Table of Contents

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.

Hak Cipta

Kontributor

Jean-Pierre Charras, Fabrizio Tappero, Wayne Stambaugh, Graham Keeth

Umpan balik

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 Schematic Editor

The KiCad Schematic Editor is a schematic capture 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 Schematic Editor is an integrated application where all functions of schematic drawing, PCB footprint selection, library management, and data transfer to and from the PCB design software are carried out within the editor itself.

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

The Schematic Editor includes a symbol library editor, which can create and edit symbols and manage libraries. It also integrates the following additional but essential functions needed for modern schematic capture software:

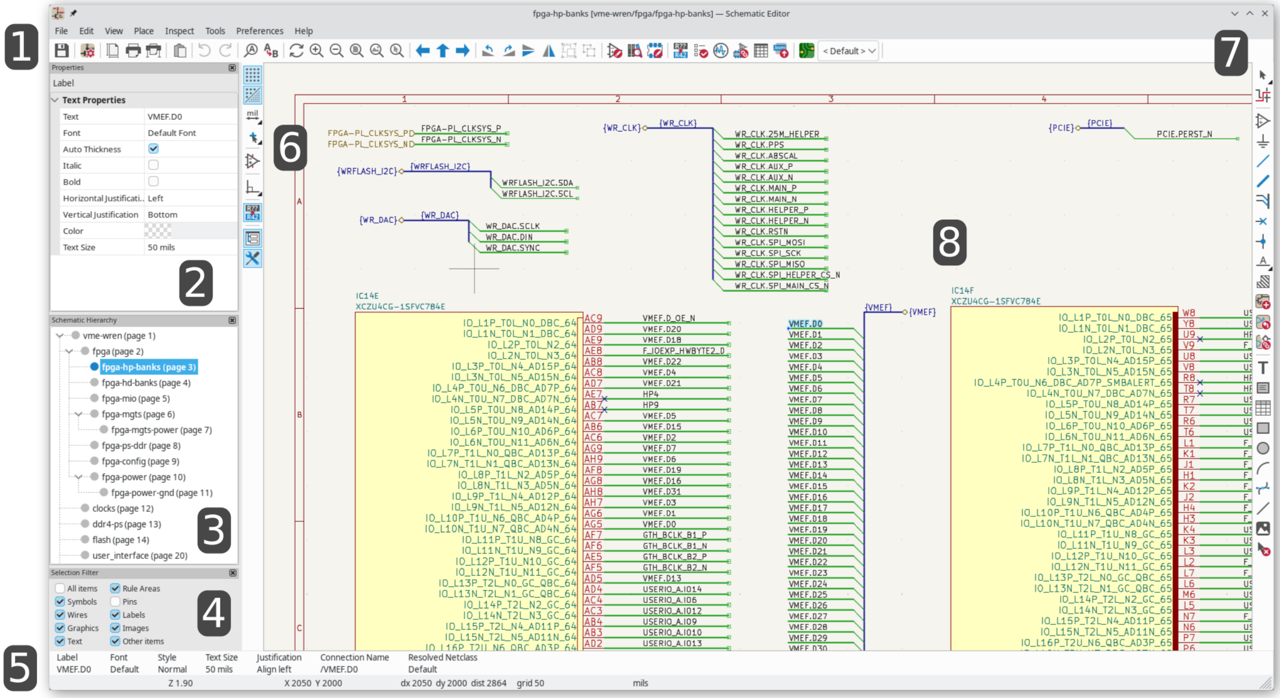
- Electrical rules check (ERC) for automatic detection of incorrect and missing connections
- Circuit simulation using ngspice
- Ekspor berkas-berkas plot dalam berbagai format (Postscript, PDF, HPGL, dan SVG)
- Pembuatan daftar komponen (*Bill of Materials*), melalui skrip Python atau XSLT, yang mampu menangani berbagai format fleksibel.

The Schematic Editor supports multi-sheet schematics in several ways:

- Flat hierarchies (schematic sheets are not explicitly connected in a master diagram).
- Simple hierarchies (each schematic sheet is used only once).
- Complex hierarchies (some schematic sheets are used multiple times).

Hierarchical schematics are described in detail later in the manual.

The Schematic Editor User Interface



fpga-hp-banks [vme-wren/fpga/fpga-hp-banks] — Schematic Editor

Properties

Label

Text Properties

Text	VMEF.D0
Font	Default Font
Auto Thickness	☑
Italic	□
Bold	□
Horizontal Justificati...	Left
Vertical Justification	Bottom
Color	
Text Size	50 mils

Schematic Hierarchy

- vme-wren (page 1)
  - fpga (page 2)
    - fpga-hp-banks (page 3)
    - fpga-hd-banks (page 4)
    - fpga-mio (page 5)
    - fpga-mgts (page 6)
      - fpga-mgts-power (page 7)
    - fpga-ps-ddr (page 8)
    - fpga-config (page 9)
    - fpga-power (page 10)
      - fpga-power-gnd (page 11)
  - clocks (page 12)
  - ddr4-ps (page 13)
  - flash (page 14)
  - user_interface (page 20)

Selection Filter

☐ All items ☑ Rule Areas
☑ Symbols ☐ Pins
☑ Wires ☑ Labels
☑ Graphics ☑ Images
☑ Text ☑ Other items

Label	Font	Style	Text Size	Justification	Connection Name	Resolved Netclass
VMEF.D0	Default	Normal	50 mils	Align left	/VMEF.D0	Default

Z 1.90 X 2050 Y 2000 dx 2050 dy 2000 dist 2864 grid 50 mils

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

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

Navigating the editing canvas

The editing canvas displays the schematic being designed. You can pan and zoom to different parts of the schematic and open any schematic sheet in the design.

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 every item in the schematic (not including the drawing sheet). For instance, if there are items placed outside of the drawing sheet, they will be visible after zooming to objects.
- 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.

Tombol *Hotkey*

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.

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.
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.

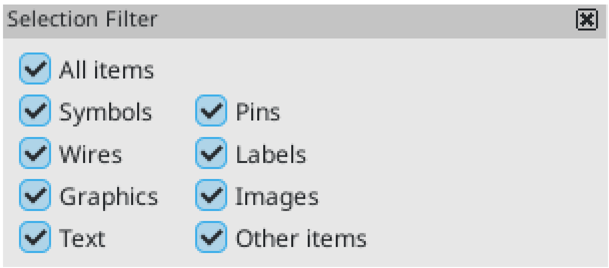
The following modifier keys apply when dragging to perform a rectangular selection:

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 left corner of the Schematic 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 busy schematic.

The **All items** checkbox is a shortcut to turn the other items on and off. 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.



Selection Filter ☒

☑ All items

☑ Symbols ☑ Pins

☑ Wires ☑ Labels

☑ Graphics ☑ Images

☑ Text ☑ Other items

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.

Left toolbar display controls

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

⠿	Turns grid display on/off. **Note:** by default, hiding the grid does not disable grid snapping. This behavior can be changed in the Display Options section of Preferences.
▨	Turns item-specific grid overrides on/off.
in mil mm	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).
▷	Turns invisible pin display on/off.
∠ ∟ ∠	Switches the line mode between free angle, 90 degree mode, and 45 degree mode for placement of new wires, buses, and graphical shapes. 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 `Shift` + `Space`.
R?? R42	Turns automatic symbol annotation on/off. When on, symbols will have their reference designators automatically set to the lowest available reference when they are added to the schematic.
☰	Opens and closes the docked Hierarchy Navigator panel.
🛠	Opens and closes the docked Properties Manager panel.

Membuat dan Mengedit Skematik

Pengenalan

A schematic designed with KiCad is more than a simple graphic representation of an electronic device. It is normally the entry point of a development chain that allows for:

- Melakukan validasi berdasarkan suatu aturan tertentu (Electrical Rules Check) untuk mendeteksi kesalahan dan kelalaian.
- Automatically generating a bill of materials.
- Pembuatan *netlist* untuk perangkat lunak simulasi seperti SPICE.
- Defining a circuit for transferring to PCB layout.

A schematic mainly consists of symbols, wires, labels, junctions, buses and power symbols. For clarity in the schematic, you can place purely graphical elements like bus entries, comments, and polylines.

Symbols are added to the schematic from symbol libraries. After the schematic is made, the set of connections and footprints is imported into the PCB editor for designing a board.

Schematics can be contained in a single sheet or split among multiple sheets. In KiCad, multi-sheet schematics are organized hierarchically, with a root sheet and sub-sheet(s). Each sheet is its own `.kicad_sch` file and is itself a complete KiCad schematic. Working with hierarchical schematics is described in the Hierarchical Schematics chapter.

Schematic editing operations

Schematic editing 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 button or click and drag the mouse. The palette will show the most recently selected item 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 Schematic Editor Preferences.

	Selection tool (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.

	Highlight a net by marking its wires and net labels with a different color. If the PCB Editor is also open then copper corresponding to the selected net will be highlighted as well. Net highlighting can be cleared by clicking with the highlight tool in an empty space, or by using the Clear Net Highlighting hotkey (~).
	Display the symbol selector dialog to place a new symbol.
	Display the power symbol selector dialog to place a new power symbol.
	Draw a wire.
	Draw a bus.
	Draw wire-to-bus entry points. These elements are only graphical and do not create a connection, thus they should not be used to connect wires together.
	Place a "no-connection" flag. These flags should be placed on symbol pins which are meant to be left unconnected. "No-connection" flags indicate to the Electrical Rule Checker that the pin is intentionally unconnected and not an error. They also affect schematic connectivity for stacked symbol pins.
	Place a junction. This connects two crossing wires or a wire and a pin, which can sometimes be ambiguous without a junction (i.e. if a wire end or a pin is not directly connected to another wire end).
	Place a local label, directive label, global label, or hierarchical label. Local labels connect items located *in the same sheet*. Local labels with the same name are connected as long as they are in the same sheet. Global labels connect items located anywhere in the schematic. All global labels with the same name are connected, even when located on different sheets. Hierarchical labels are used to create a connection between a subsheet and the sheet's parent sheet. See the Hierarchical Schematics section for more information about hierarchical labels, sheets, and pins. Directive labels are used to assign net classes to individual nets. They can also be used to assign net classes and component classes to nets and symbols, respectively, inside rule areas. Only one label type is visible in the toolbar at a time; you can expand the palette to choose another tool by clicking and holding/dragging.
	Place a rule area.
	Place a hierarchical subsheet. You must specify the file name for this subsheet.
	Place a hierarchical sheet pin on a sheet corresponding to a hierarchical label that has been added in the target sheet.
	Sync hierarchical sheet pins and hierarchical labels. This displays a list of all the hierarchical labels in each subsheet and lets you manage the corresponding hierarchical sheet pins.

▦	Place a table.
■	Draw a rectangle.
●	Draw a circle.
◜	Draw an arc.
~	Draw a bezier curve.
/	Draw graphic lines. **Note:** Lines are graphical objects and are not the same as wires placed with the Wire tool. They do not connect anything.
🖼	Place a bitmap image.
✖	Delete clicked items.

Grids and snapping

Schematic elements such as symbols, wires, text, and graphic lines are snapped to the grid when moving, dragging, and drawing them. Additionally, the wire and label tools snap to other connected items such as pins, wires, and labels even when grid snapping is disabled.

Both grid and connected 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**
`Ctrl`	Disable grid snapping.
`Shift`	Disable connected object snapping.

The default grid size is 50 mil (0.050") or 1.27 millimeters. This is the recommended grid for placing symbols and wires in a schematic and for placing pins when designing a symbol in the Symbol Editor. Smaller grids can also be used, but this is intended only for text and symbol graphics, and not recommended for placing pins and wires.

NOTE

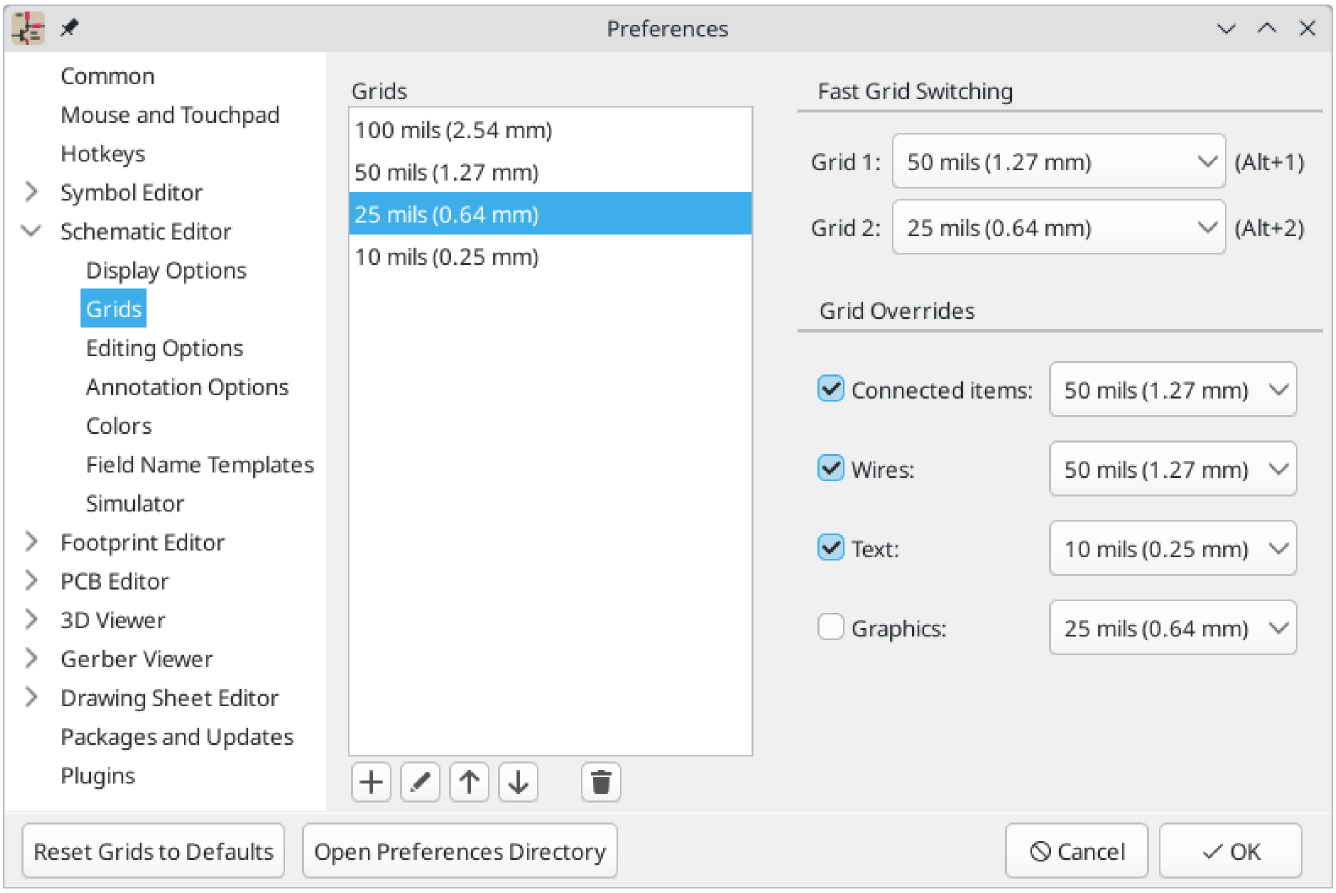
Wires connect with other wires or pins only if their ends coincide **exactly**. Therefore it is very important to keep symbol pins and wires aligned to the grid. It is recommended to always use a 50 mil grid when placing symbols and drawing wires because the KiCad standard symbol library and all libraries that follow its style also use a 50 mil grid. **Using a grid size other than 50 mil will result in schematics without proper connectivity!**

NOTE

Symbols, wires, and other elements that are not aligned to the grid can be snapped back to the grid by selecting them, right clicking, and clicking **Align Elements to Grid**.

You can adjust the grid size 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...**.



Grids

- 100 mils (2.54 mm)
- 50 mils (1.27 mm)
- 25 mils (0.64 mm)
- 10 mils (0.25 mm)

Fast Grid Switching

Grid 1: 50 mils (1.27 mm) (Alt+1)

Grid 2: 25 mils (0.64 mm) (Alt+2)

Grid Overrides

☑ Connected items: 50 mils (1.27 mm)

☑ Wires: 50 mils (1.27 mm)

☑ Text: 10 mils (0.25 mm)

□ Graphics: 25 mils (0.64 mm)

In this dialog you can select an active grid from the list of grids, reorder the list of grids (↑ / ↓), and add (+), remove (🗑), or edit (✎) 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 Alt + 1 and Alt + 2.

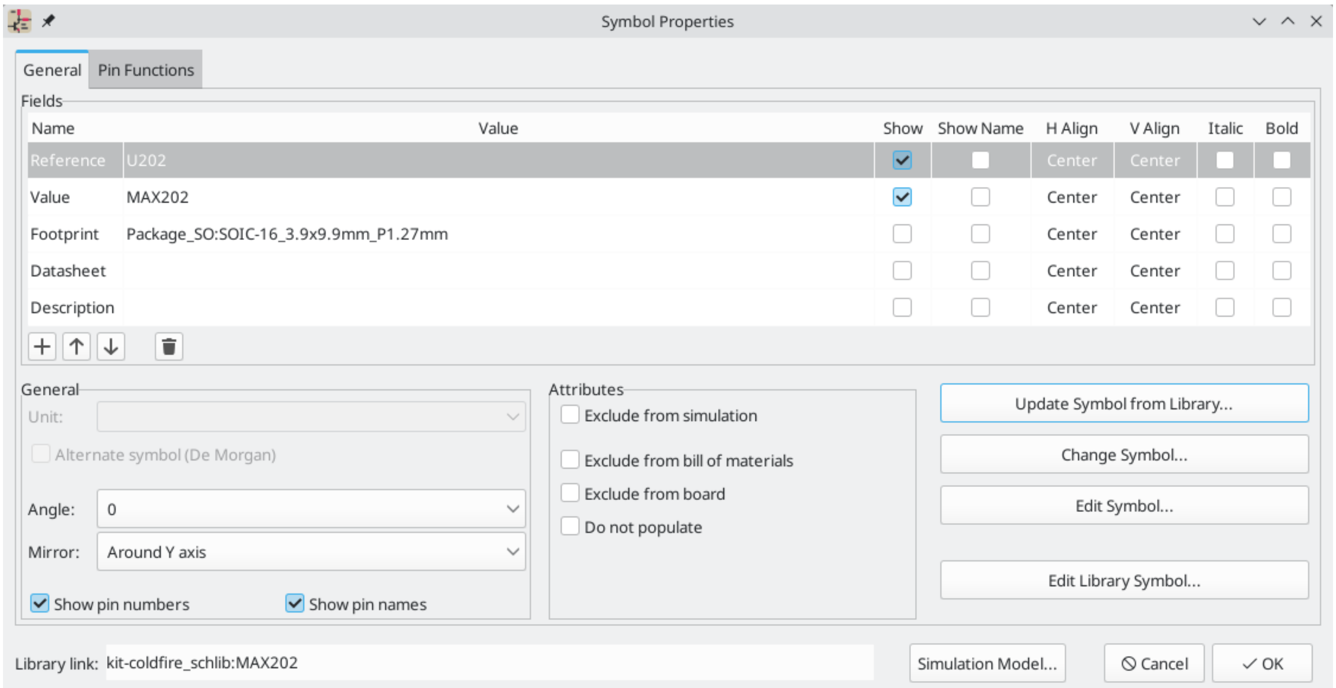
Finally, you can configure grid overrides for different types of objects. Grid overrides let you set particular grid sizes for different types of objects which will be used instead of the default grid when working with those objects. For example, you can set a 50 mil grid for wires and connected items while using smaller grids to finely position text and graphics. Grid overrides can be individually enabled and disabled in this dialog, or globally enabled and disabled using the ▨ button on the left toolbar (Ctrl + Shift + G).

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 symbols, 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.



Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Reference	U202	☑	□	Center	Center	□	□
Value	MAX202	☑	□	Center	Center	□	□
Footprint	Package_SO:SOIC-16_3.9x9.9mm_P1.27mm	□	□	Center	Center	□	□
Datasheet		□	□	Center	Center	□	□
Description		□	□	Center	Center	□	□

General

Unit:

□ Alternate symbol (De Morgan)

Angle: 0

Mirror: Around Y axis

☑ Show pin numbers ☑ Show pin names

Attributes

□ Exclude from simulation

□ Exclude from bill of materials

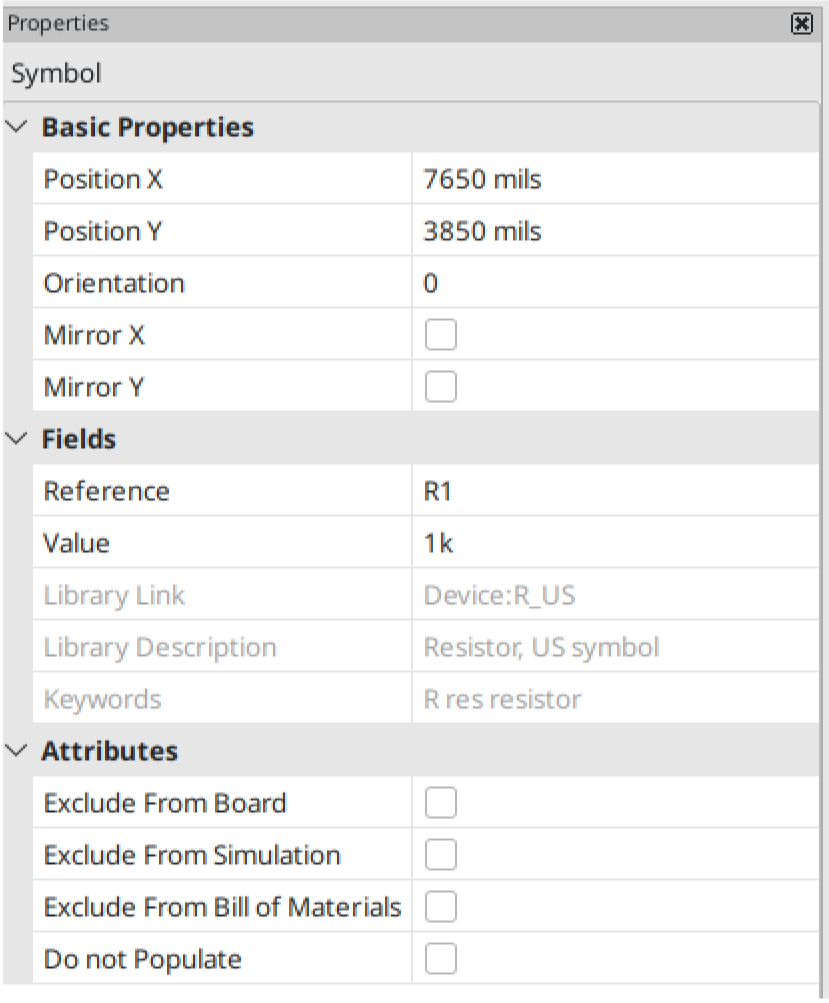
□ Exclude from board

□ Do not populate

Library link: kit-coldfire_schlib:MAX202

You can only use the properties dialog to edit one item at a time. To edit multiple items, use the Properties Manager, described below. There are also other tools that can be used to edit specific types of objects in bulk, such as the Edit Text and Graphics tool for editing visual properties of text, symbol fields, labels, and graphic shapes, or the Symbol Fields Table for editing symbol fields in bulk.

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

Symbol

Basic Properties	
Position X	7650 mils
Position Y	3850 mils
Orientation	0
Mirror X	☐
Mirror Y	☐
Fields	
Reference	R1
Value	1k
Library Link	Device:R_US
Library Description	Resistor, US symbol
Keywords	R res resistor
Attributes	
Exclude From Board	☐
Exclude From Simulation	☐
Exclude From Bill of Materials	☐
Do not Populate	☐

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.

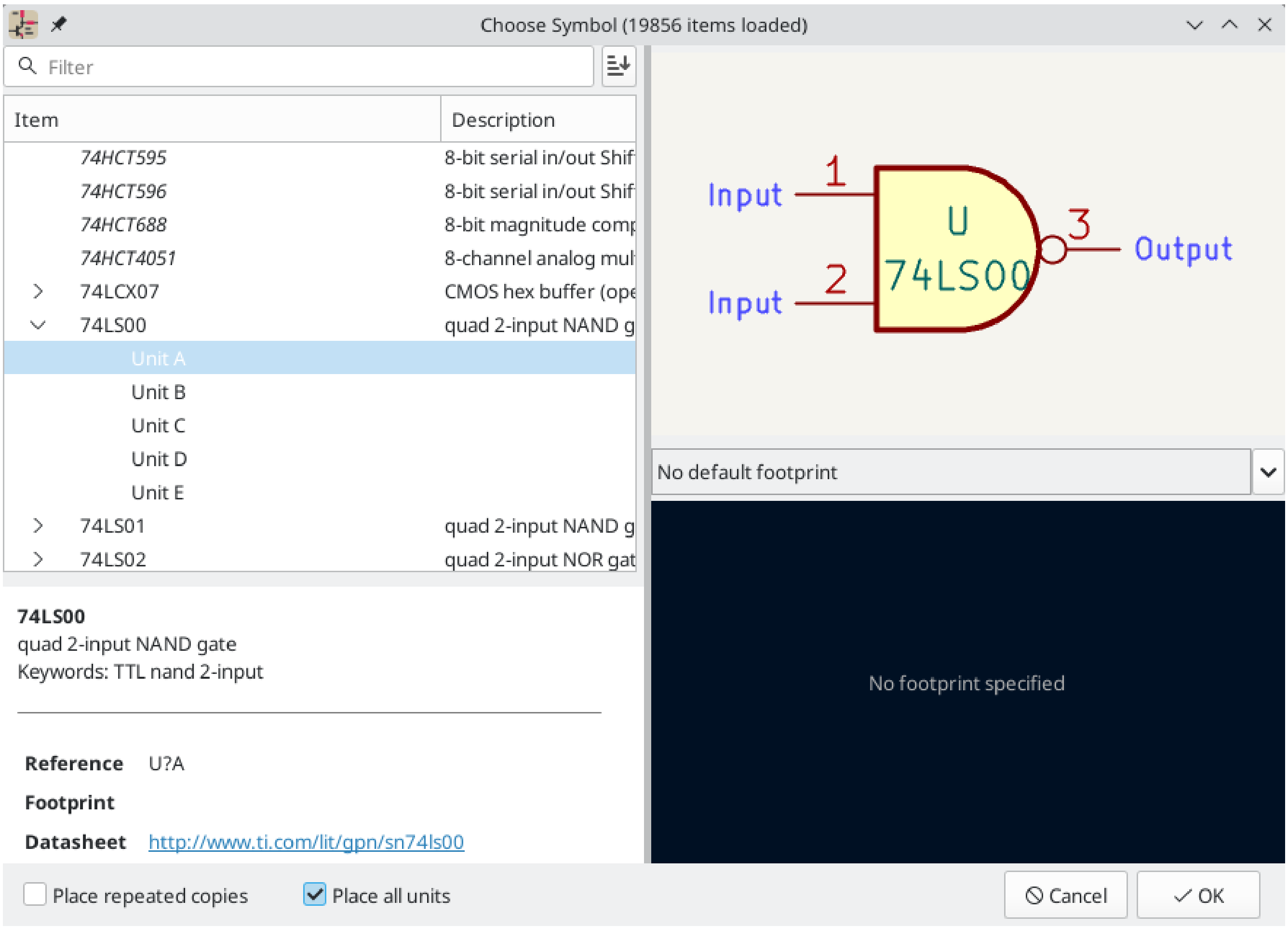
In properties dialogs and many other dialogs, any field that contains a numeric value can also accept a basic math expression that results in a numeric value.

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`.

Working with symbols

Placing symbols

To place a symbol in your schematic, use the button or the A hotkey. The Choose Symbols dialog appears and lets you select a symbol to add. Symbols are grouped by symbol library.



Item	Description
74HCT595	8-bit serial in/out Shif
74HCT596	8-bit serial in/out Shif
74HCT688	8-bit magnitude comp
74HCT4051	8-channel analog mul
74LCX07	CMOS hex buffer (ope
74LS00	quad 2-input NAND g
Unit A	
Unit B	
Unit C	
Unit D	
Unit E	
74LS01	quad 2-input NAND g
74LS02	quad 2-input NOR gat

74LS00
quad 2-input NAND gate
Keywords: TTL nand 2-input

Reference U?A
Footprint
Datasheet http://www.ti.com/lit/gpn/sn74ls00

□ Place repeated copies ☑ Place all units

By default, only the symbol/library name and description columns are shown. Additional columns can be added by right-clicking the column header and selecting **Select Columns**.

The Choose Symbol dialog filters symbols by name, keywords, description, and all additional symbol fields according to what you type into the search field. You can choose to sort search results alphabetically or by best match by clicking on the ⇣ button.

Some advanced filters are available:

- **Wildcards:** `*` matches any number of any characters, including none, and `?` matches any single character.
- **Key-value pairs:** if a library part's description or keywords contain a tag of the format "Key:123", you can match relative to that by typing "Key>123" (greater than), "Key<123" (less than), etc. Numbers may include one of the following case-insensitive suffixes:

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

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

-

Regular expressions: if you're familiar with regular expressions, these can be used too. The regular expression flavor used is the wxWidgets Advanced Regular Expression style, which is similar to Perl regular expressions.

If the symbol specifies a default footprint, this footprint will be previewed in the lower right. If the symbol includes footprint filters, alternate footprints that satisfy the footprint filters can be selected in the footprint dropdown menu at right.

After selecting a symbol to place, the symbol will be attached to the cursor. Left clicking the desired location in the schematic places the symbol into the schematic. Before placing the symbol in the schematic, you can rotate it, mirror it, and edit its fields, by either using the hotkeys or the right-click context menu. These actions can also be performed after placement.

If the **Place repeated copies** option is checked, after placing a symbol KiCad will start placing another copy of the symbol. This process continues until the user presses `Esc`.

For symbols with multiple units, if the **Place all units** option is checked, after placing the symbol KiCad will start placing the next unit in the symbol. This continues until the last unit has been placed or the user presses `Esc`.

Placing power symbols

A power symbol is a symbol representing a connection to a power net. The symbols are grouped in the `power` library, so they can be placed using the symbol chooser. However, as power placements are frequent, the ⏚ tool is available. This tool is similar, except that the search is done directly in the `power` library and any other library that contains power symbols.

Moving symbols

Symbols can be moved using the Move (`M`) or Drag (`G`) tools. These tools act on the selected symbol, or if no symbol is selected they act on the symbol under the cursor.

The **Move** tool moves the symbol itself without maintaining wired connections to the symbol pins.

The **Drag** tool moves the symbol without breaking wired connections to its pins, and therefore moves the connected wires as well.

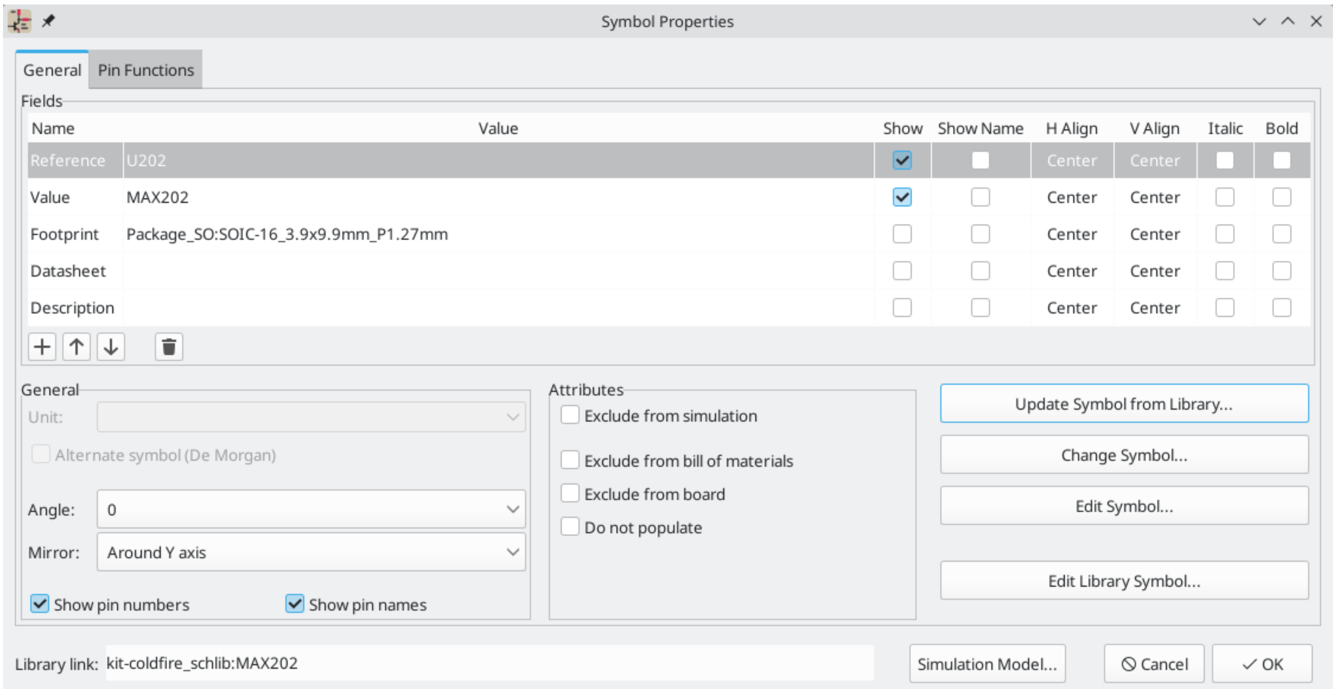
You can also Drag symbols by clicking and dragging them with the mouse, depending on the **Left button drag gesture** setting in the **Mouse and Touchpad** section of Preferences.

Symbols can also be rotated (`R`) or mirrored in the X (`X`) or Y (`Y`) directions.

Editing symbol properties

Symbols in the schematic can be individually edited, both in terms of their properties (fields, attributes, etc.) and in terms of their pins and graphics. Editing a symbol in the schematic only affects that particular instance of the symbol; it does not affect any other copies of that symbol in the schematic, and it does not affect the library symbol.

To edit the properties of a symbol in the schematic, open its properties dialog (`E`). You can also double-click the symbol.



Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Reference	U202	☑	□	Center	Center	□	□
Value	MAX202	☑	□	Center	Center	□	□
Footprint	Package_SO:SOIC-16_3.9x9.9mm_P1.27mm	□	□	Center	Center	□	□
Datasheet		□	□	Center	Center	□	□
Description		□	□	Center	Center	□	□

General

Unit:

□ Alternate symbol (De Morgan)

Angle: 0

Mirror: Around Y axis

☑ Show pin numbers ☑ Show pin names

Attributes

□ Exclude from simulation

□ Exclude from bill of materials

□ Exclude from board

□ Do not populate

Library link: kit-coldfire_schlib:MAX202

The Symbol Properties window displays all the fields of a symbol in a table. New fields can be added, and existing fields can be deleted, edited, reordered, moved, or resized. Fields can be arbitrarily named, but names beginning with `ki_`, e.g. `ki_description`, are reserved by KiCad and should not be used for user fields. All symbol fields will be added to the symbol's corresponding footprint when the PCB is updated from the schematic.

Each field's name and value can be visible or hidden, and there are several formatting options: horizontal and vertical alignment, orientation, position, font, text color, text size, and bold/italic emphasis. Field autoplacement can also be enabled on a per-field basis. The displayed position is always indicated for a normally displayed symbol (no rotation or mirroring) and is relative to the anchor point of the symbol.

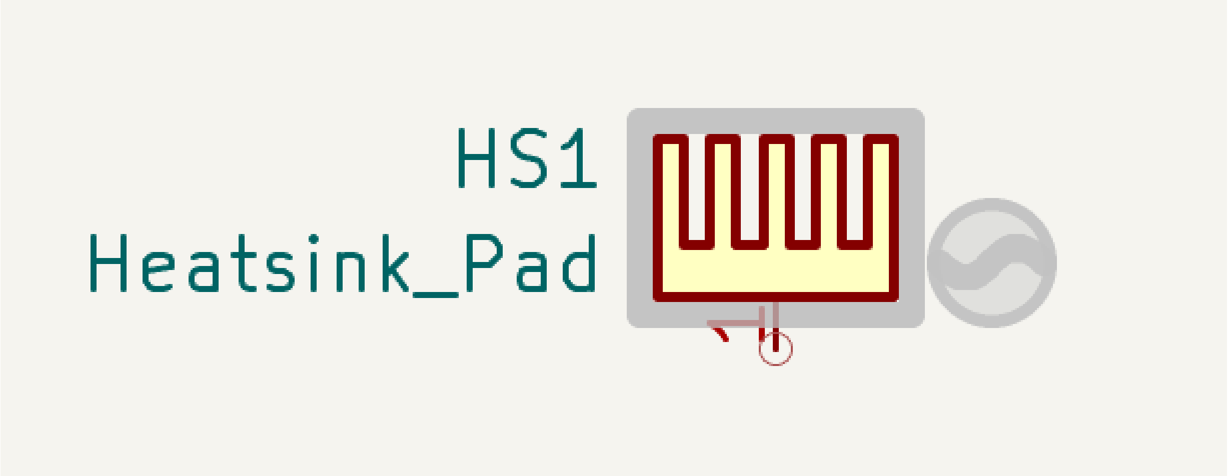
> **NOTE** Formatting options for symbol fields can be shown or hidden by right-clicking on the header row of the symbol field table and enabling or disabling the desired columns. Not all columns are shown by default.

Several fields have special behavior:

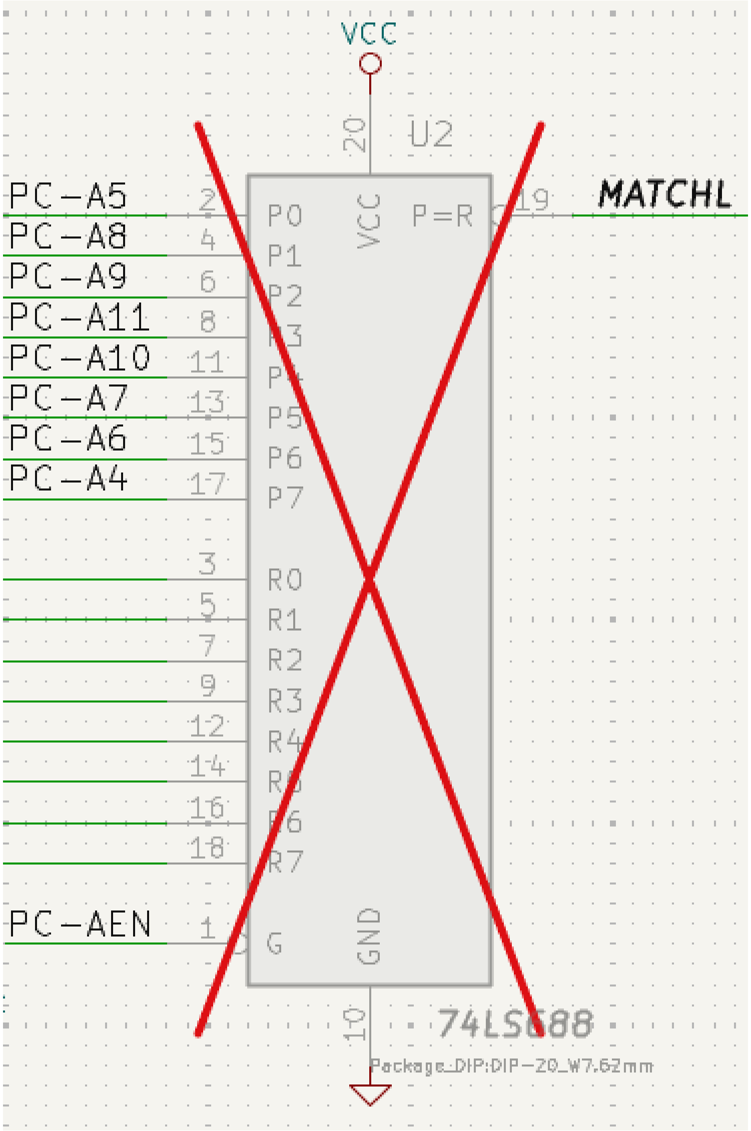
- The **Footprint** field defines which footprint will correspond to the symbol in the board design. When the footprint field is selected, you can click the |||\ button to open the footprint chooser to assign a footprint to the symbol. See the Assigning Footprints section for other ways to assign footprints.
- The **Datasheet** field can contain the manufacturer's datasheet for the symbol. You can right click a symbol in the editing canvas and choose **Show Datasheet** (D) to open the datasheet listed in the symbol. A symbol's datasheet can be a local file or a file at a remote URL, like the manufacturer's website. You can choose a local file using a file browser by selecting the datasheet field in the symbol's properties, then clicking the 🌐 button. If you enable the **Embed File** checkbox in the file browser, the datasheet will be embedded in the schematic instead of being referenced as an an external file. This means the datasheet will be available on any computer. For more information, see the embedded files documentation.

Symbols have several attributes that affect how the symbols are treated by other parts of KiCad.

Exclude from simulation prevents the symbol from being included in SPICE simulations. Symbols that are excluded from simulation are drawn with a grey outline around them and a simulation waveform icon to their bottom right, as shown below. The color of the outline and icon is configurable by editing the "Excluded-from-simulation Markers" color in the selected colorscheme. The visual marker (the outline and the icon) can be disabled completely by disabling **View** → **Mark items which are excluded from simulation**.



- **Exclude from bill of materials** prevents the component from being included in BOM exports.
- **Exclude from board** means that the symbol is schematic-only, and a corresponding footprint will not be added to the PCB.
- **Do not populate** means that the component should not be attached to the PCB, although a corresponding footprint should still be added to the board. DNP symbols appear desaturated and with a red "X" over them in the schematic, as shown below. The color of the "X" is configurable by editing the "DNP Markers" color in the selected colorscheme.



To edit the symbols's form, i.e. its pins and graphics, you need to use the symbol editor. There are two buttons for opening a symbol in the editor, depending on whether you want to edit a single copy of a symbol in the schematic or a symbol's source copy in the library.

- **Edit Symbol...** will open the specific instance of the symbol in the symbol editor. Editing this symbol will only affect this one instance of the symbol in the schematic. It will not affect other instances of the symbol in the schematic, and it will not affect the library copy of the symbol. You can also open a schematic symbol in the symbol editor by right clicking the symbol in the schematic and selecting **Edit with symbol editor** (Ctrl + E).
- **Edit Library Symbol...** will open the library copy of the symbol in the symbol editor. Editing the library copy of the symbol will edit the symbol in the symbol library, but will not immediately affect any instances of that symbol in the schematic. To update symbols in the schematic with changes to the library symbol, use the **Update Symbol from Library...** tool. Editing the library symbol in this way is equivalent to opening the symbol editor, opening the appropriate symbol in its library, and editing it.

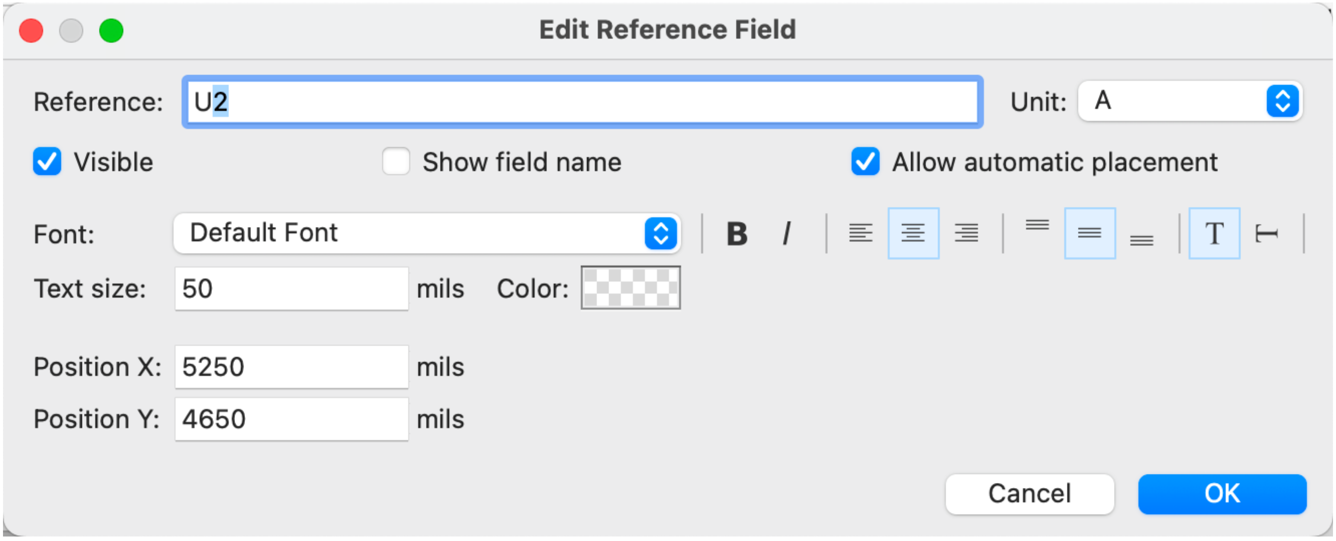
The **Update Symbol from Library...** button is used to update the schematic's copy of the symbol to match the copy in the library. The **Change Symbol...** button is used to swap the current symbol to a different symbol in the library. These functions are described later.

The **Simulation Model…** button opens the Simulation Model Editor for specifying the symbol's behavior in SPICE simulations.

Editing symbol fields individually

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

Some symbol fields have their own hotkey to edit them directly. With the symbol selected, the Reference, Value, and Footprint fields can be edited with the U, V, or F hotkeys, respectively.



Edit Reference Field

Reference: U2

Unit: A

☑ Visible

□ Show field name

☑ Allow automatic placement

Font: Default Font

Text size: 50 mils

Color:

Position X: 5250 mils

Position Y: 4650 mils

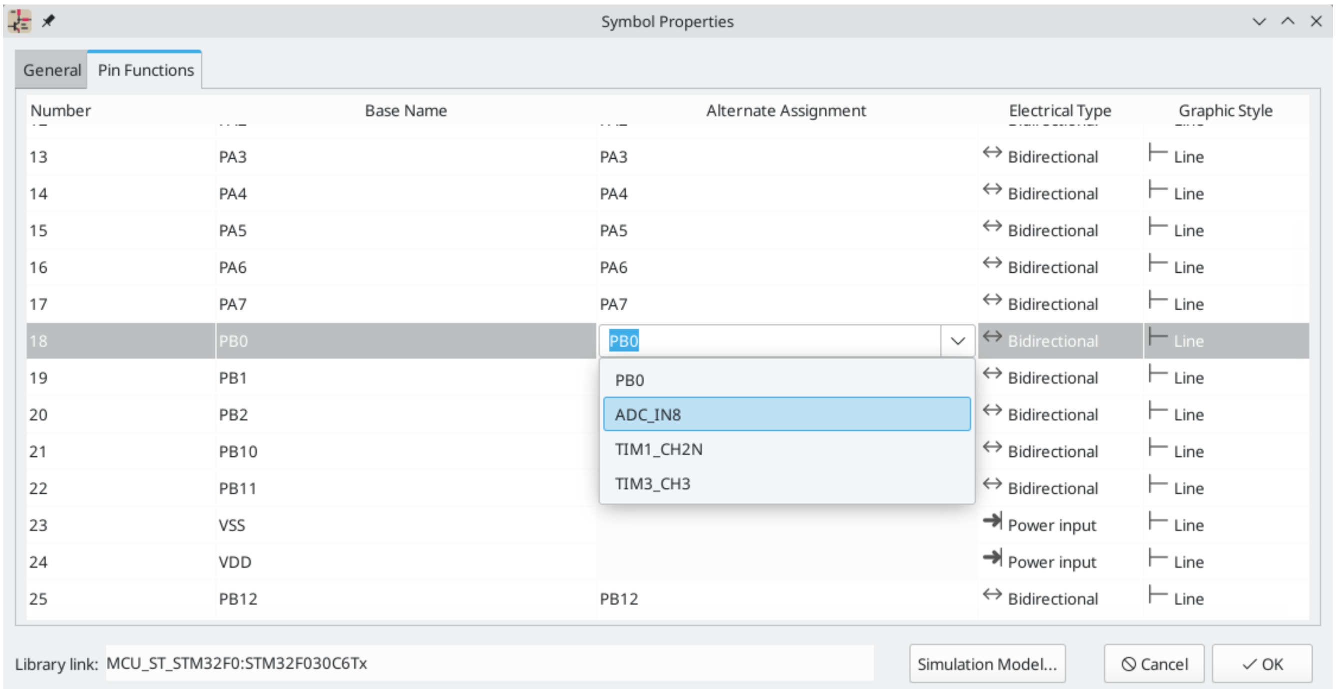
The options in this dialog are the same as those in the full Symbol Properties dialog, but are specific to a single field.

Symbol fields can be automatically moved to an appropriate location with the Autoplace Fields action (select a symbol and press O). Field autoplacement is configurable in the Schematic Editor's Editing Options, including a setting to always autoplace fields. You can also disable autoplacement for individual fields in the Symbol Properties or Field Properties dialogs.

Alternate pin functions

Symbol pins can have alternate pin functions defined for them. Alternate pin functions allow you to select a different name, electrical type, and graphical style for a pin when a symbol has been placed in the schematic. This can be used for pins that have multiple functions, such as microcontroller pins.

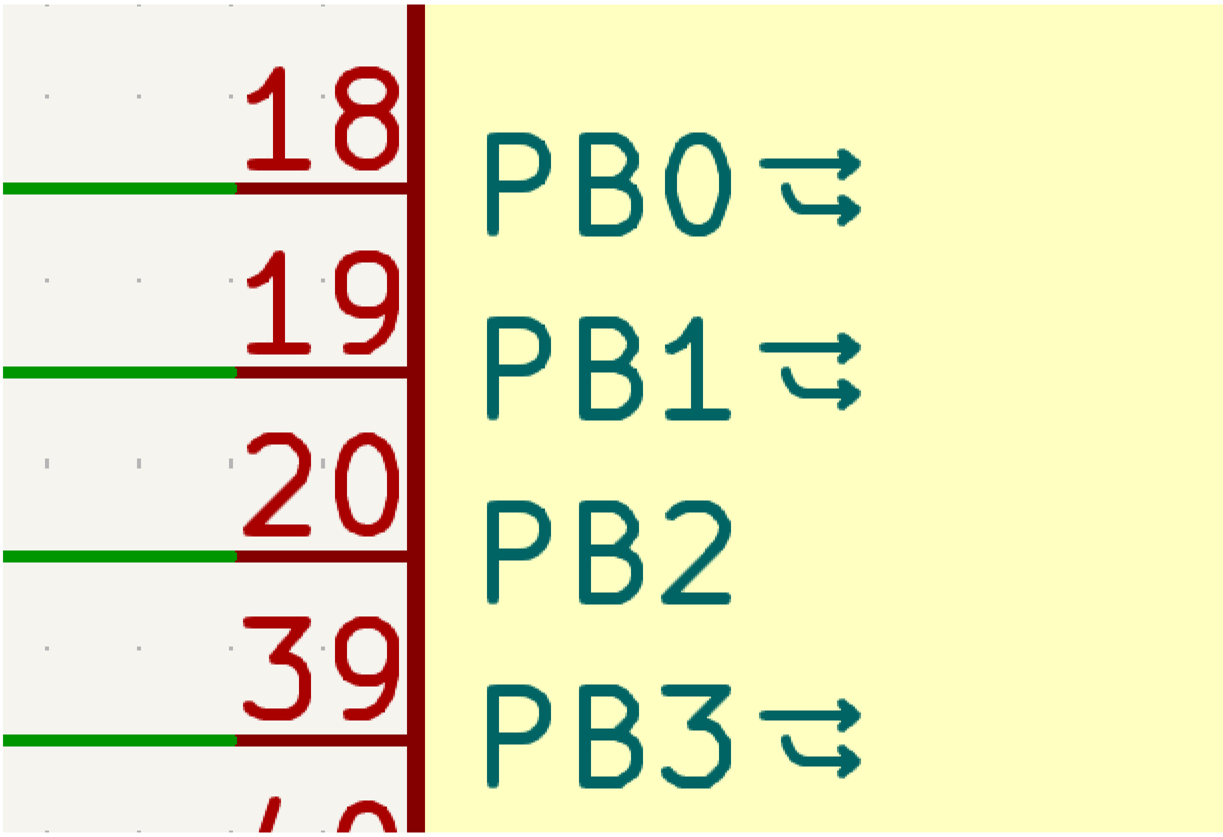
Alternate pin functions are selected once a symbol has been placed in the schematic. The pin function is selected in the **Pin Functions** tab of the Symbol Properties dialog. Alternate definitions are selectable in the dropdown in the Alternate Assignment column. You can also select an alternate pin by right-clicking the pin and selecting a new function from the **Pin Function** menu.



Number	Base Name	Alternate Assignment	Electrical Type	Graphic Style
13	PA3	PA3	↔ Bidirectional	Line
14	PA4	PA4	↔ Bidirectional	Line
15	PA5	PA5	↔ Bidirectional	Line
16	PA6	PA6	↔ Bidirectional	Line
17	PA7	PA7	↔ Bidirectional	Line
18	PB0	PB0	↔ Bidirectional	Line
19	PB1		↔ Bidirectional	Line
20	PB2		↔ Bidirectional	Line
21	PB10		↔ Bidirectional	Line
22	PB11		↔ Bidirectional	Line
23	VSS		⇥ Power input	Line
24	VDD		⇥ Power input	Line
25	PB12	PB12	↔ Bidirectional	Line

Library link: MCU_ST_STM32F0:STM32F030C6Tx

Pins that have alternate functions available are displayed with a small graphical indicator next to the pin name, as shown in the screenshot below. To globally show or hide these indicators, use **View → Show Pin Alternate Icons**.



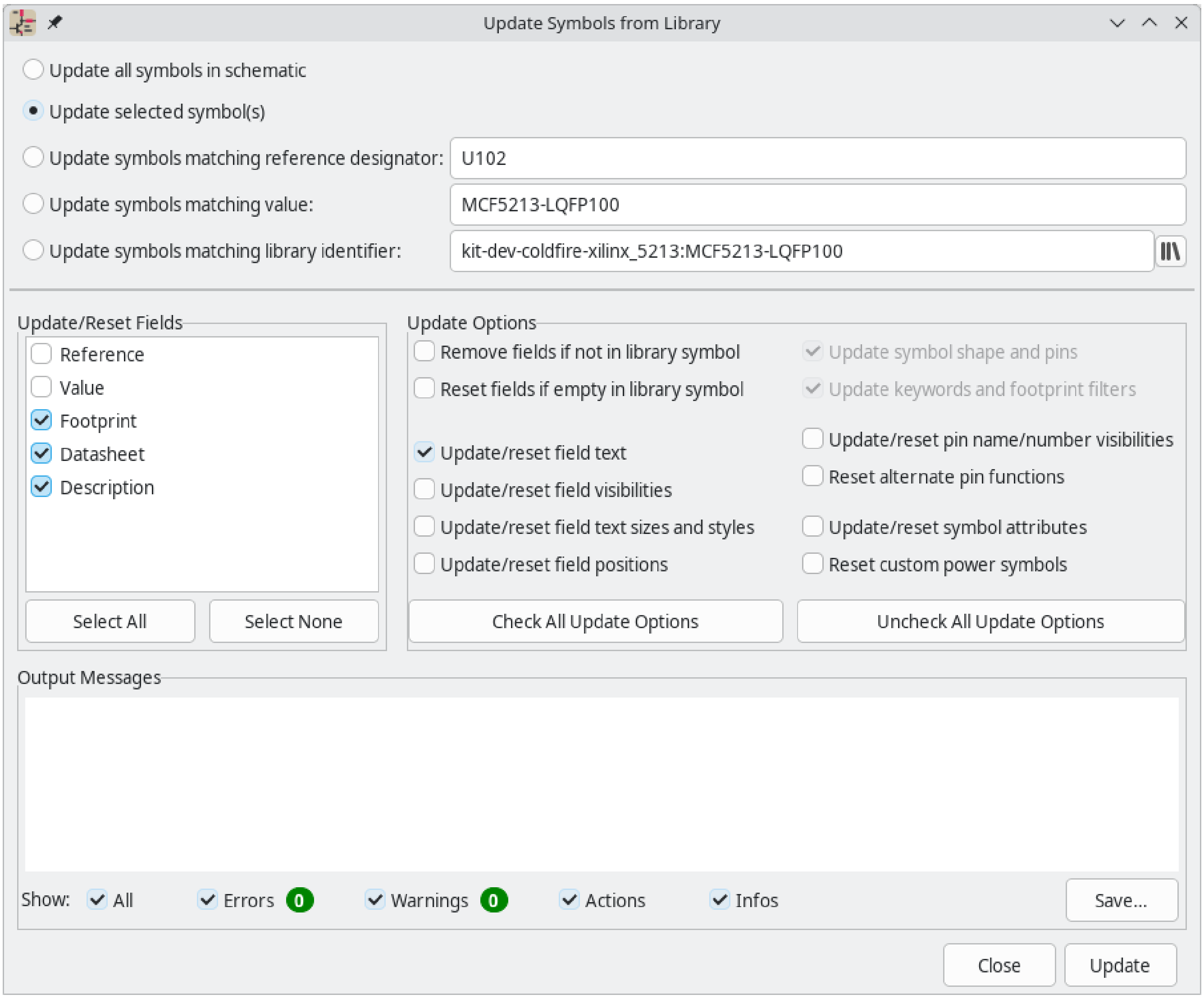
For information on how to add alternate pin functions to symbols, see the symbol editor documentation.

Updating and exchanging symbols

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

> **NOTE** You can use the Compare Symbol with Library tool to inspect the differences between a symbol in a schematic with its corresponding library symbol.

To update symbols in the schematic to match the corresponding library symbol, use **Tools** → **Update Symbols from Library...**, or right click a symbol and select **Update Symbol...**. You can also access the tool from the symbol properties dialog.



Update Symbols from Library

○ Update all symbols in schematic
● Update selected symbol(s)
○ Update symbols matching reference designator: U102
○ Update symbols matching value: MCF5213-LQFP100
○ Update symbols matching library identifier: kit-dev-coldfire-xilinx_5213:MCF5213-LQFP100

Update/Reset Fields
☐ Reference
☐ Value
☑ Footprint
☑ Datasheet
☑ Description

Update Options
☐ Remove fields if not in library symbol
☐ Reset fields if empty in library symbol
☑ Update/reset field text
☐ Update/reset field visibilities
☐ Update/reset field text sizes and styles
☐ Update/reset field positions
☑ Update symbol shape and pins
☑ Update keywords and footprint filters
☐ Update/reset pin name/number visibilities
☐ Reset alternate pin functions
☐ Update/reset symbol attributes
☐ Reset custom power symbols

Output Messages

Show: ☑ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

The top of the dialog has options to choose which symbols will be updated:

- **Update all symbols in schematic**: all symbols in the schematic will be updated to match the library versions of the symbols.
- **Update selected symbol(s)**: symbols that are selected in the schematic will be updated.
- **Update symbols matching reference designator**: symbols matching the specified reference designator will be updated. The reference designator field supports wildcards: `*` matches any number of any characters, including none, and `?` matches any single character.
- **Update symbols matching value**: symbols with the specified value will be updated. The value field supports wildcards: `*` matches any number of any characters, including none, and `?` matches any single

character.

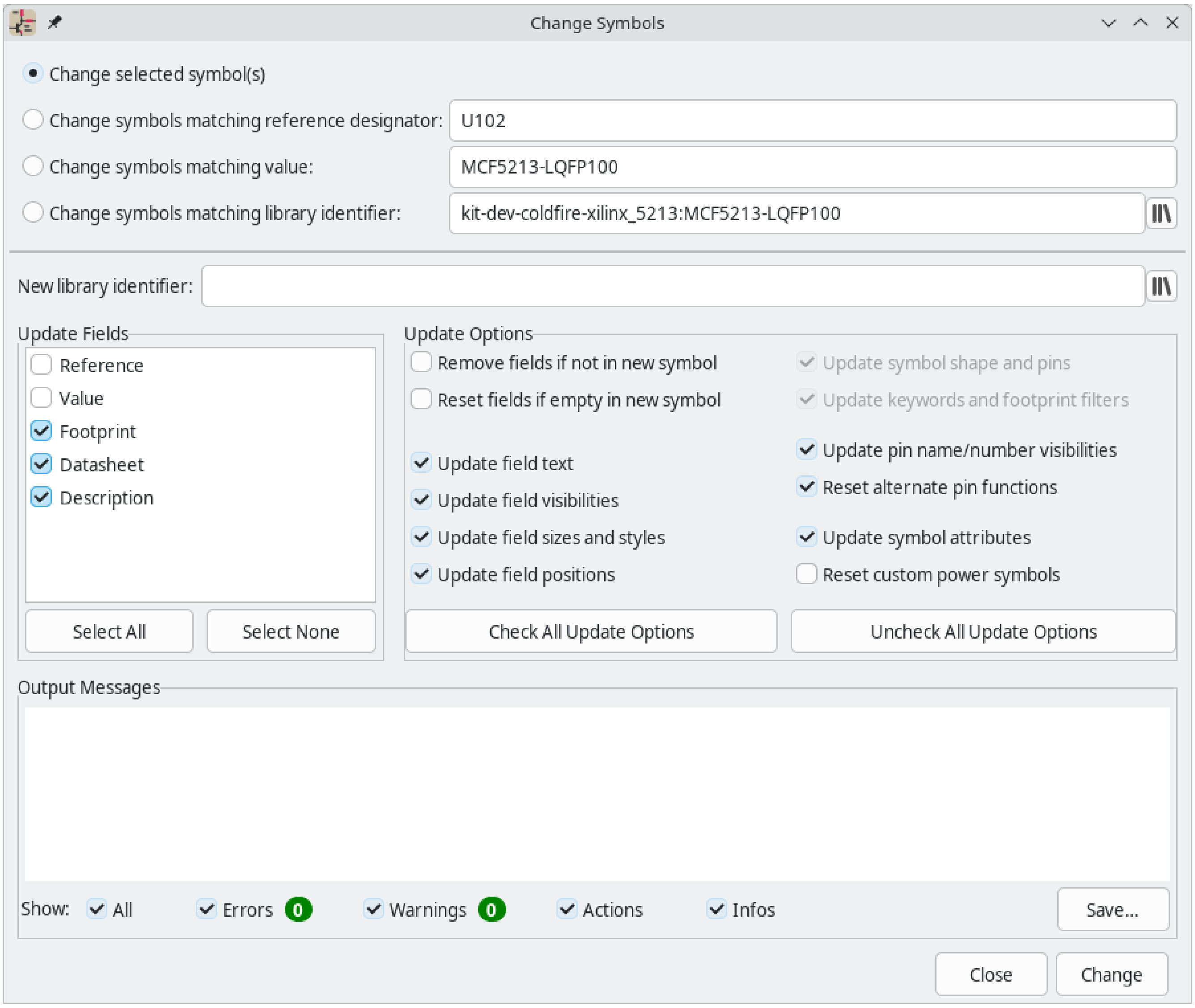
- **Update symbols matching library identifier**: symbols that match the specified library identifier will be updated. Library identifiers consist of the symbol library name and the symbol name, separated by `:`.

The middle of the dialog has options to control what parts of the symbol will be updated. On the left, you can select which fields will be modified (updated or reset). On the right, you can select how to update those fields:

- **Remove fields if not in library symbol**: if selected, any fields that are in the schematic version of the symbol but not the library version will be deleted.
- **Reset fields if empty in library symbol**: if selected, any fields that are empty in the library version of the symbol will be set to empty in the schematic version of the symbol.
- **Update/reset field text**: if selected, field contents in the schematic version of the symbol will be updated to match the fields in the library version of the symbol. Any fields that are empty in the library version of the symbol will not be updated unless **Reset fields if empty in library symbol** is selected.
- **Update/reset field visibilities**: if selected, fields in the schematic version of the symbol will have their visibility updated to match the library version of the symbol.
- **Update/reset field text sizes and styles**: if selected, fields in the schematic version of the symbol will have their text sizes and styles updated to match the library version of the symbol.
- **Update/reset field positions**: if selected, fields in the schematic version of the symbol will be moved to match the locations of the fields in the library version of the symbol.
- **Update symbol shape and pins**: the symbol's shape and pins are always updated to match the library version of the symbol.
- **Update keywords and footprint filters**: The symbol's keywords and footprint filters are always updated to match the library version of the symbol.
- **Update/reset pin name/number visibilities**: if selected, the visibility of pin names and numbers in the schematic version of the symbol will be updated to match the visibility of the pin names and numbers in the library version of the symbol.
- **Reset alternate pin functions**: if selected, alternate pin functions selected for the symbol's pins will be reset to default pin functions.
- **Update/reset symbol attributes**: if selected, the schematic symbol attributes (**do not populate**, **exclude from simulation**, **exclude from BOM**, **exclude from board**) will be updated to match the library version of the symbol.
- **Reset custom power symbols**: if selected, the `Value` field of power symbols in the schematic will be updated to match the library versions of the symbols. If not selected, the `Value` field of power symbols will not be updated, even if the `Value` field of other non-power symbols would be updated. Note that changing the `Value` field of power symbols will change the global net associated with the power symbol.

The bottom of the dialog displays messages describing the update actions that have been performed, with filters for which types of messages to display (errors, warnings, actions, and/or infos).

To change an existing symbol to a different symbol, use **Edit** → **Change Symbols...**, or right click an existing symbol and select **Change Symbol...**. This dialog is also accessible from the symbol properties dialog.



● Change selected symbol(s)

○ Change symbols matching reference designator: U102

○ Change symbols matching value: MCF5213-LQFP100

○ Change symbols matching library identifier: kit-dev-coldfire-xilinx_5213:MCF5213-LQFP100

New library identifier:

Update Fields

- □ Reference
- □ Value
- ☑ Footprint
- ☑ Datasheet
- ☑ Description

Update Options

- □ Remove fields if not in new symbol
- □ Reset fields if empty in new symbol
- ☑ Update field text
- ☑ Update field visibilities
- ☑ Update field sizes and styles
- ☑ Update field positions
- ☑ Update symbol shape and pins
- ☑ Update keywords and footprint filters
- ☑ Update pin name/number visibilities
- ☑ Reset alternate pin functions
- ☑ Update symbol attributes
- □ Reset custom power symbols

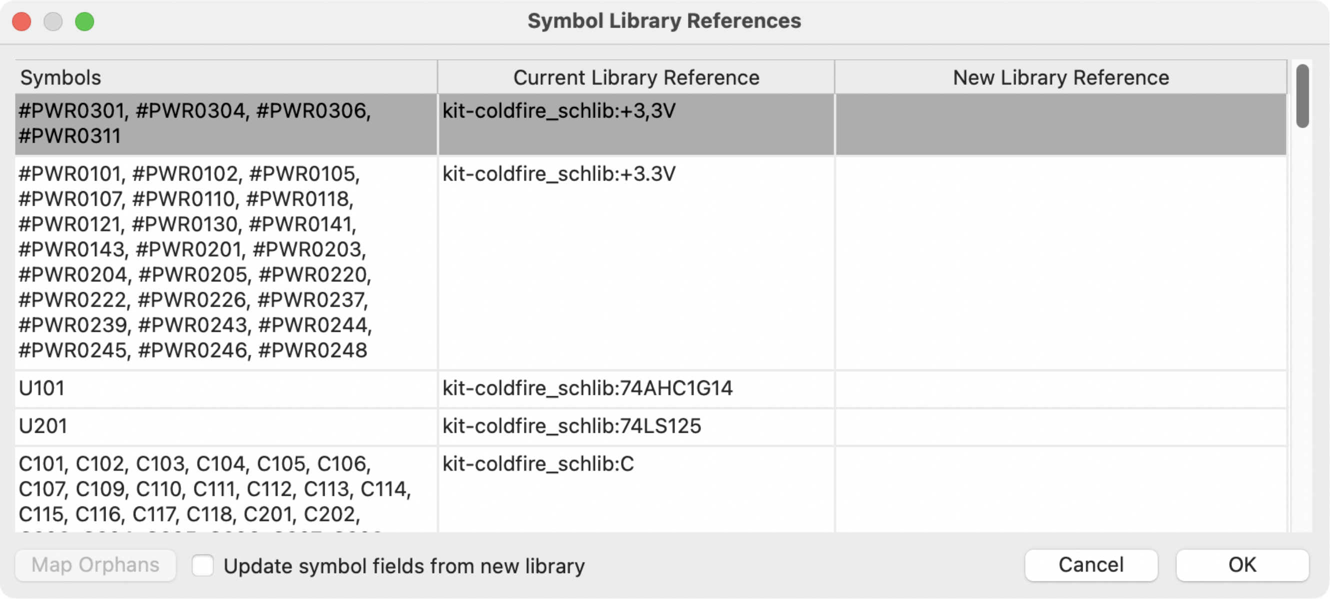
Output Messages

Show: ☑ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

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

Another way to swap existing symbols for new ones is to use **Tools → Edit Symbol Library Links...**. This dialog contains a table of every symbol in the design, grouped by current library symbol. By choosing a new symbol in the **New Library Reference** column, you can make all instances of the existing symbol instead point to the new symbol. If the **Update symbol fields from new library** option is used, the contents of the existing symbols' fields will be updated to match the new symbols' fields.

The **Map Orphans** button attempts to automatically remap orphaned symbols to symbols with the same name in an active library. For example, if there is a symbol with the current library reference `mylib:symbol123`, but the `mylib` library cannot be found, the **Map Orphans** button will attempt to find a symbol named `symbol123` in any of the libraries that are present. This button is only enabled if orphaned symbols are present in the schematic (see the legacy schematics section).



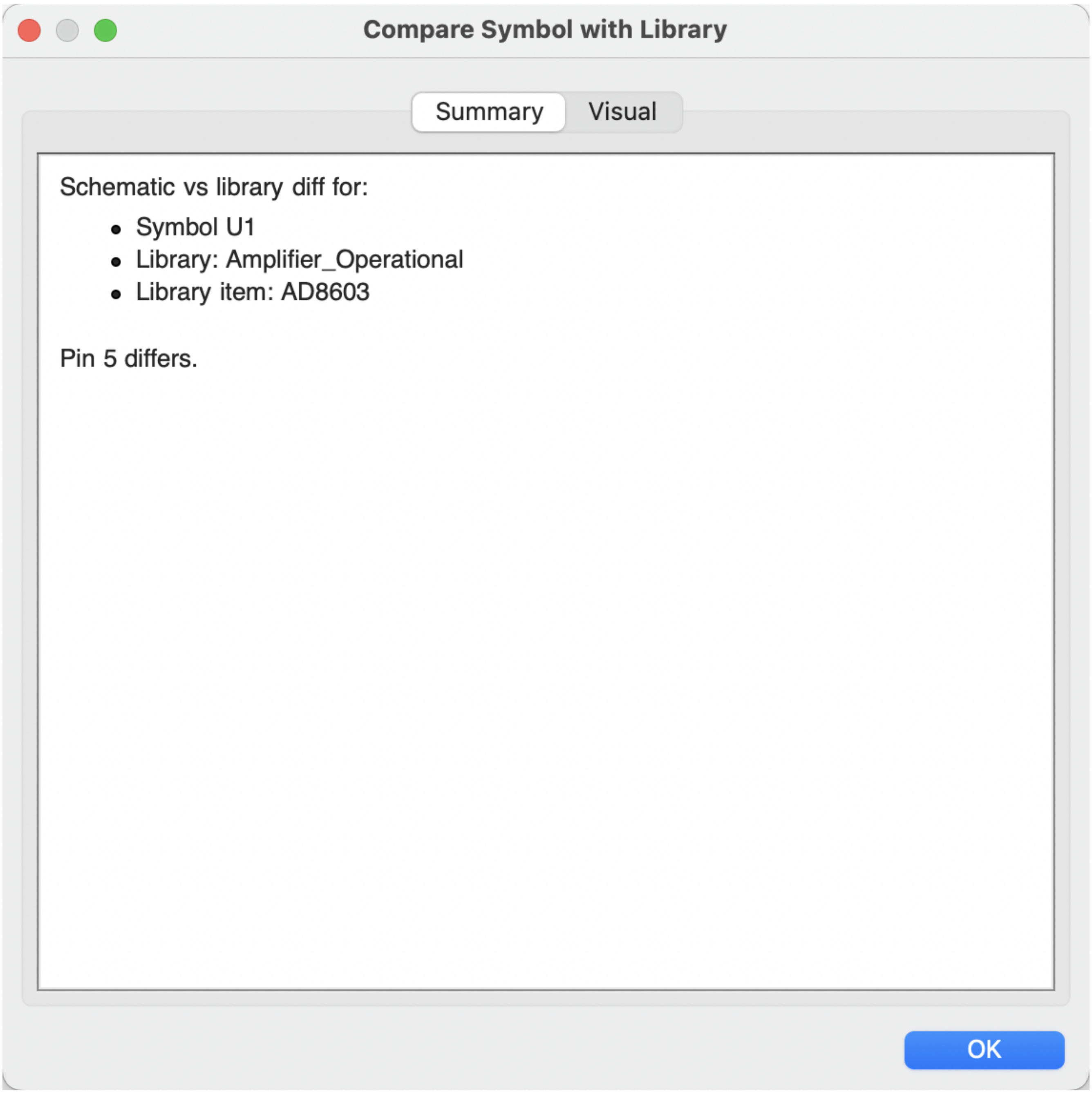
Symbols	Current Library Reference	New Library Reference
#PWR0301, #PWR0304, #PWR0306, #PWR0311	kit-coldfire_schlib:+3,3V	
#PWR0101, #PWR0102, #PWR0105, #PWR0107, #PWR0110, #PWR0118, #PWR0121, #PWR0130, #PWR0141, #PWR0143, #PWR0201, #PWR0203, #PWR0204, #PWR0205, #PWR0220, #PWR0222, #PWR0226, #PWR0237, #PWR0239, #PWR0243, #PWR0244, #PWR0245, #PWR0246, #PWR0248	kit-coldfire_schlib:+3.3V	
U101	kit-coldfire_schlib:74AHC1G14	
U201	kit-coldfire_schlib:74LS125	
C101, C102, C103, C104, C105, C106, C107, C109, C110, C111, C112, C113, C114, C115, C116, C117, C118, C201, C202,	kit-coldfire_schlib:C	

□ Update symbol fields from new library

This dialog is primarily useful for managing symbols that appear in multiple libraries, when you want to switch from one library to another. For example, if a schematic uses symbols that are in both a global library and a project-specific library, the Symbol Library References dialog could be used to switch between using the global symbols or the equivalent project-specific symbols. It does not have features for fine-grained control of how fields are updated; for that, use the Change Symbols dialog.

Comparing symbols between schematic and library

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

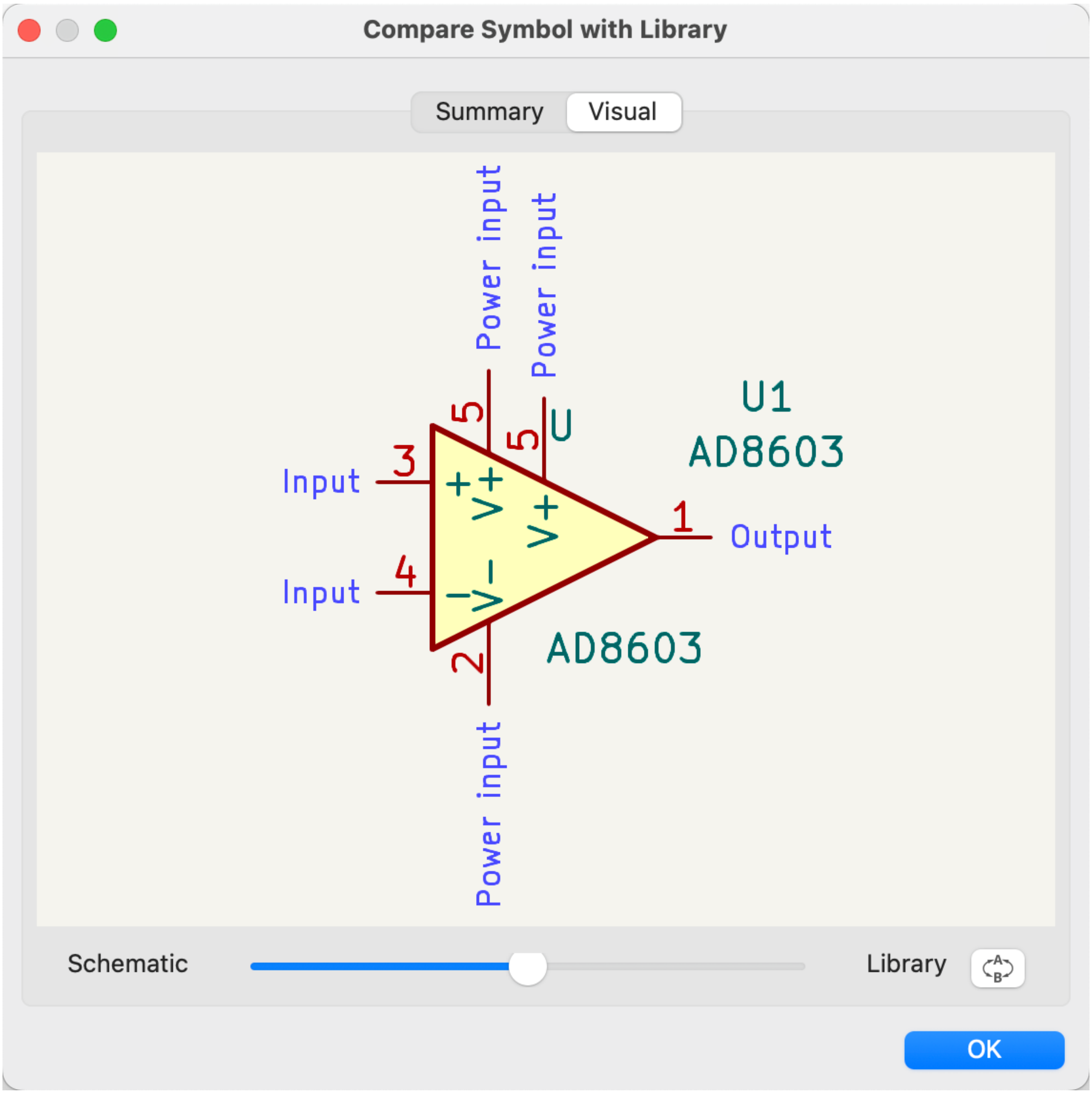


Schematic vs library diff for:

- Symbol U1
- Library: Amplifier_Operational
- Library item: AD8603

Pin 5 differs.

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



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

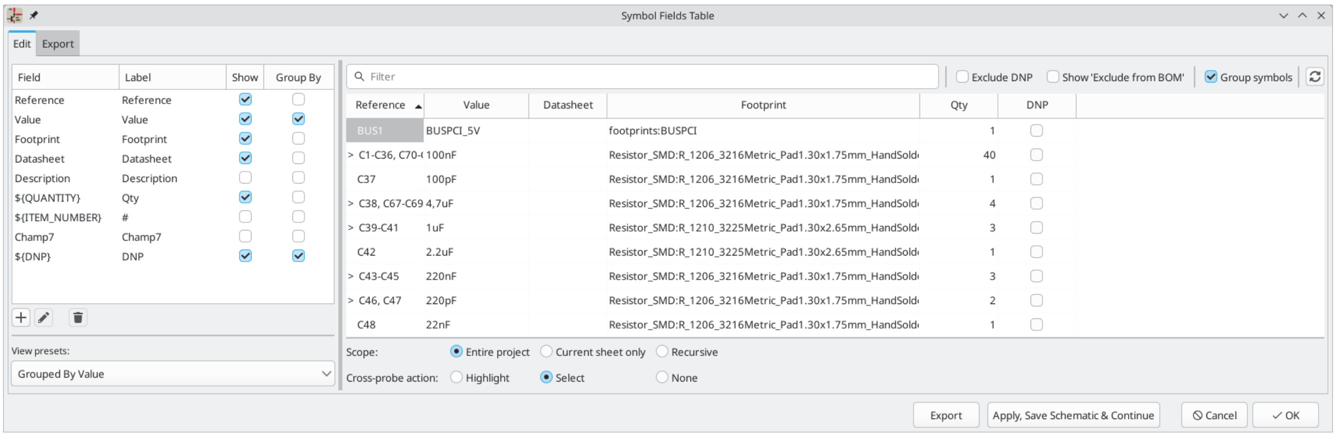
By default, the comparison displays both versions of the symbol superimposed on each other. To see the changes more easily, you can drag the slider at the bottom of the tab to the right to emphasize the library version of the symbol in the superimposed view (making the schematic version of the symbol more transparent) or drag it to the left to emphasize the schematic version (making the library version more transparent). At the far right and left ends of the slider, the schematic and library versions of the symbol, 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 schematic and library versions.

The screenshot above shows a visual comparison with the schematic version of the symbol deemphasized. You can see a partially transparent pin 5 (from the schematic version of the symbol) is in a different location than the fully opaque pin 5 (from the library symbol). This indicates that the pin was moved in either the schematic or library version of the symbol.

Symbol Fields Table

The Symbol Fields Table allows you to view and modify field values for all symbols in a spreadsheet interface. The Symbol Fields Table also acts as a BOM export tool. You can open the Symbol Fields Table with the ▦ button.



Field	Label	Show	Group By
Reference	Reference	☑	□
Value	Value	☑	☑
Footprint	Footprint	☑	□
Datasheet	Datasheet	☑	□
Description	Description	□	□
${QUANTITY}	Qty	☑	□
${ITEM_NUMBER}	#	□	□
Champ7	Champ7	□	□
${DNP}	DNP	☑	☑

View presets: Grouped By Value

□ Exclude DNP □ Show 'Exclude from BOM' ☑ Group symbols

Reference	Value	Datasheet	Footprint	Qty	DNP
BUS1	BUSPCI_5V		footprints:BUSPCI	1	□
> C1-C36, C70-(	100nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	40	□
C37	100pF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	1	□
> C38, C67-C69	4,7uF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	4	□
> C39-C41	1uF		Resistor_SMD:R_1210_3225Metric_Pad1.30x2.65mm_HandSold	3	□
C42	2.2uF		Resistor_SMD:R_1210_3225Metric_Pad1.30x2.65mm_HandSold	1	□
> C43-C45	220nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	3	□
> C46, C47	220pF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	2	□
C48	22nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	1	□

Scope: ● Entire project ○ Current sheet only ○ Recursive

Cross-probe action: ○ Highlight ● Select ○ None

Cells are navigated with the arrow keys, or with `Tab` / `Shift` + `Tab` to move right / left and `Enter` to move down, respectively.

A range of cells can be selected by clicking and dragging. The whole range of selected cells will be copied (`Ctrl` + `C`) or pasted into (`Ctrl` + `V`) on a copy or paste action. Copying a range of cells from the table can be useful for creating a BOM. More details of copying and pasting cells are described below.

The left pane contains a list of all available symbol fields, as well as some virtual fields such as Quantity and Item Number. You can add or remove any symbol field from the main table on using the **Include** checkboxes (fields can also be shown or hidden by right-clicking on the header of the main table). New symbol fields can be added using the + button; a field with that name will be added to every symbol. To rename the field, which changes the field name in all symbols, use the ✎ button. The 🗑 button deletes the field from all symbols. You can collapse this panel by clicking the ⬅ button.

Each field has its own column label, which is displayed at the top of the corresponding column in the symbol fields table and in exported BOMs. The column label for each field is shown in the second column of in the left pane. A column label does not have to match the field name. To change a field's column label, click in the **BOM Name** column for that field, then enter a new name.

Similar symbols can optionally be grouped by any symbol field using the **Group By** checkboxes. Symbols are grouped into a single row in the table if all of their **Group By** fields are identical. The grouped row can be expanded to show the individual symbols by clicking the arrow at the left of the row. The **Group Symbols** checkbox enables or disables symbol grouping, and the 🔄 button recalculates groupings.

Presets are available to configure the list of fields. Presets store which fields are displayed, which fields are used for grouping, and the column order. You can create and save your own presets or use one of several default presets. Custom presets can be deleted in this dialog or in the Schematic Setup dialog.

Symbols can be filtered by reference designator using the **Filter** textbox at the top. The filter supports wildcards: `*` matches any number of any characters, including none, and `?` matches any single character.

You can also change the display scope using the dropdown menu to the right of the filter box. You can show only symbols in the current sheet, the current sheet and all of its subsheets, or the entire project.

The ⚙ menu contains some additional options for showing or hiding symbols.

- **Include 'DNP' Symbols** includes or excludes symbols with the **Do Not Populate** attribute from both the **Edit** tab and any BOM exports.
- **Include 'Exclude from BOM' Symbols** includes or excludes symbols with the **Exclude from BOM** attribute from the **Edit** tab. Symbols with the **Exclude from BOM** attribute are always excluded from BOM exports.

You can cross-probe from this dialog by selecting a row in the table. Depending on the cross-probe options selected in the ⚙ menu, this can highlight the corresponding symbol in the schematic, select the corresponding symbol in the schematic, or do nothing. The selection action can also select the symbol's footprint in the board editor, depending on the PCB Editor cross-probing settings configured in Preferences.

The Symbol Fields Table is also a bill of materials tool. You can use the **Export** button to save the symbol fields to an external file. The fields are exported to the BOM exactly as they are currently shown in the spreadsheet view. File format settings are configured in the **Export** tab. For more information about exporting a BOM, see the BOM tool documentation.

Virtual fields

If you create a field in the Symbol Fields Table whose name begins with a text variable, a virtual field will be created. Virtual fields have a value that is evaluated for each symbol based on the contents of the field name. For example, a virtual field named `${SYMBOL_NAME}` will evaluate to the symbol's name for each symbol. A virtual field can contain any text, as long as it starts with a text variable, so a virtual field named `${SYMBOL_LIBRARY}:${SYMBOL_NAME}` will evaluate to `<library name>:<symbol name>` for each symbol.

Virtual fields exist only in the Symbol Fields Table and in BOM exports. While they are displayed as a column in the dialog and BOMs, and they can be used to group or sort symbols in BOM exports just like regular fields, adding a virtual field in the Symbol Fields Table does not add a corresponding field to each symbol in the schematic.

Any text variable can be used in virtual fields, including sheet and project text variables.

Text variables that correspond to symbol attributes (`${DNP}`, `${EXCLUDE_FROM_BOARD}`, `${EXCLUDE_FROM_SIM}`, `${EXCLUDE_FROM_BOM}`) are displayed specially. In the Symbol Fields Table, they are shown as checkboxes for each symbol that directly set or unset the corresponding symbol attribute. In BOM exports, they expand to the friendly name of the attribute if the attribute is set (e.g. `Excluded from board` for `${EXCLUDE_FROM_BOARD}` and `DNP` for `${DNP}`) or to an empty string if the attribute is not set.

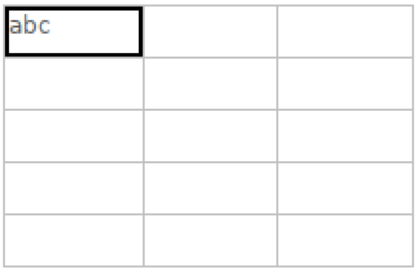
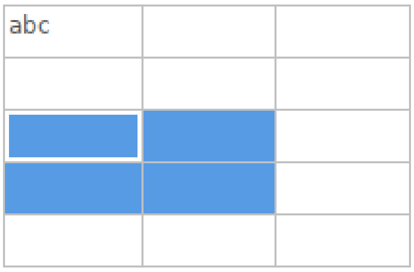
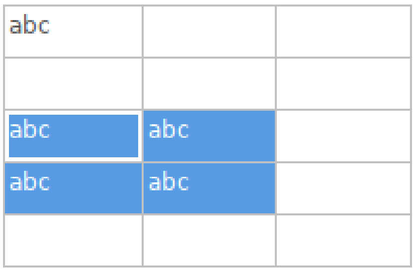
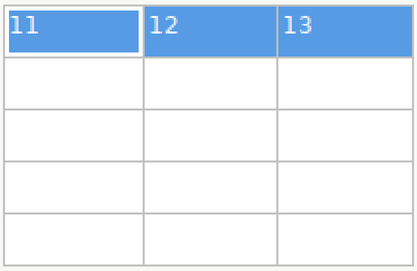
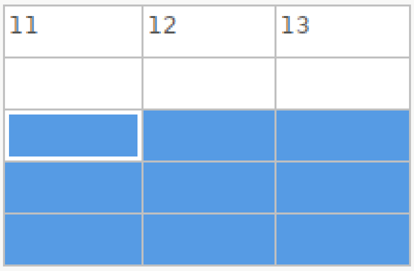
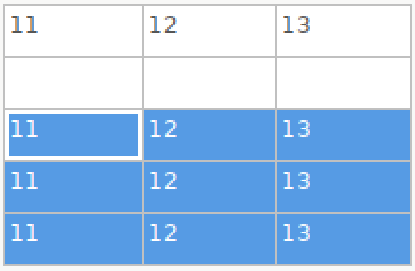
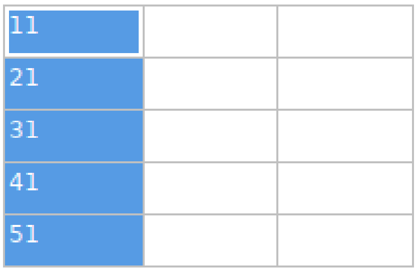
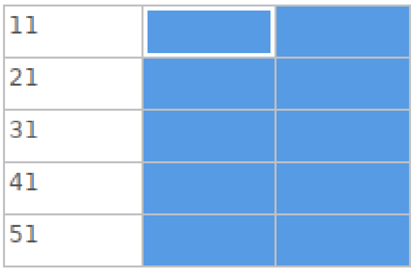
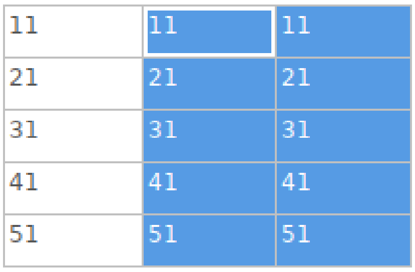
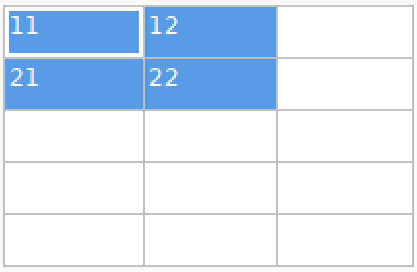
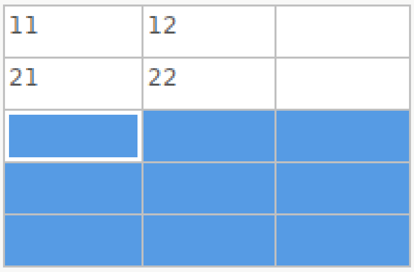
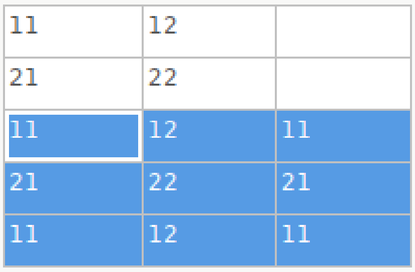
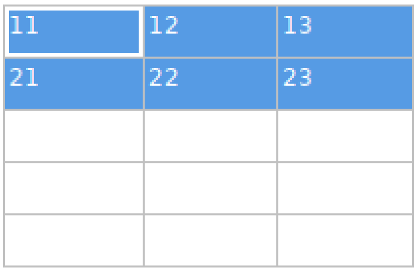
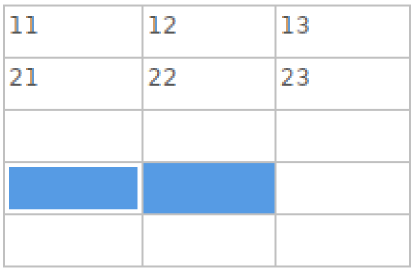
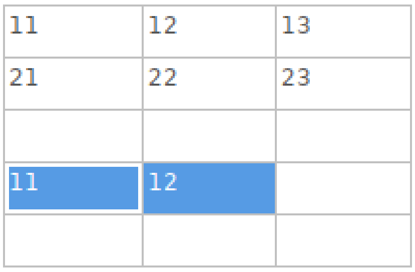
Finally, there are two special virtual fields that can be created:

- `${QUANTITY}` is a virtual field that contains the number of grouped instances of each symbol.
- `${ITEM_NUMBER}` is a virtual field that contains the row number of each symbol in the table.

Tricks to simplify filling fields

There are several special copy/paste methods in the spreadsheet for pasting values into larger regions, including auto-incrementing pasted cells. These features may be useful when pasting values that are shared in several symbols.

These methods are illustrated below.

1. Copy (Ctrl + C)	**2. Select target cells**	**3. Paste (Ctrl + V)**
 abc	 abc	 abc abc abc abc abc
 11 12 13	 11 12 13	 11 12 13 11 12 13 11 12 13 11 12 13
 11 21 31 41 51	 11 21 31 41 51	 11 11 11 21 21 21 31 31 31 41 41 41 51 51 51
 11 12 21 22	 11 12 21 22	 11 12 21 22 11 12 11 21 22 21 11 12 11
 11 12 13 21 22 23	 11 12 13 21 22 23	 11 12 13 21 22 23 11 12

NOTE These techniques are also available in other dialogs with a grid control element.

Reference designators and symbol annotation

Reference designators are unique identifiers for components in a design. They are often printed on a PCB and in assembly diagrams, and allow you to match symbols in a schematic to the corresponding components on a board.

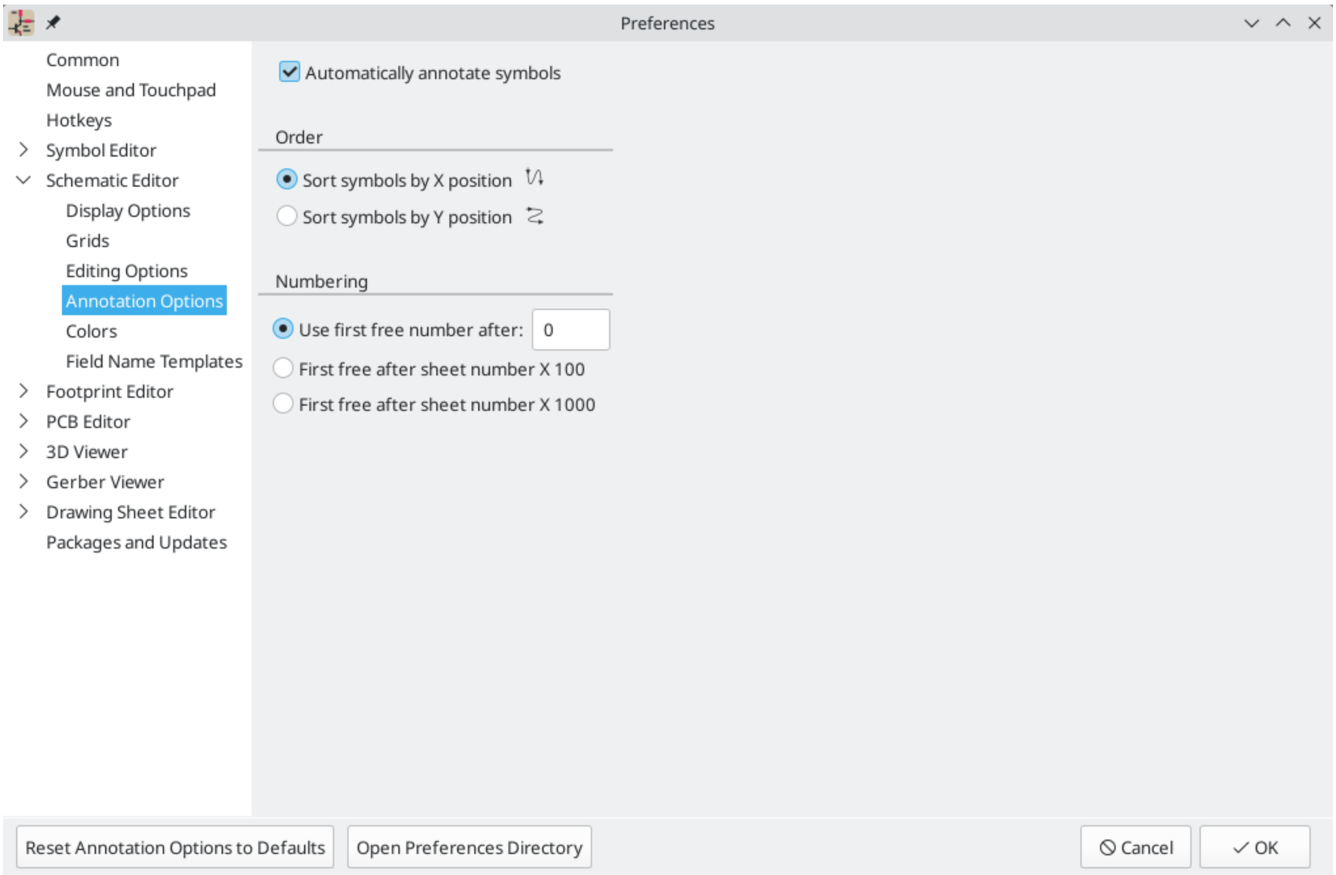
In KiCad, reference designators consist of a letter indicating the type of component (`R` for resistor, `C` for capacitor, `U` for IC, etc.) followed by a number. If the symbol has multiple units then the reference designator will also have a trailing letter indicating the unit. Symbols that don't have a reference designator set have a `?` character instead of the number. Reference designators must be unique.

Reference designators can be automatically set when symbols are added to the schematic, and you can set or reset reference designators yourself by manually editing an individual symbol's reference designator field or in bulk using the Annotation tool.

NOTE The process of setting a symbol's reference designator is called **annotation**.

Auto-annotation

When auto-annotation is enabled, symbols will be automatically annotated when they are added to the schematic. You can enable auto-annotation by checking the **Automatically annotate symbols** checkbox in the **Schematic Editor** → **Annotation Options** pane in **Preferences**. Auto-annotation can also be toggled using the R?? R42 button in the left toolbar.



Preferences

Common
Mouse and Touchpad
Hotkeys
Symbol Editor
Schematic Editor
Display Options
Grids
Editing Options
Annotation Options
Colors
Field Name Templates
Footprint Editor
PCB Editor
3D Viewer
Gerber Viewer
Drawing Sheet Editor
Packages and Updates

☑ Automatically annotate symbols

Order

● Sort symbols by X position
○ Sort symbols by Y position

Numbering

● Use first free number after: 0
○ First free after sheet number X 100
○ First free after sheet number X 1000

Reset Annotation Options to Defaults
Open Preferences Directory
Cancel
OK

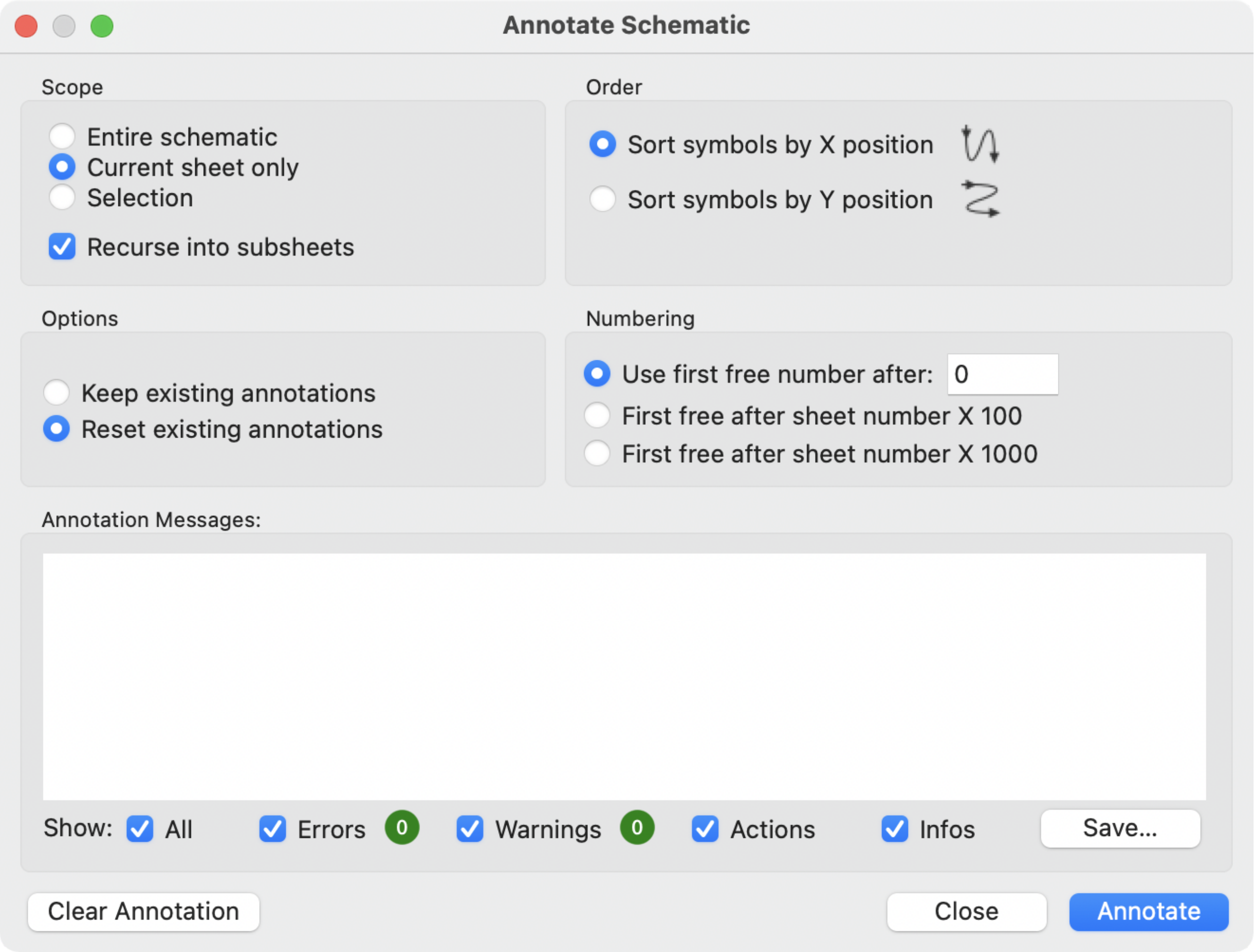
When multiple symbols are added simultaneously, they are annotated according to the **Order** setting, sorted by either X or Y position.

The **Numbering** option sets the starting number for new reference designators. This can be the lowest available number, or a number based on the sheet number.

For more information about annotation options, see the documentation for the Annotation tool.

Alat Anotasi

The Annotation tool automatically assigns reference designators to symbols in the schematic. To launch the Annotation tool, click the R?? R42 button in the top toolbar.



Annotate Schematic

Scope

○ Entire schematic
● Current sheet only
○ Selection

☑ Recurse into subsheets

Order

● Sort symbols by X position
○ Sort symbols by Y position

Options

○ Keep existing annotations
● Reset existing annotations

Numbering

● Use first free number after: 0
○ First free after sheet number X 100
○ First free after sheet number X 1000

Annotation Messages:

Show: ☑ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

The tool provides several options to control how symbols are annotated.

Scope: Selects whether annotation is applied to the entire schematic, to only the current sheet, or to only the selected symbols. If the **Recurse into subsheets** option is selected, symbols in subsheets of the selected scope will be reannotated; otherwise symbols in subsheets will not be reannotated. For example, if **Recurse into subsheets** and **Selection only** selected, symbols in any selected subsheets will be reannotated.

Options: Selects whether annotation should apply to all symbols and reset existing reference designators, or apply only to unannotated symbols.

Order: Chooses the direction of numbering. If symbols are sorted by X position, all symbols on the left side of a schematic sheet will be lower numbered than symbols on the right side of the sheet. If symbols are sorted by Y position, all symbols on the top of a sheet will be lower numbered than symbols at the bottom of the sheet.

Numbering: Selects the starting point for numbering reference designators. The lowest unused number above the starting point is picked for each reference designator. The starting point can be an arbitrary number (typically zero), or it can be the sheet number multiplied by 100 or 1000 so that each part's reference designator corresponds to the schematic page it is on.

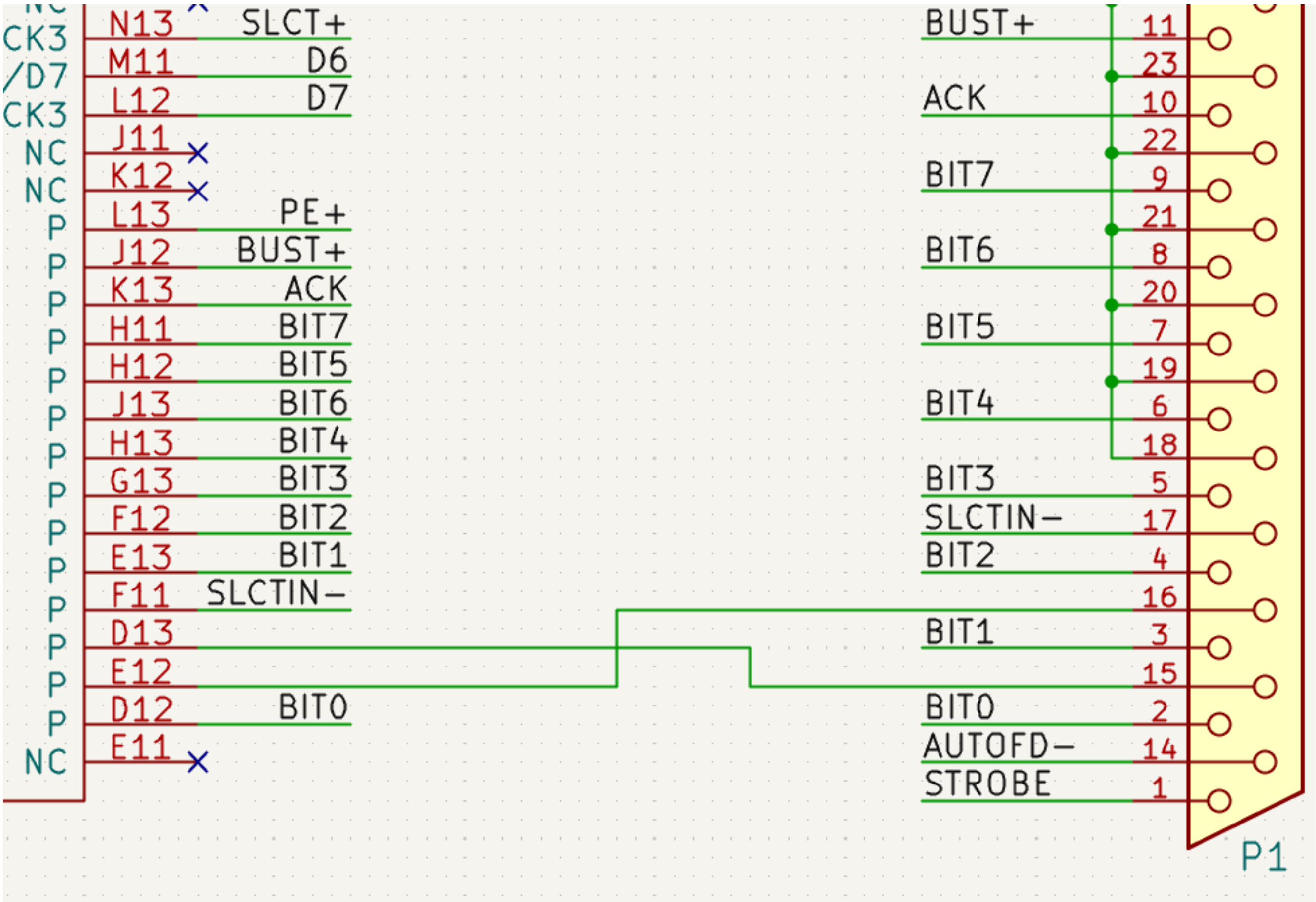
The **Clear Annotation** button clears all reference designators in the selected scope.

Annotation messages can be filtered with the checkboxes at the bottom or saved to a report using the **Save...** button.

Electrical connections

Electrical connections in KiCad are known as *nets*, where each net represents a circuit node. All symbol pins on the same net are considered electrically connected.

There are two primary ways to establish connections: wires and labels. Wires make direct connections between symbol pins and other wires, while labels connect to other labels with the same name. Both wires and labels are shown in the schematic below.



Connections can also be made with buses and with implicit connections via hidden power pins.

Net name assignment rules

Every net in the schematic is assigned a name, whether that name is specified by the user (using a label) or automatically generated by KiCad.

When multiple labels are attached to the same net, the final net name is determined in the following order, from highest priority to lowest:

1. Global labels
2. Power symbols
3. Local labels

   Hierarchical labels

5. Hierarchical sheet pins
   - Output sheet pins are higher priority than Input sheet pins

If there are multiple labels of one type attached to a net, the names are sorted alphabetically and the first is used.

If a net travels through multiple sheets of a hierarchy, it will take its name from the highest level of the hierarchy where it has a hierarchical label or local label. As usual, local labels take priority over hierarchical labels.

If none of the label types above are attached to a net, the net's name is automatically generated based on the connected symbol pins.

Wires

Wires are used to directly establish electrical connections between two points. To establish a connection, a segment of wire must be connected by its end to another segment or to a pin. Only wire ends create connections; if a wire crosses the middle of another wire, a connection will not be made.

Unconnected wire ends have a small square that indicates the connection point. The square disappears when a connection is made to the wire end. Unconnected pins have a circle, which also disappears when a connection is made.

NOTE Wires connect with other wires or pins only if their ends coincide exactly. Therefore it is important to keep symbol pins and wires aligned to the grid. It is recommended to always use a 50 mil grid when placing symbols and drawing wires because the KiCad standard symbol library and all libraries that follow its style also use a 50 mil grid.

NOTE Symbols, wires, and other elements that are not aligned to the grid can be snapped back to the grid by selecting them, right clicking, and selecting **Align Elements to Grid**.

Drawing and editing wires

To begin connecting elements with wire, use the Wire tool in the right toolbar (W). Wires can also be automatically started by clicking on an unconnected symbol pin or wire end.

You can restrict wires to 90 degree angles using the button in the left toolbar, or to 45 degree angles with the button. The button allows you to place wires at any angle. You can cycle through these modes using Shift+Space, or select the desired mode in **Preferences** → **Schematic Editor** → **Editing Options**. These modes affect graphic lines in addition to wires.

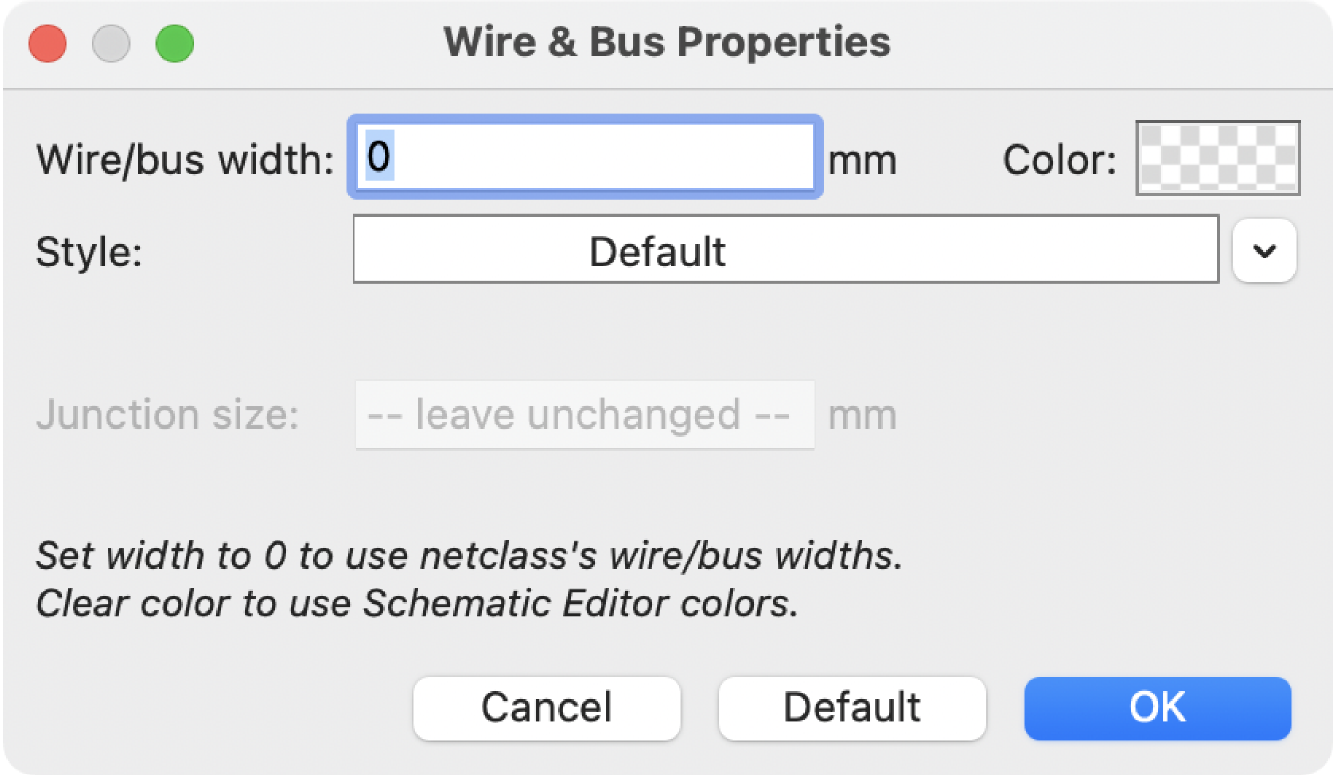
As in the PCB editor, the / hotkey switches wire posture.

Wires can be moved and edited using the Move (M) or Drag (G) tools. As with symbols, the **Move** tool moves only the selected segment, without maintaining existing connections to other segments. The **Drag** tool maintains existing connections.

You can select connected wires using the **Select Connection** tool (Alt+4). This tool selects all connected wire segments until it reaches a junction, starting with the selected segment or the segment under the cursor. Using the tool again expands the existing selection to the next junction.

You can break a wire segment into two pieces by right-clicking a wire and selecting **Slice**. The segment will be separated at the current mouse position. You can also separate a wire segment from the adjacent segments by right-clicking the segment and selecting **Break**.

Normally the line style of a wire follows the net's net class settings (nets are in the `Default` net class if no other net class is specified). However, the line style for the selected wire segments can be overridden in the wire's properties dialog (E when a wire segment is selected). The wire's width, color, and line style (solid, dashed, dotted, etc.) can be set. Setting the width to `0`, clearing the color, and using the `Default` line style uses the default width, color, and style, respectively, from the net class settings. If a wire junction is included in the selection, the junction size can also be edited here.



Wire & Bus Properties

Wire/bus width: 0 mm

Color:

Style: Default

Junction size: -- leave unchanged -- mm

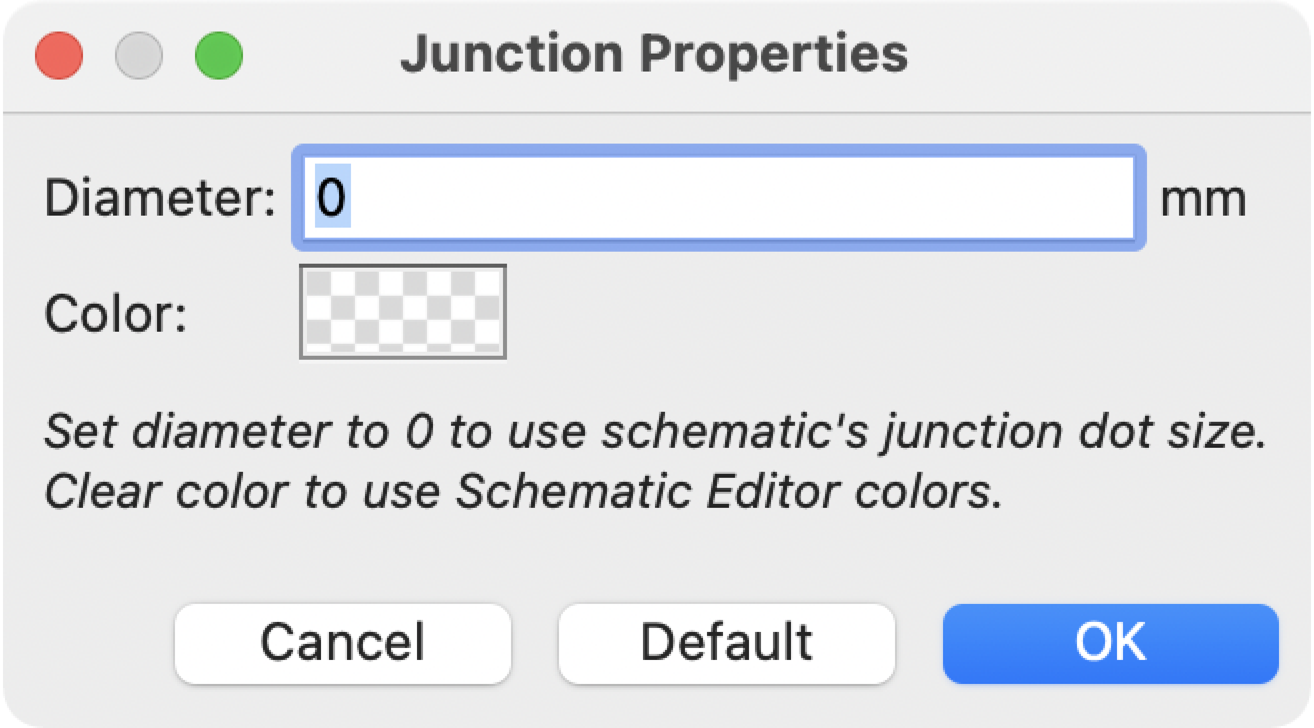
Set width to 0 to use netclass's wire/bus widths.
Clear color to use Schematic Editor colors.

Wire junctions

Wires that cross are not implicitly connected. It is necessary to join them by explicitly adding a junction dot if a connection is desired (button in the right toolbar). Junction dots will be automatically added to wires that start or end on top of an existing wire.

Junction dots are used in the schematic figure above on the wires connected to `P1` pins 18, 19, 20, 21, 22, and 23.

Junction size automatically follows the schematic's **Junction dot size** setting in **Schematic Setup** → **General** → **Formatting**. Color follows the net class setting. The automatic size and color can be overridden in each junction dot's properties; a size of `0` is equivalent to the schematic default size, and clearing the color uses the net class color.



Junction Properties

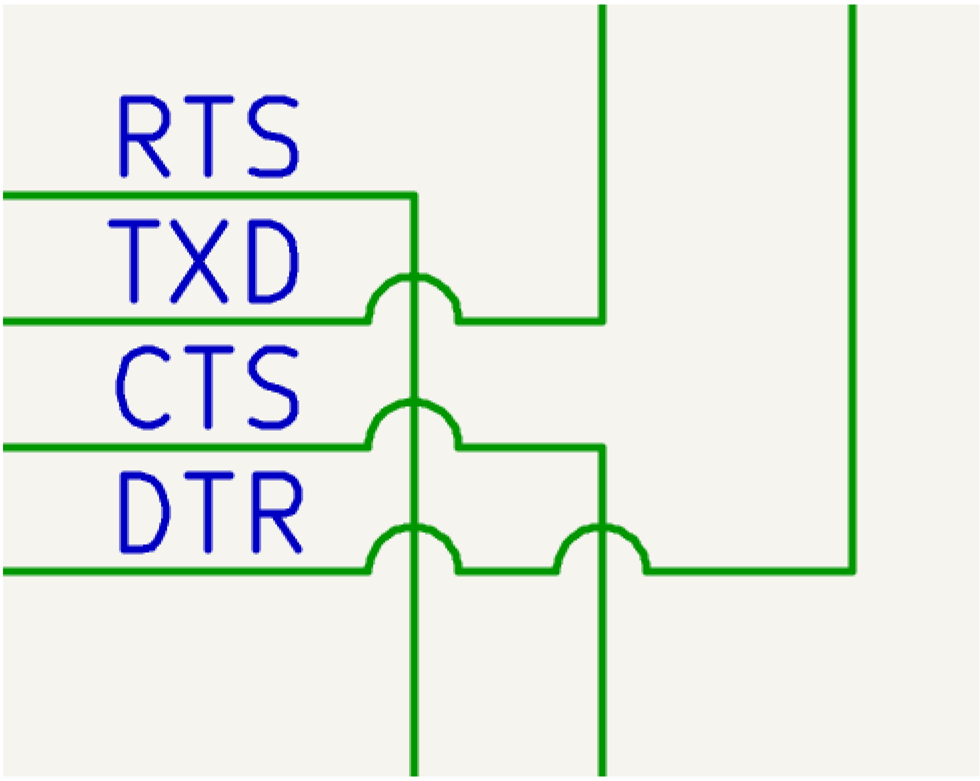
Diameter: 0 mm

Color:

Set diameter to 0 to use schematic's junction dot size.
Clear color to use Schematic Editor colors.

Wire hop-overs

When wires cross without a junction dot, they do not connect. By default, this is drawn simply as one wire crossing through the other wire. However, you can configure such crossovers to be visually indicated with hop-overs instead of simple intersections, as shown in the schematic below. The size of hop-overs is configurable for a given schematic; all hop-overs in the schematic have the same size.



To enable display of hop-overs, open the **Formatting** page of Schematic Setup and change the **Hop-over size** setting from **None** to another size.

Labels

Labels are used to assign net names to wires and pins. Wires with the same net name are considered to be connected, so labels can be used to make connections without drawing direct wire connections.

A net can only have one name. If two different labels are placed on the same net, an ERC violation will be generated. Only one of the net names will be used in the netlist. The final net name is determined according to the rules described below.

There are three types of labels, each with a different connection scope.

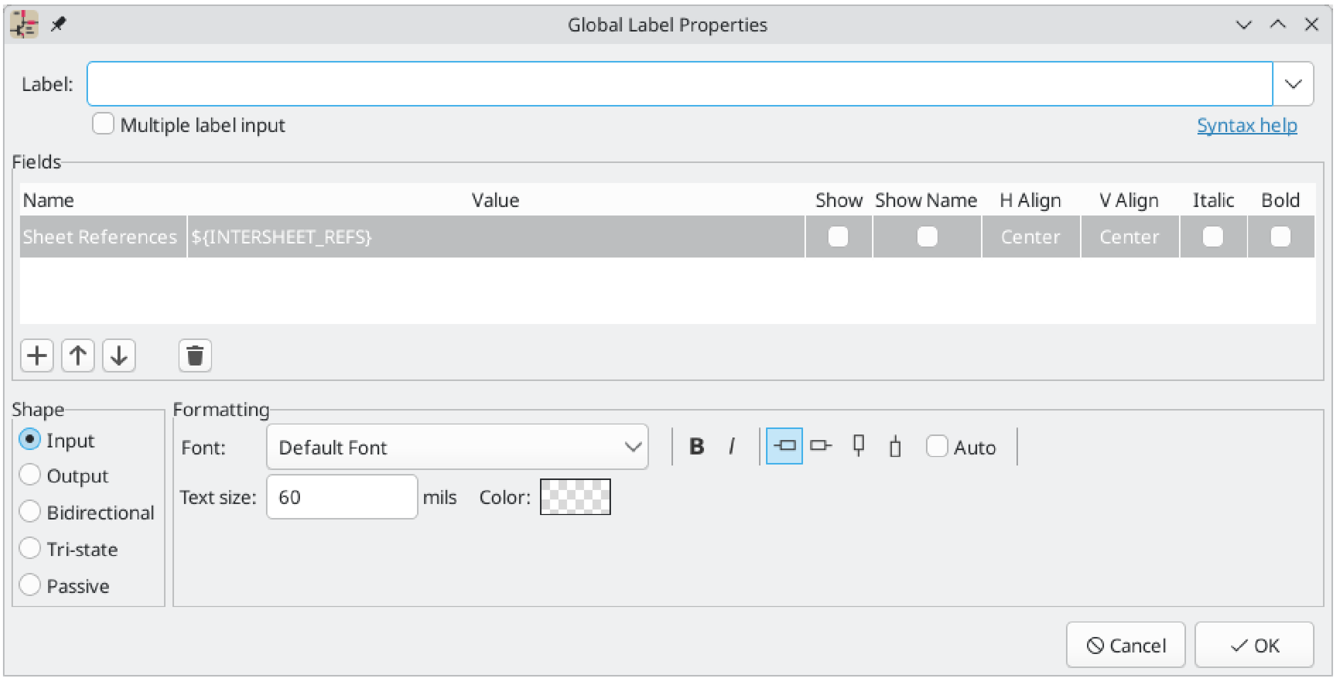
- **Local labels**, also referred to simply as labels, only make connections within a sheet. Add a local label with the A button in the right toolbar.
- **Global labels** make connections anywhere in a schematic, regardless of sheet. Add a global label with the A▷ button in the right toolbar.
- **Hierarchical labels** connect to hierarchical sheet pins and are used in hierarchical schematics for connecting child sheets to their parent sheet. Add a hierarchical label with the A◇ button in the right toolbar.

NOTE Labels that have the same name will connect, regardless of the label type, if they are in the same sheet.

TIP You can convert from one type of label to another type of label using the Change To tools.

Adding and editing labels

After using the appropriate button or hotkey to create a label, the Label Properties dialog appears.



Global Label Properties

Label:

□ Multiple label input

Syntax help

Fields

Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Sheet References	${INTERSHEET_REFS}	□	□	Center	Center	□	□

Shape
- ● Input
- ○ Output
- ○ Bidirectional
- ○ Tri-state
- ○ Passive

Formatting

Font: Default Font

B I □ Auto

Text size: 60 mils Color:

The **Label** field sets the label's name, which determines the net that the label assigns to its attached wire or pin. You can choose a label name from a list of nets that are already in the schematic by clicking the dropdown menu next to the label name field.

NOTE Net names can contain any character, including spaces and other special characters.

Label text supports markup for overbars, subscripts, etc., as well as variable substitution. Use the **Syntax help** link in the dialog for a summary.

When the **Multiple label input** option is enabled, the **Label** field supports entering multiple labels, with one label on each line. In this case, the dialog will create multiple independent labels in sequence, one per line.

TIP Multiple label input may be useful for copying labels from other sources, such as a spreadsheet.

There are several options to control the label's appearance. You can change the font, size, and color of the text, and set bold and italic emphasis. You can also set the orientation of the text relative to the label's connection point. Hierarchical and global labels have several additional options: the **Auto** option automatically sets the label orientation based on the connected schematic elements, and **Shape** option controls the shape of the label outline (**Input**, **Output**, **Bidirectional**, **Tri-state**, or **Passive**). The outline shape is purely visual and has no electrical consequence.

NOTE The default text size can be set for a schematic in Schematic Setup, and the default font can be set in Preferences.

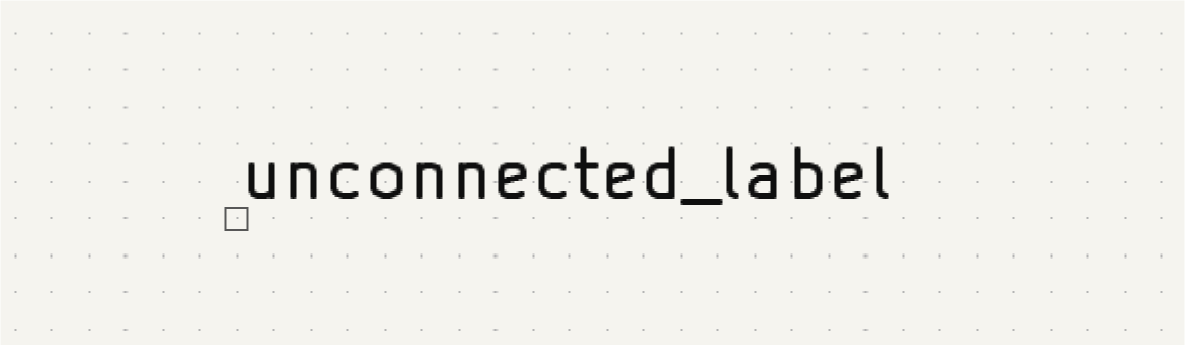
NOTE Global labels have additional settings to control margins around the label text in the Schematic Setup dialog.

Labels can also have fields added to them. Two fields have special meaning (`Net Class` and `Sheet References`, described below), but arbitrary fields can also be added. Label fields behave like symbol fields: you can show or hide their name and value and adjust the alignment, orientation, position, size, font, color, and emphasis.

NOTE Formatting options for label fields can be shown or hidden by right-clicking on the header row of the label field table and enabling or disabling the desired columns. Not all columns are shown by default.

Like symbol fields, label fields can be edited individually by opening the properties of a specific label field from the schematic (double click the label field, or use `E`).

After accepting the label properties, the label is attached to the cursor for placement. The connection point for a label is the small square in the corner of the label. The square disappears when the label is connected to a wire or the end of a pin. If multiple labels were specified in the dialog, each label is attached to the cursor for placement after the previous label is placed.



unconnected_label

The connection point's position relative to the label text can be changed by choosing a different label orientation in the label's properties, or by mirroring/rotating the label.

The Label Properties dialog can be accessed at any time by selecting a label and using the `E` hotkey, double-clicking on the label, or with **Properties...** in the right-click context menu.

Assigning net classes with labels

In addition to assigning net names, labels can be used to assign net classes. A label field named `Net Class` assigns the specified net class to the net associated with the label. To make it easier to assign net classes in this way, `Net Class` is the default name for new label fields, and `Net Class` fields present a dropdown list of all the net classes that have been specified in Schematic Setup or Board Setup.

You can also type in a net class that isn't explicitly listed in the Schematic/Board Setup priority list. Such implicit net classes can't be assigned any design settings, like net class color or track width, but they can still be used in DRC rule queries.

If multiple `Net Class` fields are added to a label, or multiple labels with `Net Class` fields are applied to a net, all of the specified net classes are assigned to the net.

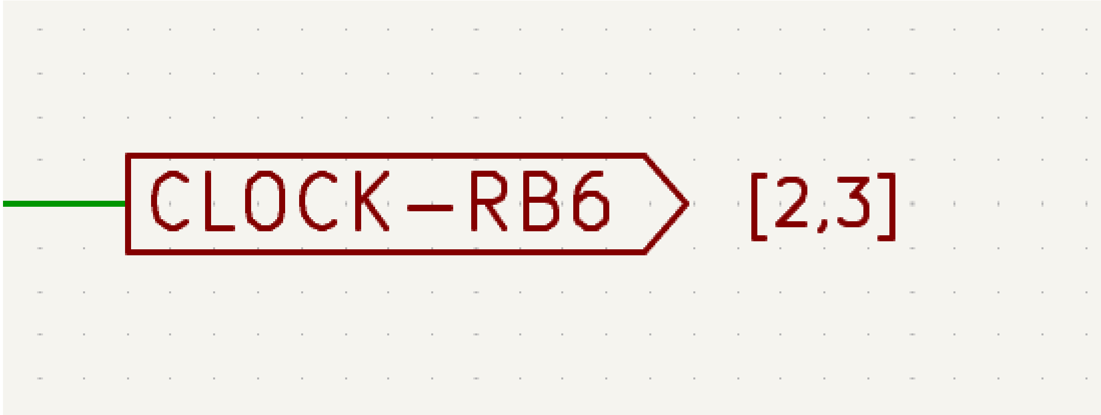
For more information about assigning net classes, see the net class documentation.

Inter-sheet references

Global labels can display inter-sheet references, which are a list of page numbers for other places in the schematic where the same global label appears. Clicking an inter-sheet reference travels to the listed page. If multiple references are listed, clicking the reference list brings up a menu to select the desired page.

Inter-sheet references are globally controlled in the Schematic Setup window's Formatting page. References can be enabled or disabled, and the displayed format for the list can be adjusted, including with optional prefix or suffix characters.

The image below shows a global label with inter-sheet references to two other schematic pages. A prefix and suffix of `[` and `]`, respectively, were added in Schematic Setup.

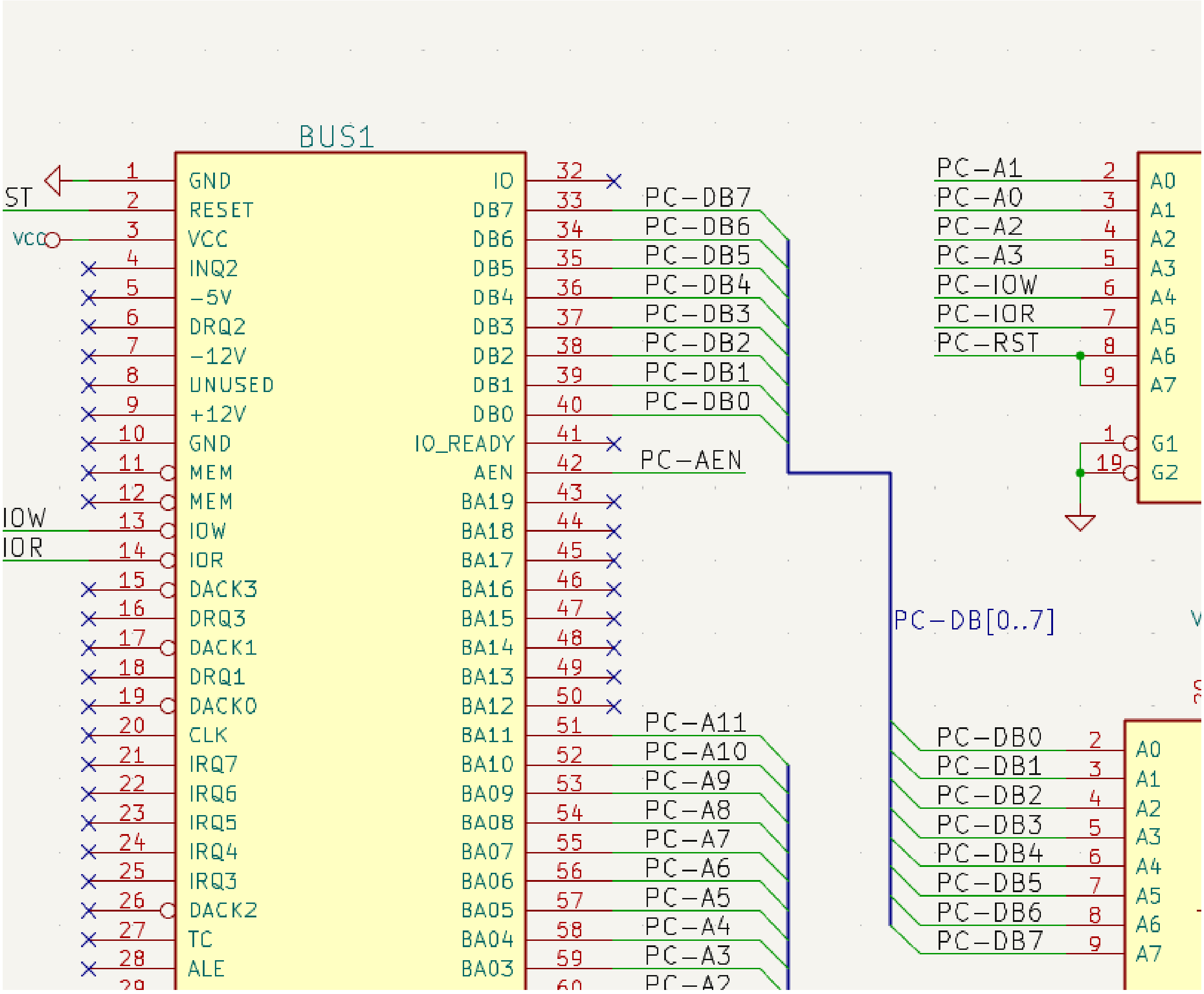


A `Sheet References` field with value `${INTERSHEET_REFS}` is automatically added to global labels, and is used to control the appearance of inter-sheet references for that label. The `${INTERSHEET_REFS}` text variable gets expanded to the full list of inter-sheet references for the global label, as configured in Schematic Setup. Visibility of inter-sheet references is globally controlled in Schematic Setup rather than with the `Sheet References` field visibility control. The `Sheet References` field has no meaning for other types of labels.

Buses

Buses are a way to group related signals in the schematic in order to simplify complicated designs. Buses can be drawn like wires using the bus tool ╱, and are named using labels the same way signal wires are.

In the following schematic, many pins are connected to buses, which are the thick blue lines in the center.



Anggota *Bus*

There are two types of bus in KiCad 6.0 and later: vector buses and group buses.

A **vector bus** is a collection of signals that start with a common prefix and end with a number. Vector buses are named `<PREFIX>[M..N]` where `PREFIX` is any valid signal name, `M` is the first suffix number, and `N` is the last suffix number. For example, the bus `DATA[0..7]` contains the signals `DATA0`, `DATA1`, and so on up to `DATA7`. It doesn't matter which order `M` and `N` are specified in, but both must be non-negative.

Vector bus prefixes cannot contain the characters `{`, `}`, `[`, or `]`, except as part of markup for superscripts, subscripts, overbars, or text variables. Prefixes may contain spaces, but the spaces must be escaped. You can escape spaces in the prefix either by preceding the space with a backslash (`\`) or by surrounding the entire prefix with double quotes (`"`). For example, `DATA\ BUS[0..7]` and `"DATA BUS"[0..7]` are both valid vector bus names.

A **group bus** is a collection of one or more signals and/or vector buses. Group buses can be used to bundle together related signals even when they have different names. Group buses use a special label syntax:

```
<OPTIONAL_NAME>{SIGNAL1 SIGNAL2 SIGNAL3}
```

The members of the group are listed inside curly braces (`{}`) separated by space characters. An optional name for the group goes before the opening curly brace. If the group bus is unnamed, the resulting nets on the PCB will just be the signal names inside the group. If the group bus has a name, the resulting nets will have the name as a prefix, with a period (`.`) separating the prefix from the signal name.

For example, the bus `{SCL SDA}` has two signal members, and in the netlist these signals will be `SCL` and `SDA`. The bus `USB1{DP DM}` will generate nets called `USB1.DP` and `USB1.DM`. For designs with larger buses that are repeated across several similar circuits, using this technique can save time.

Group buses can also contain vector buses. For example, the bus `MEMORY{A[7..0] D[7..0] OE WE}` contains both vector buses and plain signals, and will result in nets such as `MEMORY.A7` and `MEMORY.OE` on the PCB.

Group bus names cannot contain the characters `{`, `}`, `[`, or `]`, except as part of markup for superscripts, subscripts, overbars, or text variables. Group bus names and the constituent net names may contain spaces, but the spaces must be escaped. You can escape spaces either by preceding the space with a backslash (`\`) or by surrounding the entire group bus name or net name with double quotes (`"`). For example, `DATA\ BUS{SIGNAL\ 1 SIGNAL\ 2}`, `"DATA BUS"{"SIGNAL 1" "SIGNAL 2"}`, and `"DATA BUS"{SIGNAL\ 1 "SIGNAL 2"}` are all valid (and equivalent) labels for group buses.

Bus wires can be drawn and connected in the same manner as signal wires, including using junctions to create connections between crossing wires. Like signals, buses cannot have more than one name — if two conflicting labels are attached to the same bus, an ERC violation will be generated.

Koneksi Antar Anggota *Bus*

Pins connected between the same members of a bus must be connected by labels. It is not possible to connect a pin directly to a bus; this type of connection will be ignored by KiCad.

Pada contoh di atas, koneksi dibuat dengan menggunakan label yang diletakkan pada *wire* yang terhubung ke pin. Jalur masuk *bus* (*wire* dengan sudut 45 derajat) hanya bersifat grafis, dan tidak diperlukan untuk membuat koneksi logis.

In fact, using the repetition command (`Insert`), connections can be very quickly made in the following way, if component pins are aligned in increasing order (a common case in practice on components such as memories, microprocessors…):

- Place the first label (for example `PCA0`)
- Use the repetition command as much as needed to place members. KiCad will automatically create the next labels (`PCA1`, `PCA2` …) vertically aligned, theoretically on the position of the other pins.
- Tariklah *wire* di bawah label pertama. Lalu gunakan perintah repetisi untuk meletakkan *wire* berikutnya di bawah label yang sudah dibuat.
- Jika diperlukan, letakkan jalur masuk *bus* dengan cara yang sama (Letakkan jalur masuk yang pertama, lalu gunakan perintah repetisi).

NOTE

In the **Schematic Editor** → **Editing Options** section of the Preferences menu, you can set the repetition parameters:

- Horizontal pitch
- Vertical pitch
- Label increment (labels can be incremented or decremented by 1, 2, 3, etc.)

Bus unfolding

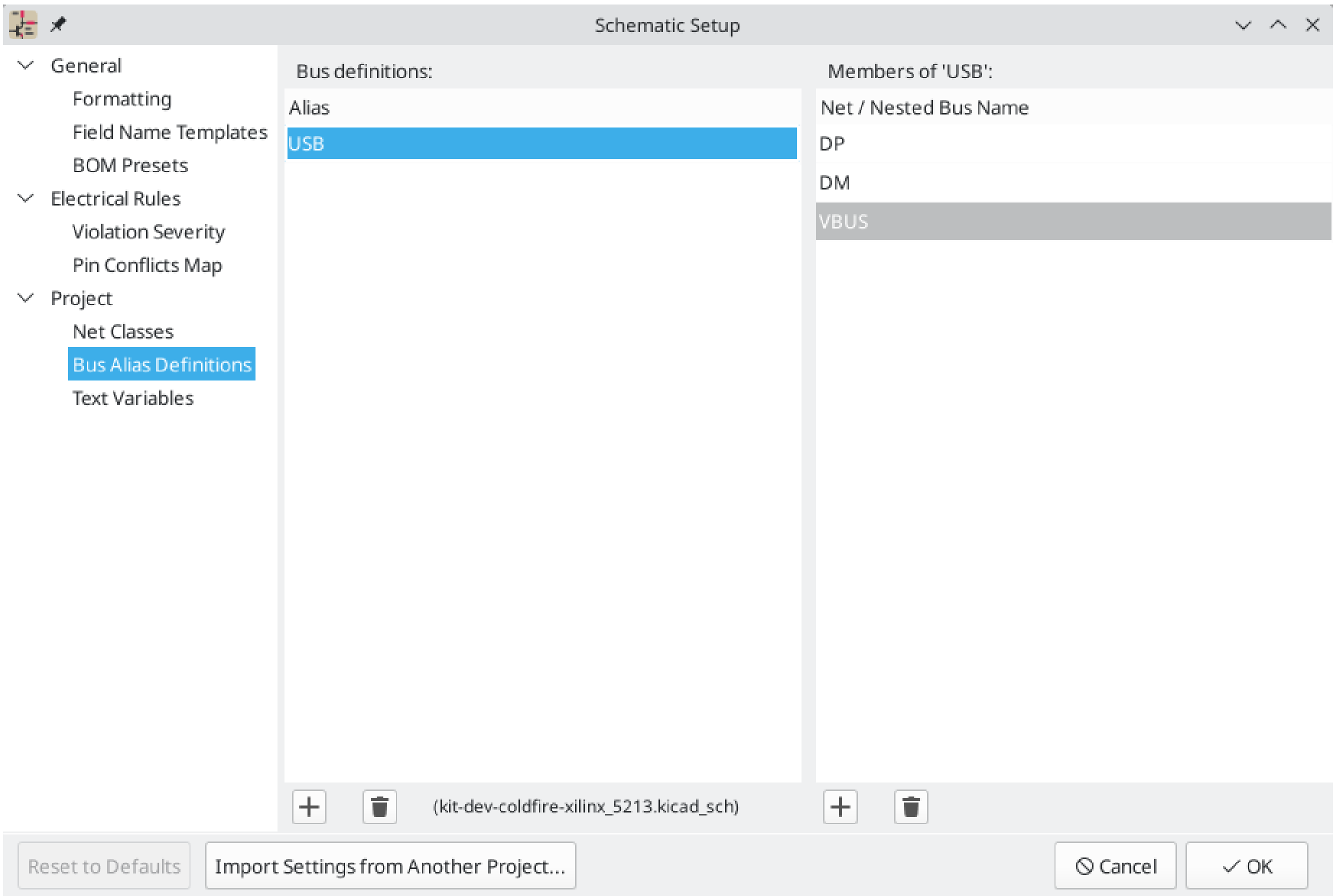
The unfold tool allows you to quickly break out signals from a bus. To unfold a signal, right-click on a bus object (a bus wire, etc) and choose **Unfold from Bus**. Alternatively, use the **Unfold Bus** hotkey (default: `C`) when the cursor is over a bus object. The menu allows you to select which bus member to unfold.

After selecting the bus member, the next click will place the bus member label at the desired location. The tool automatically generates a bus entry and wire leading up to the label location. After placing the label, you can continue placing additional wire segments (for example, to connect to a component pin) and complete the wire in any of the normal ways.

Bus aliases

Bus aliases are shortcuts that allow you to work with large group buses more efficiently. They allow you to define a group bus and give it a short name that can then be used instead of the full group name across the schematic.

To create bus aliases, open the **Bus Alias Definitions** pane in Schematic Setup.



Bus definitions:

Alias
USB

(kit-dev-coldfire-xilinx_5213.kicad_sch)

Members of 'USB':

Net / Nested Bus Name
DP
DM
VBUS

An alias may be named any valid signal name. Using the dialog, you can add signals or vector buses to the alias. As a shortcut, you can type or paste in a list of signals and/or buses separated by spaces, and they will all be added to the alias definition. In this example, we define an alias called `USB` with members `DP`, `DM`, and `VBUS`.

After defining an alias, it can be used in a group bus label by putting the alias name inside the curly braces of the group bus: `{USB}`. This has the same effect as labeling the bus `{DP DM VBUS}`: the nets will be `DP`, `DM`, and `VBUS`. You can also add a prefix name to the group, such as `USB1{USB}`, which results in nets such as

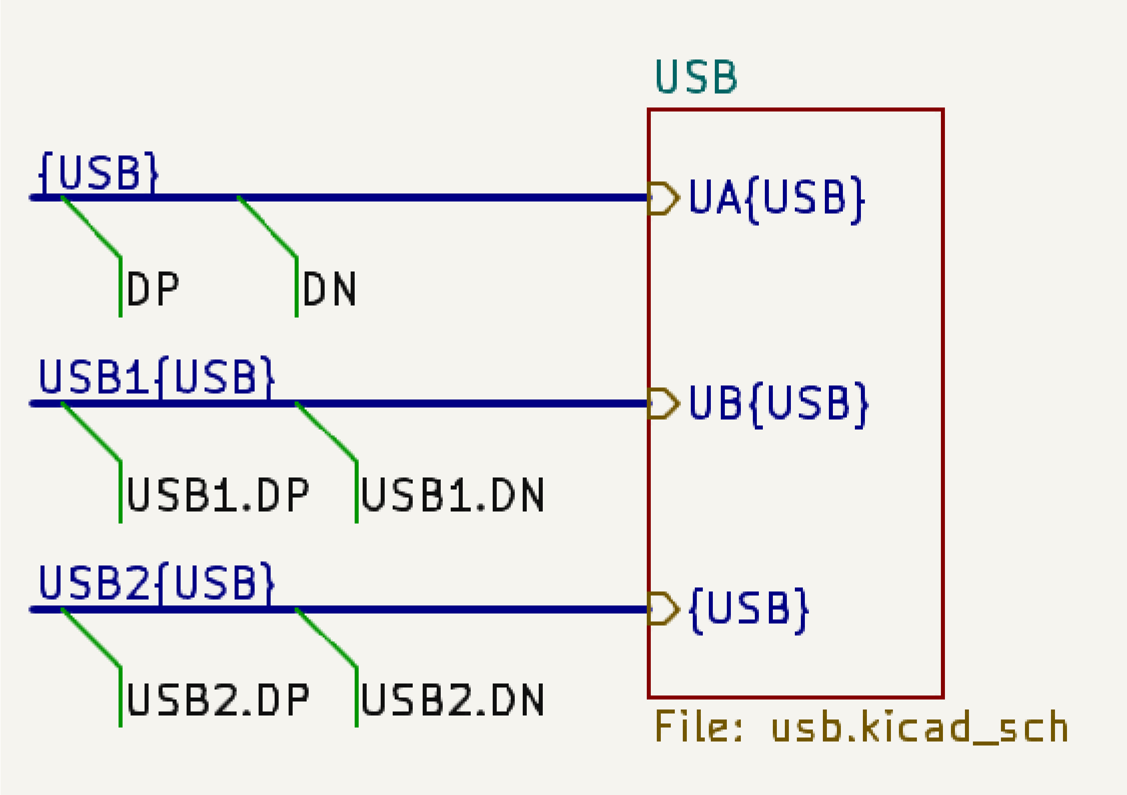
`USB1.DP`. For complicated buses, using aliases can make the labels on your schematic much shorter. Keep in mind that the aliases are just a shortcut, and the name of the alias is not included in the netlist.

Bus aliases are saved in the schematic file that is opened when the alias is created. The **Bus Alias Definitions** window shows the schematic file associated with the selected alias at the bottom of the alias list. Any aliases created in a given schematic sheet are available to use in any other schematic sheet that is in the same hierarchical design. If multiple sheets in a hierarchical design contain identically-named bus aliases, the aliases must all have the same members. ERC will report a violation if multiple bus aliases with the same name do not have consistent members.

Bus example

In the schematic below, a bus alias named `USB` is defined with members `DP`, `DN`.

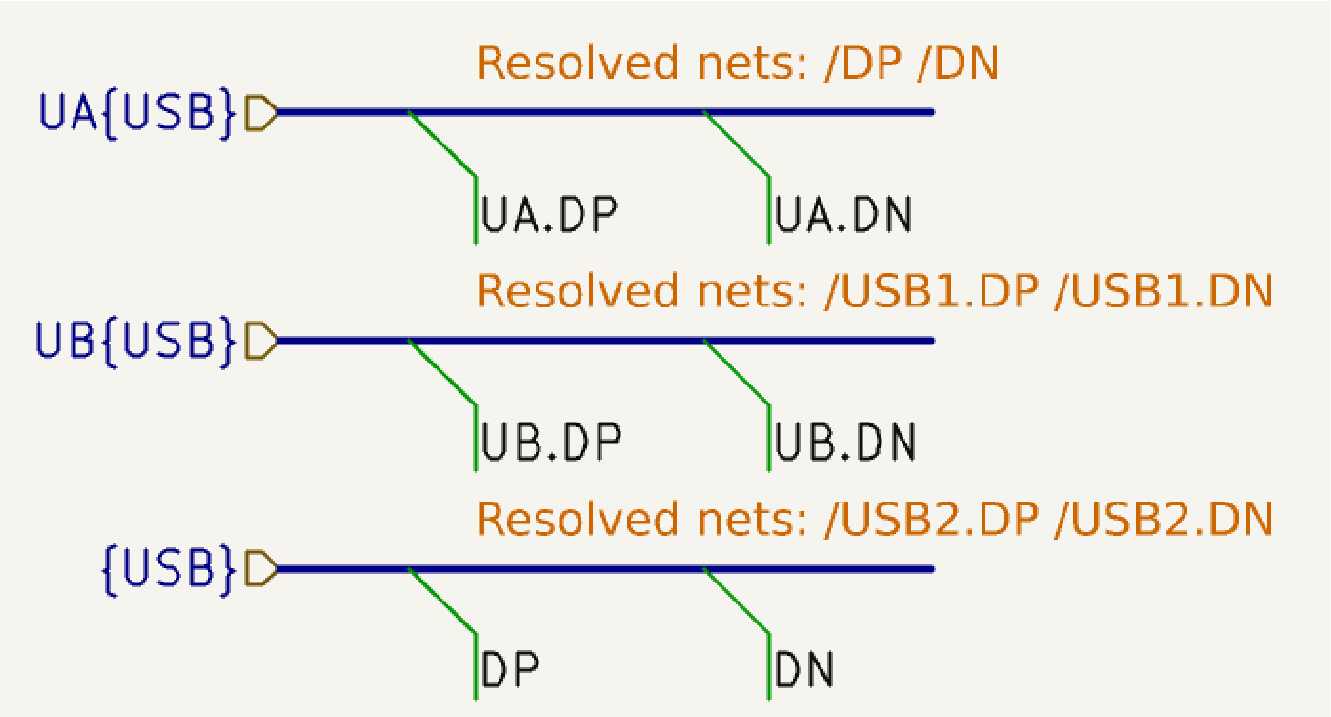
The root sheet has three buses, one with label `{USB}`, with no prefix, and two with prefixes: `USB1{USB}` and `USB2{USB}`. The first bus results in nets `DP` and `DN`, the second results in nets `USB1.DP` and `USB1.DN`, and the third results in nets `USB2.DP` and `USB2.DN`.



When connecting to a sub-sheet, the hierarchical labels (and therefore hierarchical sheet pins) are named with the same syntax. They need the same members as the connected bus. As for bus labels, hierarchical labels can define a prefix or not. In this case, the hierarchical labels are named `{USB}` (i.e. no prefix) and `UA{USB}` and `UB{USB}`.

Within the subsheet, the hierarchical labels for the bus are what define the prefixes of displayed labels of unfolded bus members on that sheet. However, the nets resolve according to the connectivity with the parent sheet.

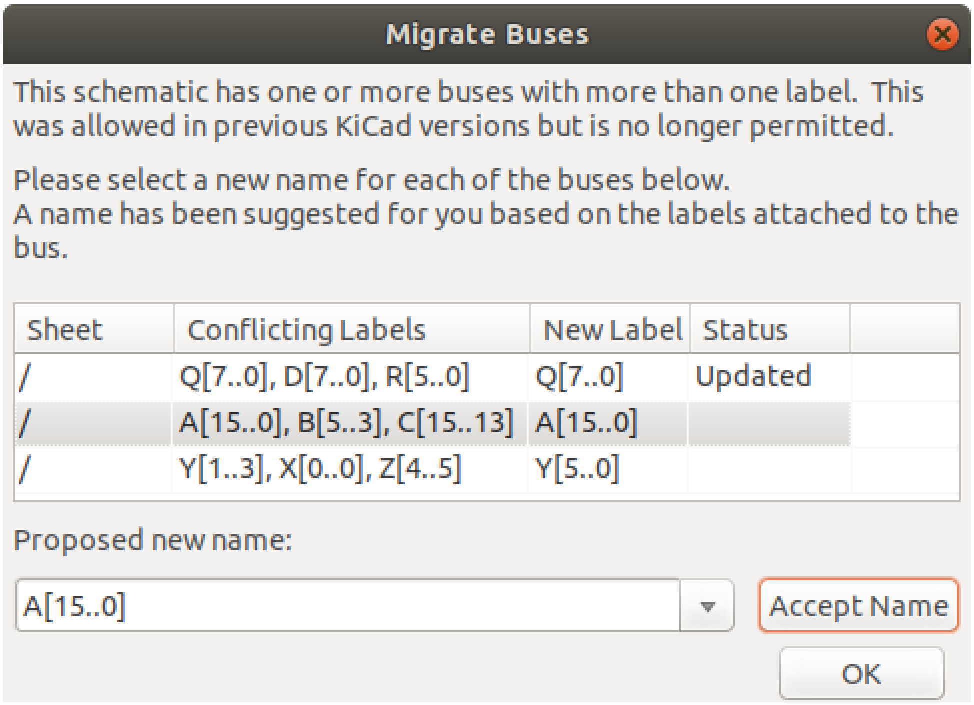
For example, the bus member labeled `UB.DP` in the subsheet has the label prefix `UB` due to the hierarchical label, and the resolved net name `/USB1.DP` due to the name of the bus in the parent sheet that connects to that hierarchical label.



Buses with more than one label

KiCad 5.0 and earlier allowed the connection of bus wires with different labels together, and would join the members of these buses during netlisting. This behavior has been removed in KiCad 6.0 because it is incompatible with group buses, and also leads to confusing netlists because the name that a given signal will receive is not easily predicted.

If you open a design that made use of this feature in a modern version of KiCad, you will see the Migrate Buses dialog which guides you through updating the schematic so that only one label exists on any given set of bus wires.



This schematic has one or more buses with more than one label. This was allowed in previous KiCad versions but is no longer permitted.

Please select a new name for each of the buses below.
A name has been suggested for you based on the labels attached to the bus.

Sheet	Conflicting Labels	New Label	Status
/	Q[7..0], D[7..0], R[5..0]	Q[7..0]	Updated
/	A[15..0], B[5..3], C[15..13]	A[15..0]	
/	Y[1..3], X[0..0], Z[4..5]	Y[5..0]	

Proposed new name: A[15..0]

For each set of bus wires that has more than one label, you must choose the label to keep. The drop-down name box lets you choose between the labels that exist in the design, or you can choose a different name by manually entering it into the new name field.

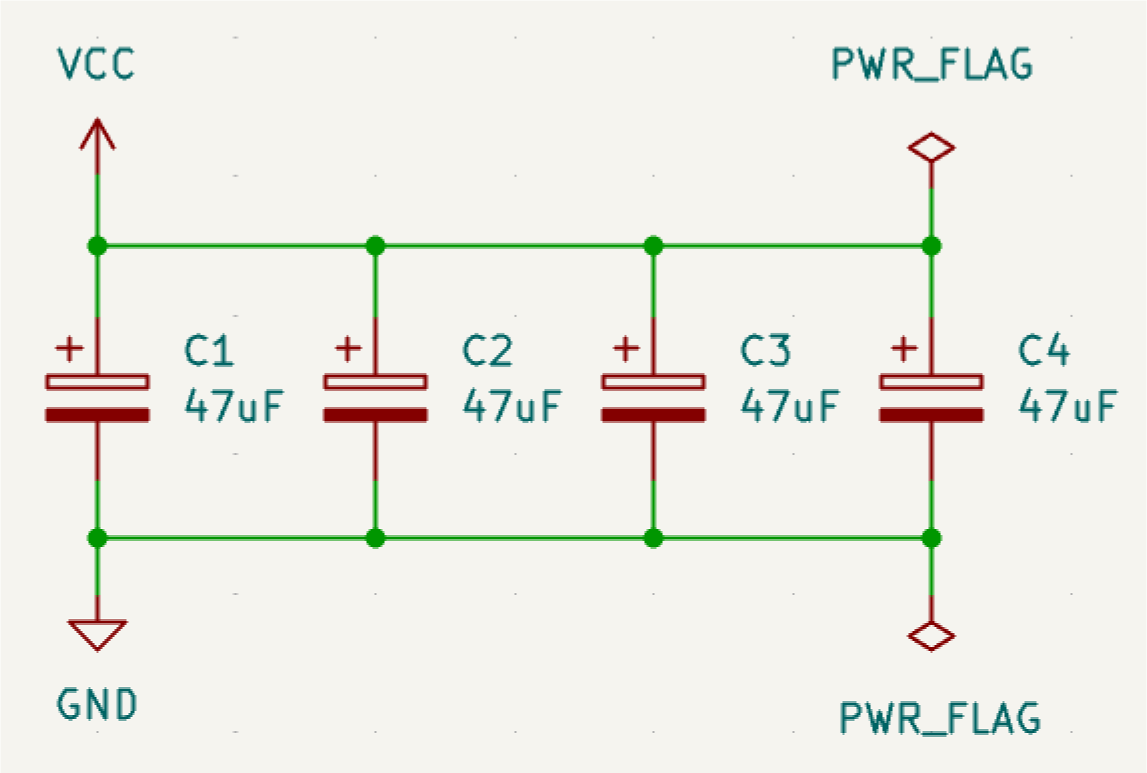
Power symbols

Power symbols are symbols that are conventionally used to represent a connection to a power net, such as `VCC` or `GND`. Power symbols are virtual: they do not represent a physical component on the PCB.

In addition to being a visual indicator that the attached net is a power rail, power symbols make global connections: two power symbols with the `Value` connect to each other anywhere in the schematic, regardless of sheet. The power symbol's `Value` field determines the name of the attached net.

NOTE In previous versions of KiCad, power symbols used invisible power input pins, which make implicit global connections based on the pin name as described below. Beginning in KiCad 8, power symbols do not need to use invisible pins, and the global connection is made based on the power symbol's value.

In the figure below, power symbols are used to connect the positive and negative terminals of the capacitors to the `VCC` and `GND` nets, respectively.



In the KiCad standard library, power symbols are found in the `power` library, but power symbols can be created in any library. Creating custom power symbols is described in the symbol editor documentation. Instead of making a new symbol, you can also modify an existing power symbol in the schematic: changing its `Value` field will change the net the power symbol connects to.

PWR_FLAG

Two `PWR_FLAG` symbols are visible in the screenshot above. They indicate to ERC that the two power nets `VCC` and `GND` are actually connected to a power source, as there is no explicit power source such as a voltage regulator output attached to either net.

Without these two flags, the ERC tool would diagnose: *Error: Input Power pin not driven by any Output Power pins.*

The `PWR_FLAG` symbol is found in the `power` symbol library. The same effect can be achieved by connecting any power output pin to the net.

Hidden power pins

When the power pins of a symbol are visible, they must be connected, as with any other signal. However, symbols are sometimes drawn with hidden power input pins, which are connected implicitly. KiCad automatically connects invisible pins with type Power Input to a global net with the same name as the pin. For example, if a symbol has a hidden power input pin named `VCC`, this pin will be globally connected to the `VCC` net on all sheets. This kind of implicit connection is not recommended in new designs.

WARNING

Care must be taken with hidden power input pins because they can create unintentional connections. By nature, hidden pins are invisible and do not display their pin name. This makes it easy to accidentally connect two power pins to the same net. For this reason, **using invisible power pins in symbols is not recommended** and is only supported for compatibility with legacy designs and symbols.

NOTE

Hidden pins can be shown in the schematic by checking the **Show hidden pins** option in the **Schematic Editor** → **Display Options** section of the preferences, or by selecting **View** → **Show hidden pins**. There is also a toggle icon ▷ on the left toolbar.

No-connection flags

No-connection flags (✕) are used to indicate that a pin is intentionally unconnected. These flags prevent "unconnected pin" ERC warnings for pins that are intentionally unconnected. Also, while symbol pins that are stacked on top of each other are normally connected to the same net, if a no-connection flag is added to the stacked pins they will instead be connected to separate nets.

Note that no-connection flags are distinct from the "unconnected" symbol pin type, although they both prevent "unconnected pin" ERC warnings on the pin in question and prevent stacked pins from connecting to each other.

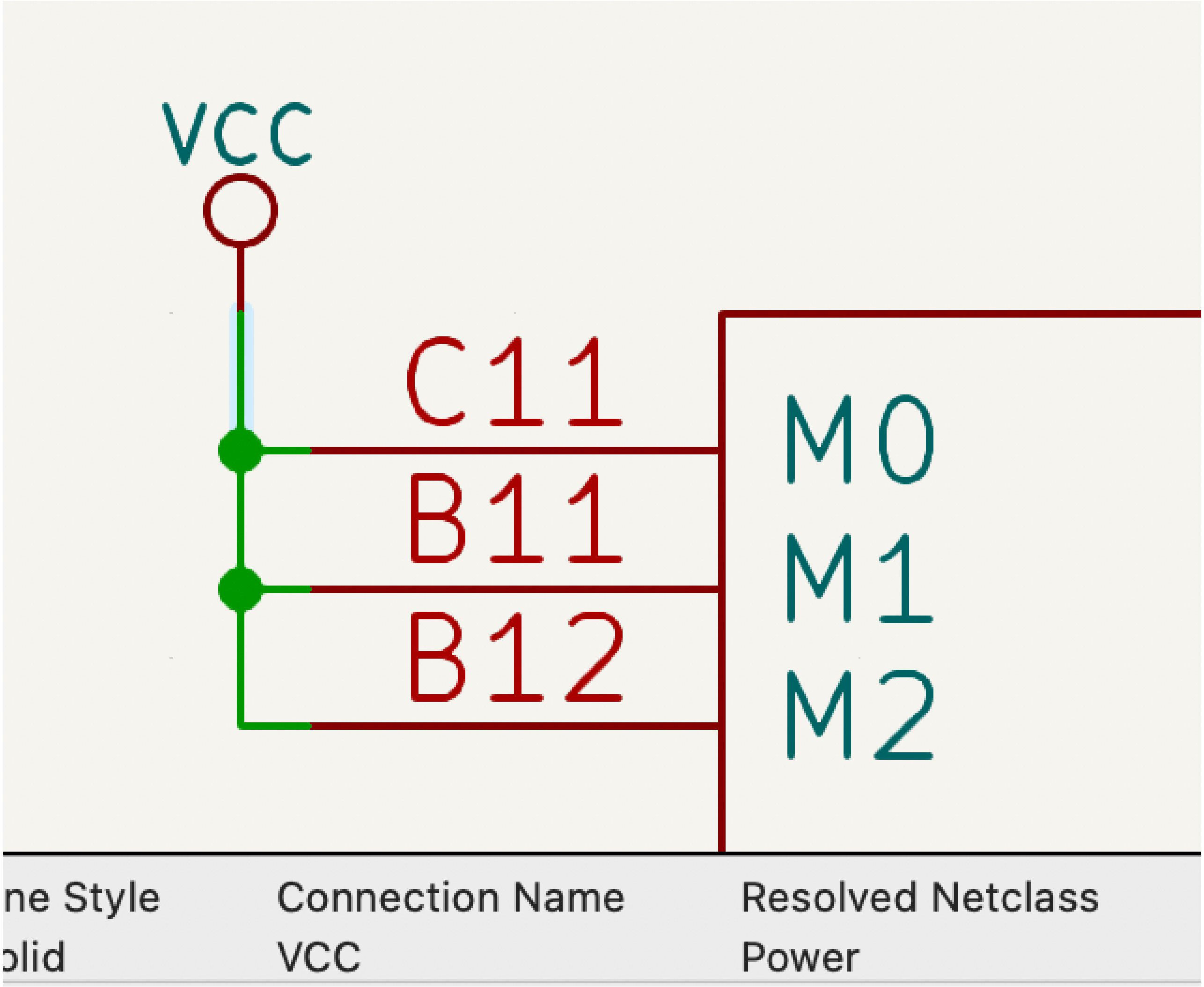
Net classes

Net classes are named groupings of nets that can be assigned design rules (for the PCB) and graphical properties (for the schematic).

More than one net class can be assigned to a net (through a combination of graphical assignments and net class patterns). 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. Nets can be added to net classes in either the schematic or board using pattern-based assignments described below. Nets can also be assigned to net classes in the schematic using graphical assignments with net class directives or net labels.

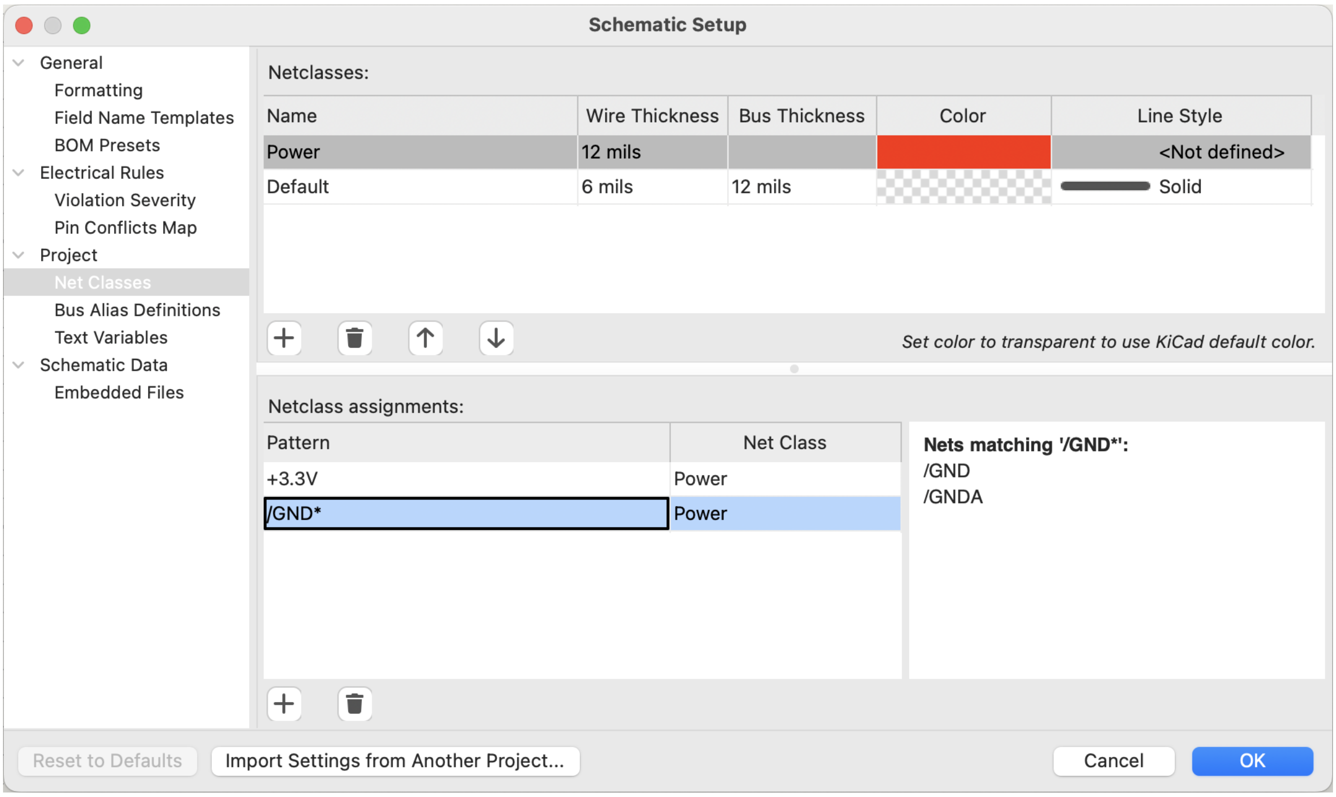
Selecting a wire or label displays the net's net class in the message panel at the bottom of the window.



ne Style	Connection Name	Resolved Netclass
olid	VCC	Power

Managing net classes in Schematic Setup

Net classes are managed in the **Net Classes** panel of the **Schematic Setup** dialog.



Netclasses:

Name	Wire Thickness	Bus Thickness	Color	Line Style
Power	12 mils		🟥	<Not defined>
Default	6 mils	12 mils	▨	Solid

Set color to transparent to use KiCad default color.

Netclass assignments:

Pattern	Net Class
+3.3V	Power
/GND*	Power

Nets matching '/GND*':
/GND
/GNDA

The top pane lists the net classes that exist in the design. The `Default` net class always exists, and you can add additional net classes with the + button or remove the selected net class with the 🗑 button.

Net classes can be moved up and down in priority order with the ↑ and ↓ buttons. Note that the `Default` net class will always be the lowest priority net class and can therefore not be moved.

Each net class can have unique graphic properties that determine how wires of that net class are displayed in the schematic. Wire and bus thicknesses, color, and line style (solid, dashed, dotted, etc.) can all be adjusted. Setting the color to transparent will use the theme's default wire/bus color for the net class, which is configurable in Preferences. By default any color that is configured for a net class controls the color is used to draw wires in that net class. If the **Highlight netclass colors** setting is enabled in the Display Options section of the Schematic Editor preferences, this color will instead be used to draw a highlight around wires in that netclass, and the wires themselves will always be drawn with the color scheme's wire color.

You can also set board design rules for each net class, although the DRC fields are hidden by default. Right click the header row to show or hide additional columns. For more information about setting net class design rules, see the PCB editor documentation.

All net class parameters for user-defined net classes are optional. However, all properties belonging to the `Default` net class must be set. When a net has more than one net class assigned, the appropriate value for graphic properties or board design rules is taken from the highest priority assigned net class with the relevant value set. If only one net class is assigned which contains missing properties, any missing values will be taken from the `Default` net class.

The bottom pane lists pattern-based net class assignments. Each row has a net name pattern and a net class; nets with names that match the pattern are assigned to the specified net class. If a net matches multiple patterns, the net is assigned to all of the matching net classes. You can sort the list of net class assignment patterns by pattern or by net class name by clicking on the corresponding column header.

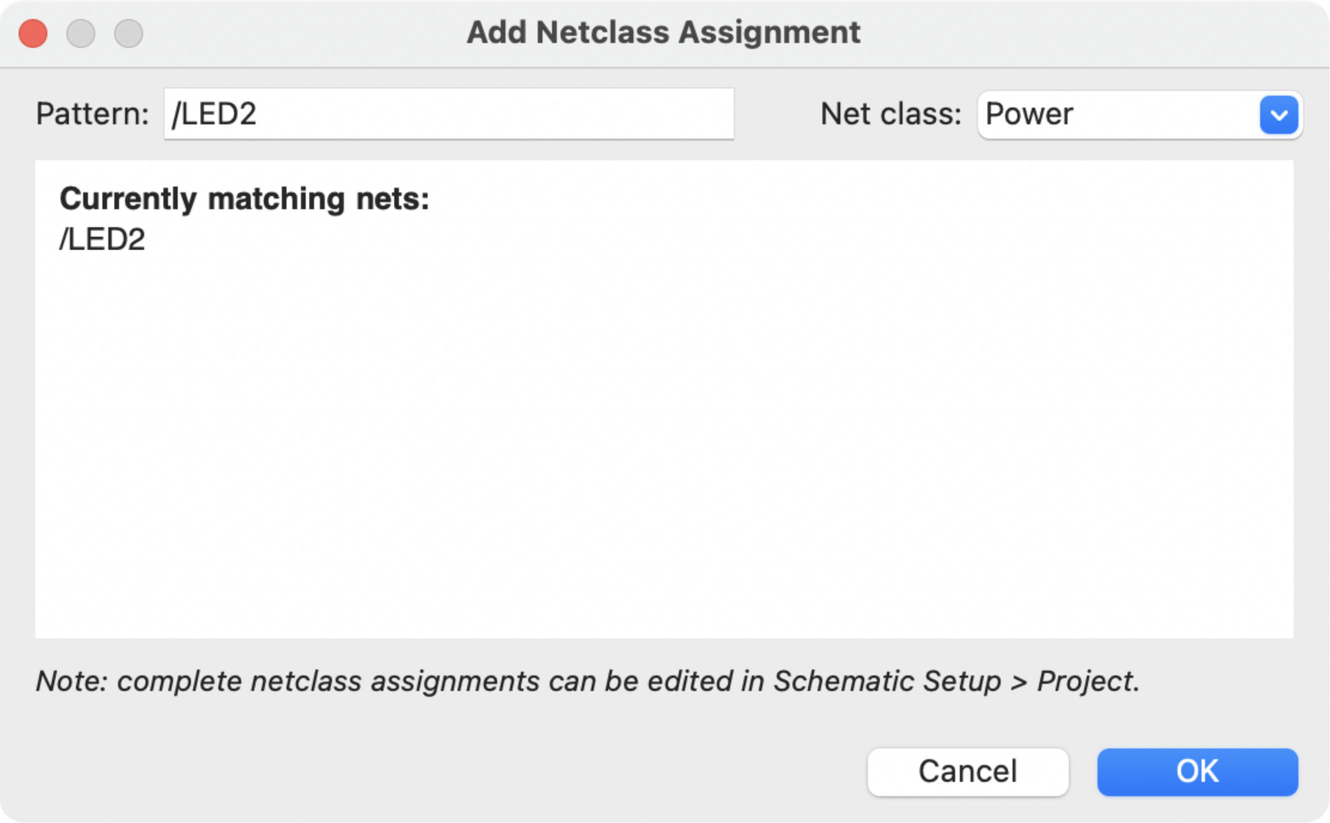
Pattern-based net class assignments are dynamic: when a new net is added that matches an existing pattern, it will be assigned to the associated net class automatically. Net patterns can use both wildcards (`*` to match any number of any characters, including none, and `?` to match any character) and regular expressions. The nets that match the selected pattern are displayed to the right of the pattern list.

For example, the `net*` pattern matches nets named `net`, `net1`, `network`, and any other net name beginning with `net`. Because `*` has a slightly different meaning in a regular expression (`*` matches zero or more of the preceding character), the `net*` pattern would also match a net named `ne`.

> **NOTE** Remember that net names must include the full sheet path. For example, a locally labeled net in the root sheet has a name prefixed with `/`.

Use the + button to add a net class assignment pattern or the 🗑 button to remove a pattern.

Instead of adding net class patterns in the Schematic Setup dialog, you can directly create net class patterns from the schematic canvas. Right click a net and select **Assign Netclass...** to bring up the **Add Netclass Assignment** dialog. The net class pattern is pre-filled with the name of the selected net, but the pattern can be changed if desired. All nets matching the pattern are displayed in the dialog. This method can only be used on nets with an assigned name.



Pattern: /LED2

Net class: Power

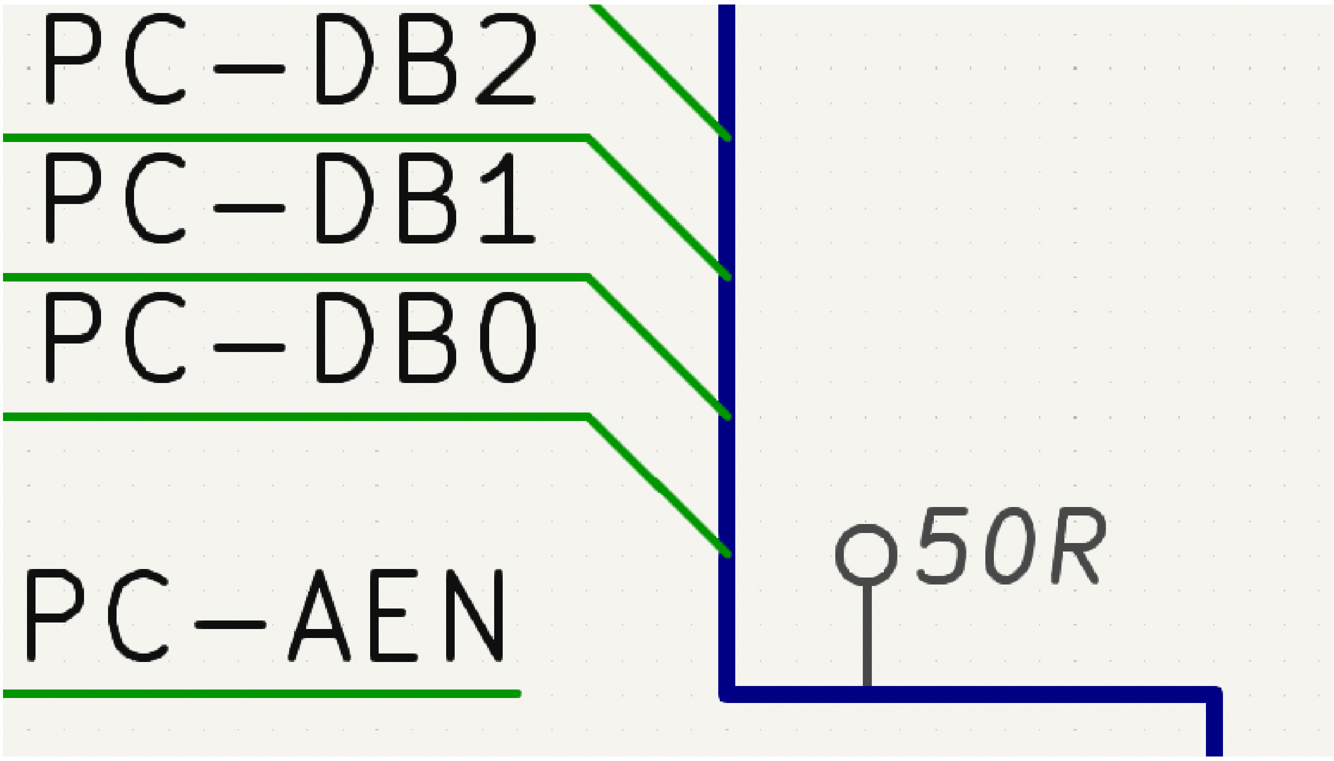
Currently matching nets:
/LED2

Note: complete netclass assignments can be edited in Schematic Setup > Project.

Graphically assigning net classes in the schematic

As an alternative to pattern-based net class assignment, net classes can be graphically assigned to nets in the schematic using either **directive labels**, **net labels**, or **rule areas**.

In the image below, a directive label is used to assign signals to the `50R` net class.

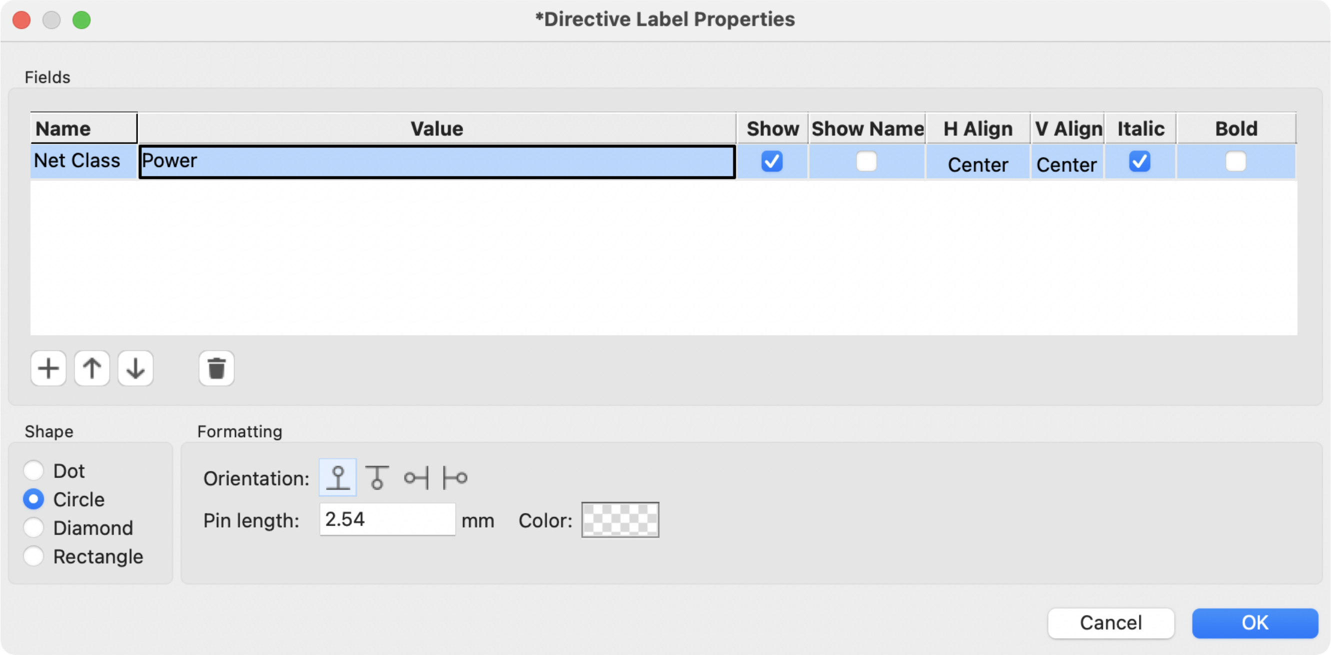


Directive labels are added with the ⚲A button in the right toolbar. They behave like labels, except that they cannot be used to name a net. The attached net is assigned a net class according to the value of the directive's `Net Class` field. The `Net Class` field presents a dropdown list of all the net classes that have been specified in Schematic Setup or Board Setup.

You can also type in a net class that isn't explicitly listed in the Schematic/Board Setup priority list. Such implicit net classes can't be assigned any design settings, like net class color or track width, but they can still be used in DRC rule queries.

If multiple `Net Class` fields are added to a directive label, or multiple directive labels with `Net Class` fields are applied to a net, all of the specified net classes are assigned to the net.

If a directive is attached to a bus, all members of the bus are assigned to the specified net class.



*Directive Label Properties

Fields

Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Net Class	Power	☑	☐	Center	Center	☑	☐

Shape

○ Dot
● Circle
○ Diamond
○ Rectangle

Formatting

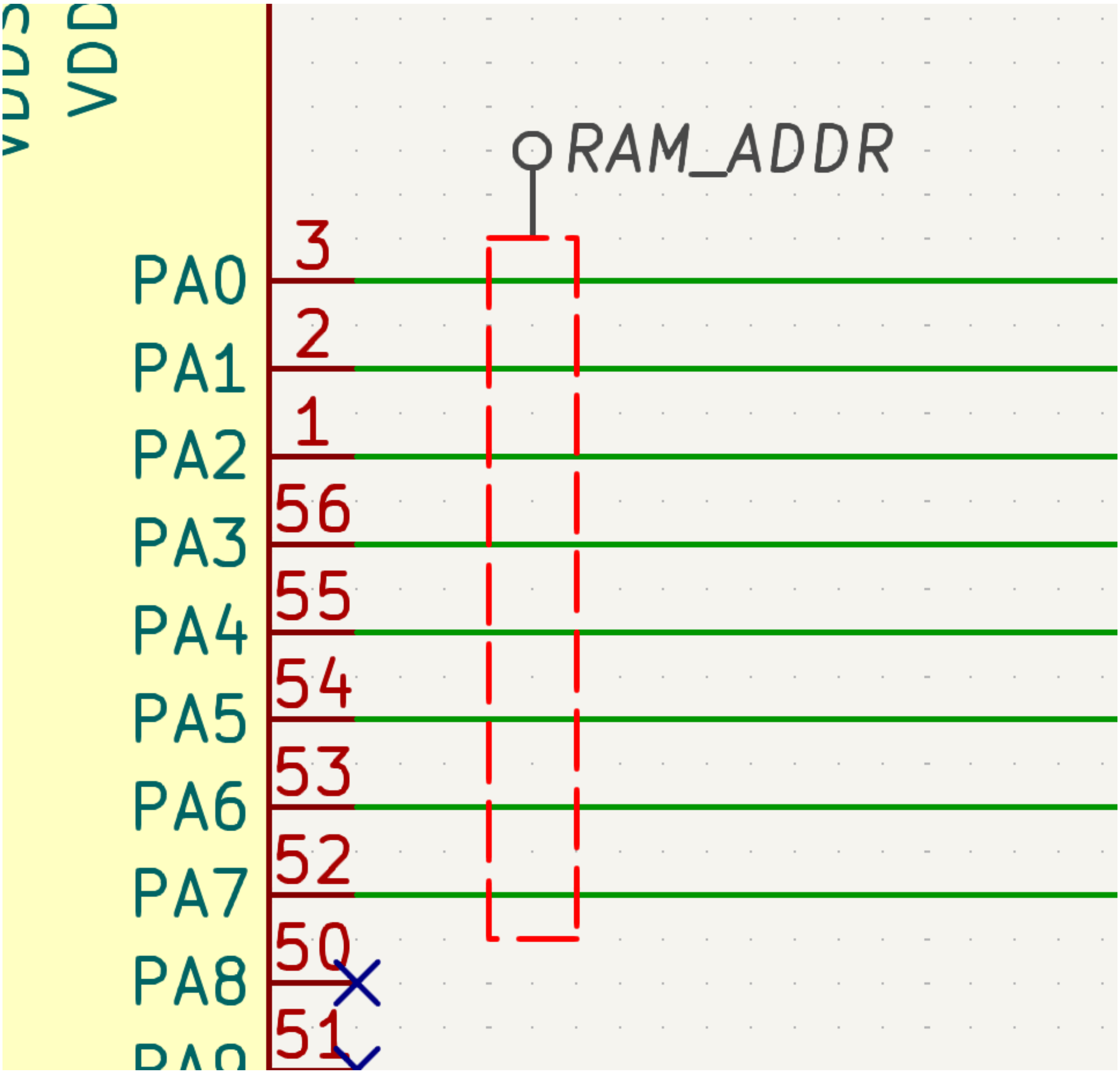
Orientation:

Pin length: 2.54 mm Color:

In addition to the associated net class, you can edit the directive's **shape** (dot, circle, diamond, or rectangle), **orientation**, **pin length**, and **color** in the directive's properties.

> **NOTE** Net labels can also be used to assign net classes to nets by adding a `Net Class` field to the label.

The Rule Area tool (▨) can be used to draw a shape to which net class directives can be attached. Any wire, bus, label, or symbol pin which crosses or is inside the rule area will be assigned the net class of a net class directive attached to the rule area border. An example is shown in the image below; all wires passing through the rule area will be assigned the `RAM_ADDR` net class.



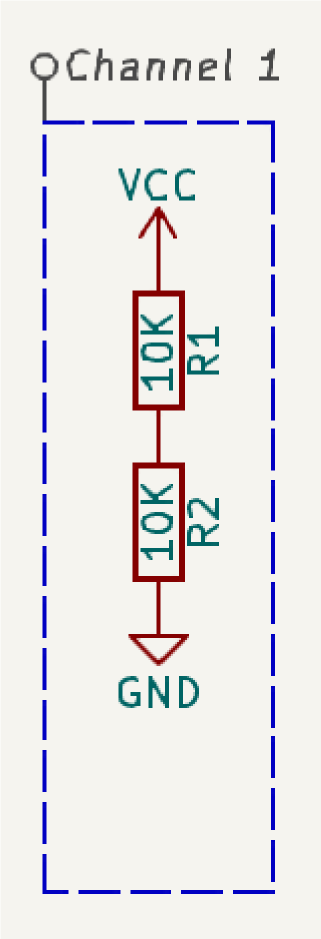
You can show or hide directive labels in the schematic using the **View** → **Show Directive Labels** option.

Component classes

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.

To assign a component class to a symbol, you can add a symbol field named `Component Class` to the symbol. The symbol will then be a member of the component class named by the field.

You can also assign component classes using directive labels (⚲A) in combination with rule areas (▨). The Rule Area tool can be used to draw a shape to which directive labels can be attached. Any symbol which crosses or is inside the rule area will be assigned to the component class specified by the directive label attached to the rule area border. An example is shown in the image below; R1 and R2 will be assigned to the `Channel 1` component class.



Component classes can also be assigned in the PCB Editor by creating rules for automatic component class assignment.

Components can have more than one class, and symbols take on a class if any of their sub-units have that class. If multiple `Component Class` fields are added to a directive label, or multiple directive labels with `Component Class` fields are applied to a rule area, the symbols in the rule area will take on all of the specified component classes.

Rule areas

Rule areas are regions of the schematic that apply certain properties to the symbols and nets inside the area.

You can use rule areas to apply any of the following attributes to the symbols inside the rule area:

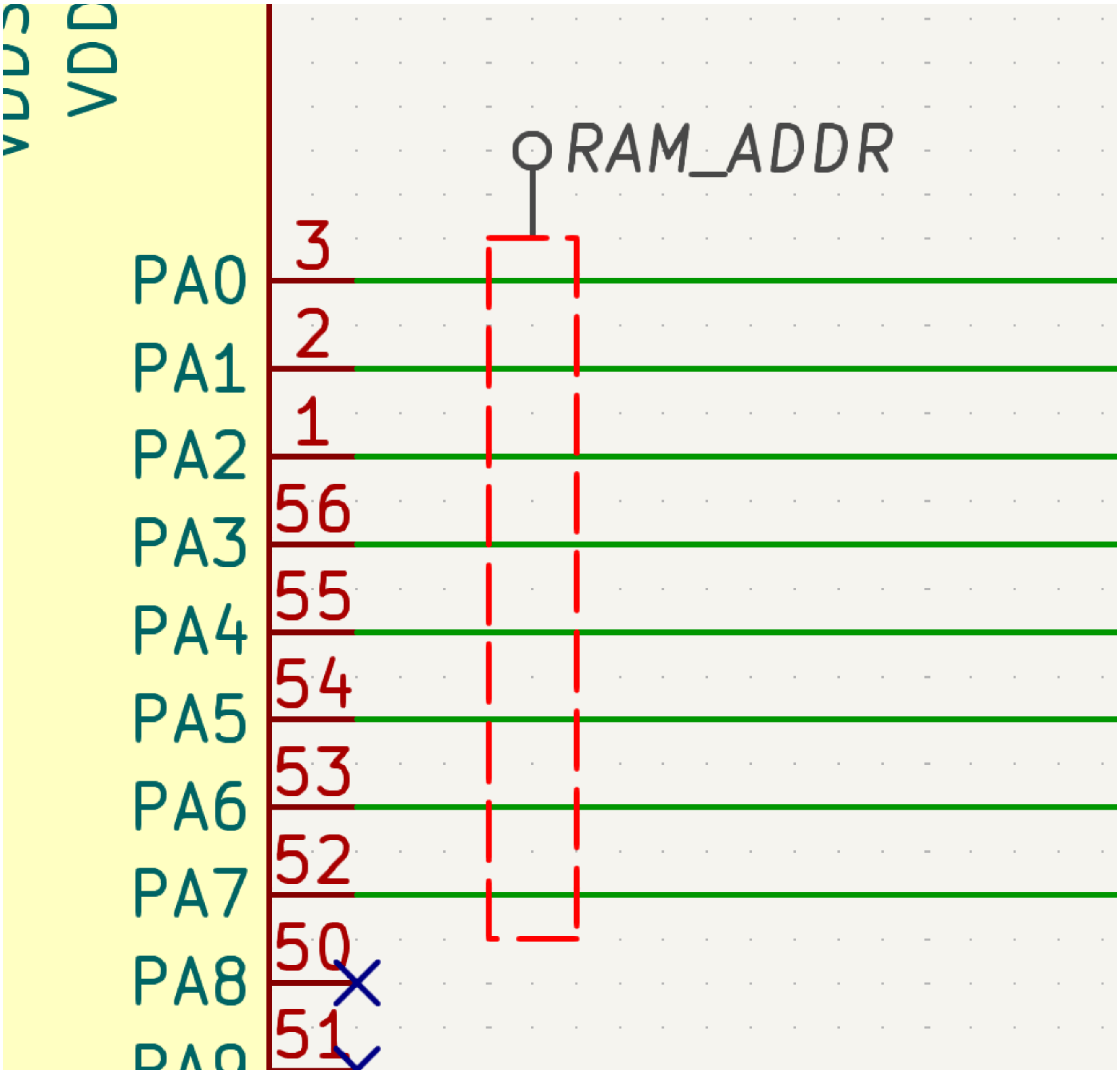
- Exclude from simulation

- Exclude from bill of materials
- Exclude from board
- Do not populate

A rule area that has any of those attributes enabled will apply the enabled attributes to all symbols that are fully or partially enclosed by the rule area.

You can also use a rule area, in combination with a directive label, to apply net classes and component classes to symbols and nets inside the rule area. Any wire, bus, label, or symbol pin which crosses or is inside the rule area will be assigned the net class specified by any directive labels attached to the rule area border. Any symbols that are fully or partially enclosed by the rule area will be assigned the component class specified by any attached directive labels.

In the image below, all wires passing through the rule area will be assigned the `RAM_ADDR` net class.



To draw a rule area, use the Rule Area tool (▨ button).

Click in the canvas to begin drawing the rule area outline.

2. Click to place additional corners.

3. Double click to complete the outline.

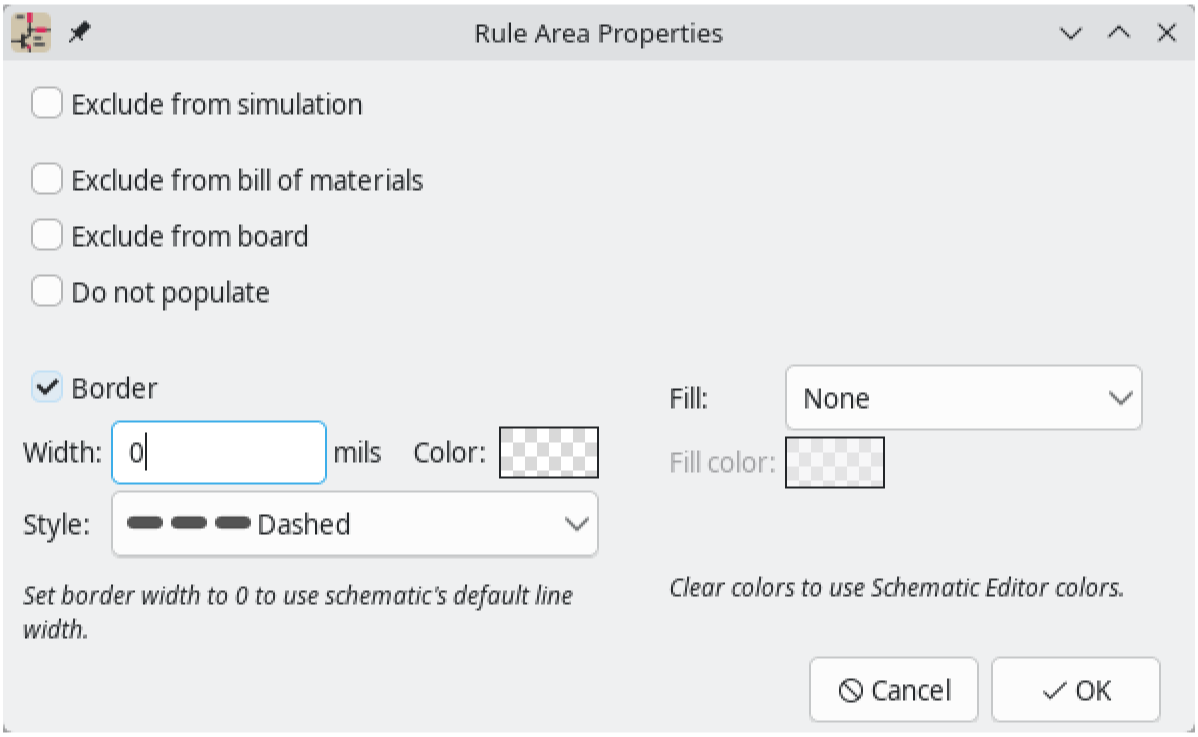
The rule area tool obeys the selected line mode: the outline is restricted to horizontal, vertical, and 45 degree lines unless free angle mode is enabled.

> **NOTE** You can switch line modes using `Shift` + `Space` or using the / / flyout button group in the left toolbar.

To use a rule area to assign a net class or component class, you must add a directive label to the rule area.

1. Activate the directive label tool (button on the right toolbar). By default, the directive label tool is grouped in a palette with the other label tools and therefore may be hidden.
2. In the directive label's properties dialog, set the label's `Net Class` and/or `Component Class` fields to the desired net class or component class, respectively.
3. Place the directive label in the canvas so that the label's connection point is on the border of the rule area, as shown in the screenshot above.

To enable attributes (exclude from simulation, exclude from BOM, exclude from board, or DNP) for the rule area, open the rule area's properties dialog (`E`). Any attributes that are enabled in the rule area's properties are applied to symbols that are fully or partially enclosed by the rule area.



Rule Area Properties

☐ Exclude from simulation

☐ Exclude from bill of materials

☐ Exclude from board

☐ Do not populate

☑ Border

Width: 0 mils Color:

Style: Dashed

Set border width to 0 to use schematic's default line width.

Fill: None

Fill color:

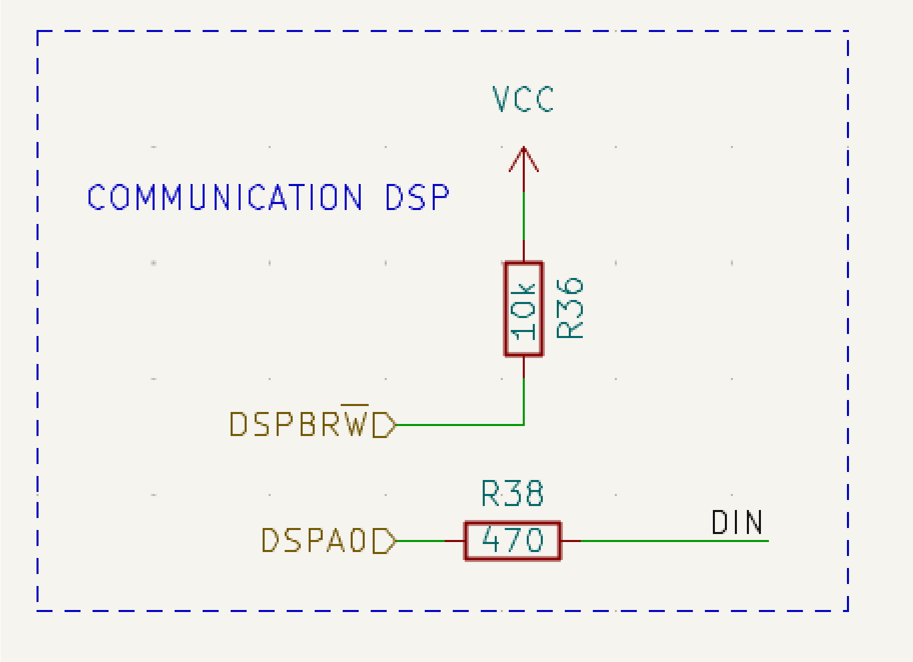
Clear colors to use Schematic Editor colors.

You can also configure the graphical style of the rule area's border and fill here. The border can be set to any color or line width, and various line styles (solid, dashed, dotted, etc.) are available. If the line width is set to `0`, the default line width from the **Formatting** page of Schematic Setup is used. The fill can be set to **None** (default), **Solid**, or one of several types of hatching. You can also configure the fill color. If either the border or fill color is not set, that color is taken from the Rule Areas color in the color theme.

Graphics and text

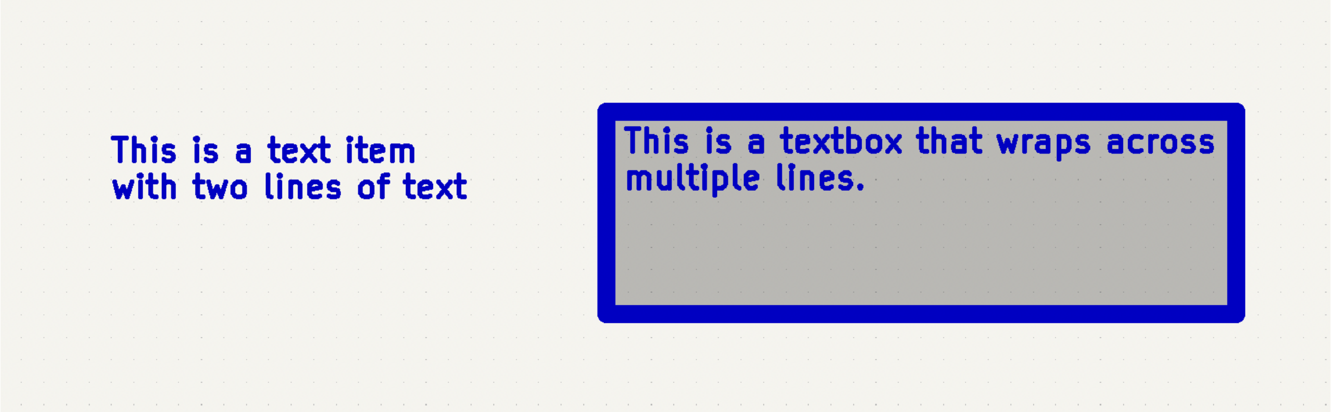
Text, graphic shapes, and images can be added to schematics for documentation purposes. These items do not have any electrical effect on the schematic.

The image below shows graphic lines and text ("COMMUNICATION DSP") in addition to symbols and several types of labels.



Text and text boxes

Two kinds of text can be added to schematics, which are referred to as text (T) and text boxes (▤). Both are added using their respective buttons in the right toolbar. Text boxes are similar to regular text except that they have an optional border and they automatically reflow text within that border.

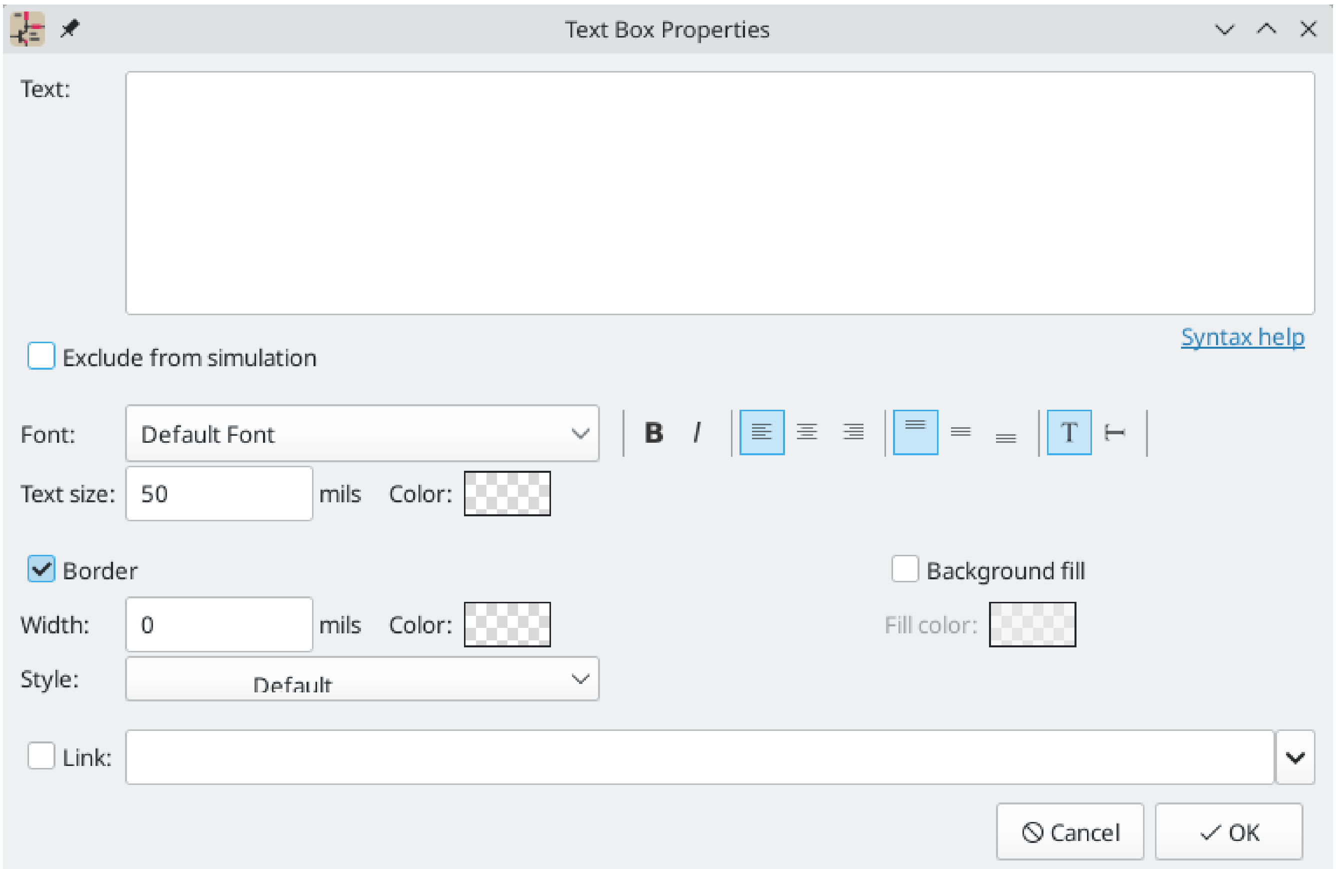


This is a text item
with two lines of text

This is a textbox that wraps across
multiple lines.

Both kinds of text item support multiline text and basic formatting features, but text boxes wrap text to fit in the outline and have additional formatting options. All text has adjustable fonts, color, size, bold and italic emphasis, left and right alignment, and vertical and horizontal orientation. Text boxes additionally support horizontal centering, vertical alignment options, and colored borders and fill. You can also adjust the padding on each side of text in a text box (padding can be set using the Properties Manager, but not using the Text Box Properties dialog).

NOTE The default text size can be set for a schematic in Schematic Setup, and the default font can be set in Preferences.



Text Box Properties

Text:

□ Exclude from simulation

Font: Default Font

Text size: 50 mils Color:

☑ Border

Width: 0 mils Color:

Style: Default

□ Background fill

Fill color:

□ Link:

Links

In a text object, text box, or table cell, any text beginning with `http://` or `https://` is treated as a URL and automatically turned into a link. Clicking on such a link opens the URL in your web browser. If the text contains multiple URLs, each URL is turned into a separate link.

Alternatively, you can explicitly turn an entire text object or text box into a link by entering a target in the **Link** box in the text properties. This turns the entire text item or text box into a clickable link. Such text can only link to a single URL, but the displayed text can be anything, rather than simply the URL.

You can link to different kinds of resources depending on the link target. The target of an explicit link can be:

- a sheet in the current schematic, using `#` followed by the page number
- a local file on your machine, using a URL with the `file://` scheme
- a website, using a URL with the `http://` or `https://` scheme
- another resource, using a URL with the appropriate scheme, e.g. `ftp://`

If no protocol prefix is used, the target is assumed to be a local file as if the `file://` scheme was used.

Sheet, file, and web links can be autofilled using the dropdown meu in the link target box. Other kinds of links cannot be autofilled but will work if your system can handle them.

Fonts

Text and text boxes support custom fonts, which are selectable with the **Font** dropdown in the properties dialog for the text. In addition to the KiCad font, you can use any TTF font installed on your computer.

> **NOTE** User fonts are not embedded 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. For maximum compatibility, use the KiCad font.

Text markup

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

Feature	Markup Syntax	Result
Superscript	`text^{superscript}`	text$^{\text{superscript}}$
Subscript	`text_{subscript}`	text$_{\text{subscript}}$
Overbar	`~{text}`	$\overline{\text{text}}$
Project text variables	`${variable}`	*variable_value*
Built-in text variables	`${refdes:field}`	*field_value* of symbol *refdes*

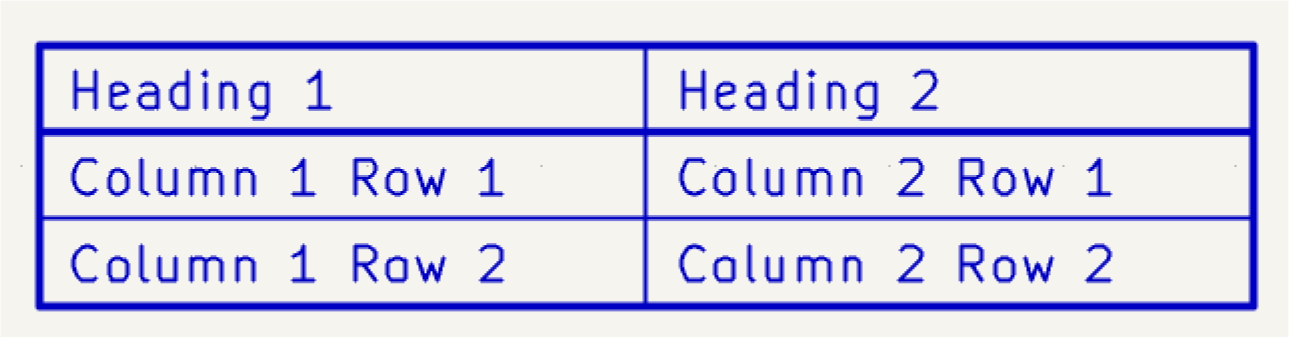
> **NOTE** Project text variables must be defined in Schematic Setup before they can be used. There are also a number of built-in system text variables.

Simulation directives

Text and textboxes can contain simulation directives for SPICE simulations. The **Exclude from simulation** checkbox prevents text from being interpreted as a simulation directive.

Tables

You can use a table to organize text in a tabular format. Tables have customizable borders, cell sizes, colors, and headers.



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.

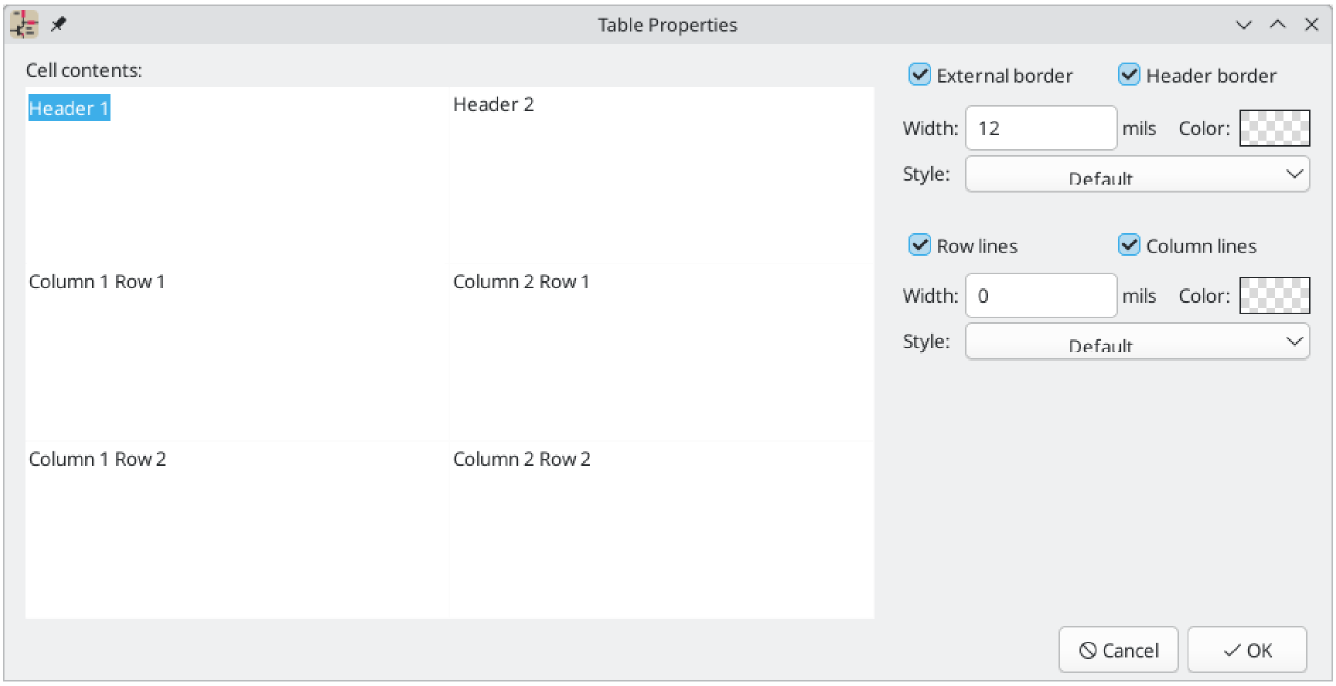
Once you have created a table, you can edit the table as a whole or edit cells individually. Creating a new table automatically opens the Table Properties dialog to edit the entire table.

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.



Cell contents:

Header 1	Header 2
Column 1 Row 1	Column 2 Row 1
Column 1 Row 2	Column 2 Row 2

☑ External border ☑ Header border
Width: 12 mils Color:
Style: Default

☑ Row lines ☑ Column lines
Width: 0 mils Color:
Style: Default

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 markup section (superscripts, subscripts, strikethroughs, etc.).

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

- 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. When set to 0, the line width uses the default symbol line width configured in the **Formatting** panel of Schematic Setup. The line color is controlled by the **Color** picker, and 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...** (E). 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.

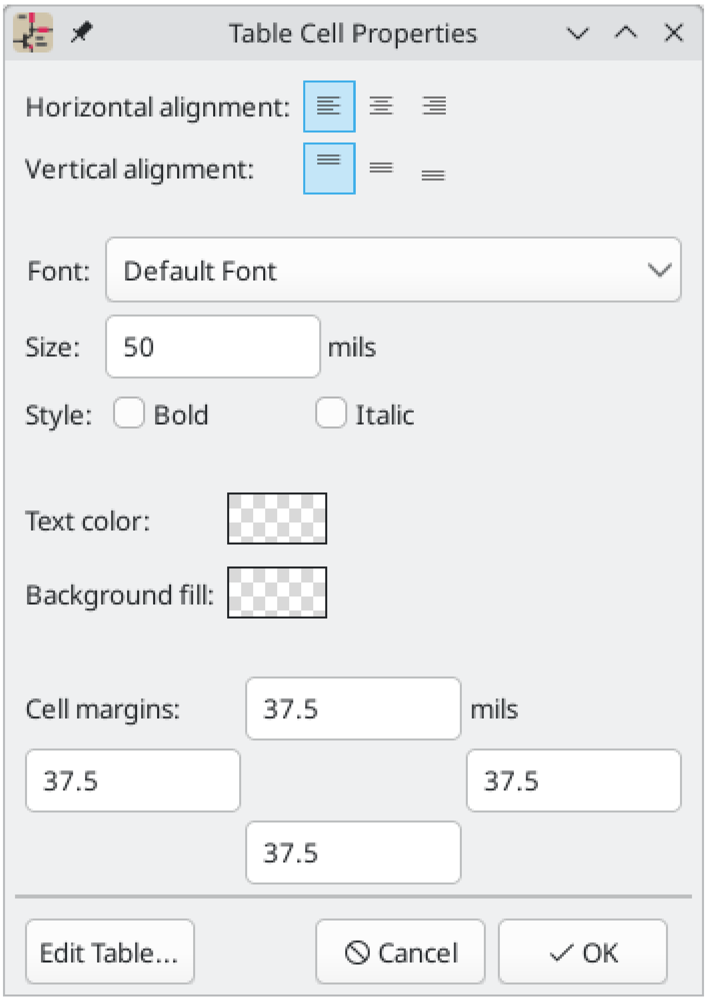


Table Cell Properties

Font: Default Font

Size: 50 mils

Style: □ Bold □ Italic

Cell margins: 37.5 mils

37.5

37.5

37.5

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.

- **Horizontal alignment** and **Vertical alignment** control how text is positioned within the 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.
- **Size** controls the size of the text in the cell.
- The **Bold** and **Italic** checkboxes bold and italicize the text, respectively. These are three-state checkboxes, which can be set to off, on, or no change. No change is useful when multiple cells with different bold/italic settings are being edited at the same time.
- The **Text color** and **Background fill** color pickers control the color of the text and the cell background, respectively.
- 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.

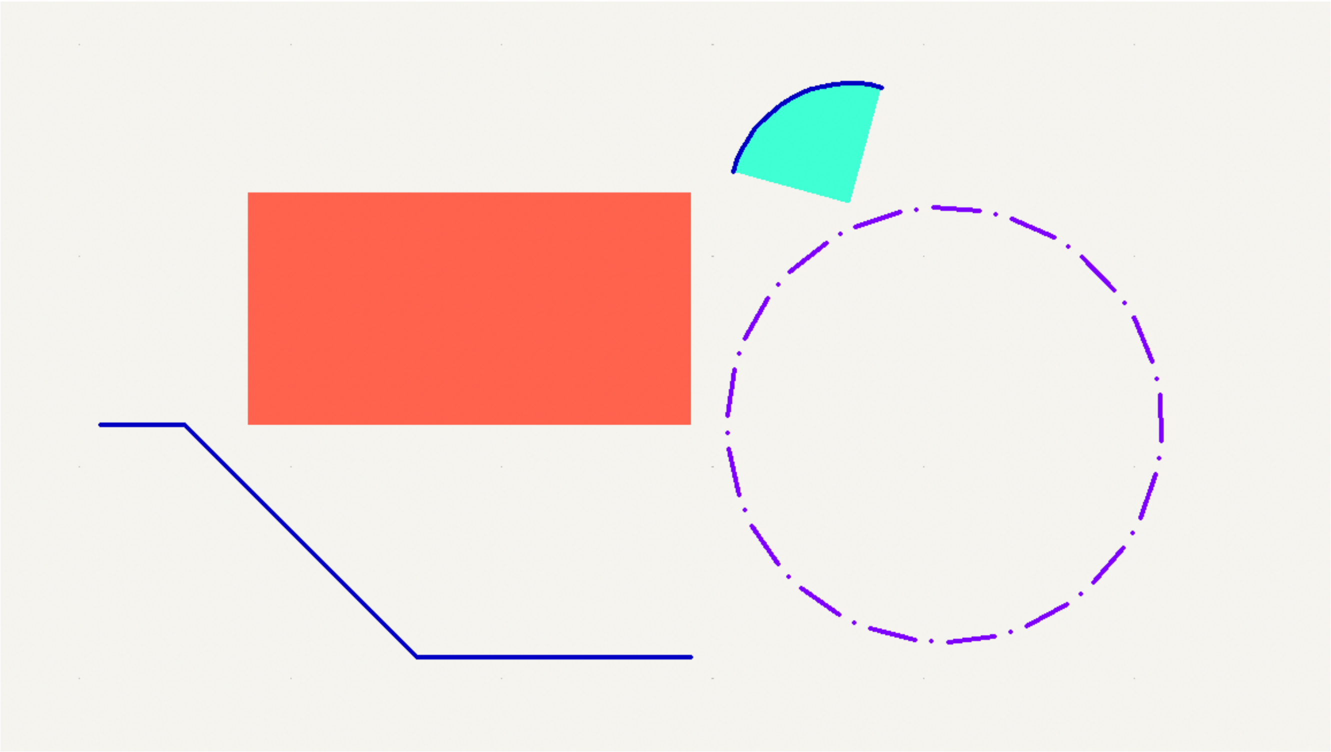
To add rows or columns, select a cell next to where the new row or column should go, right click, then choose **Add Row Above**, **Add Row Below**, **Add Column Before**, or **Add Column After**, as desired.

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.

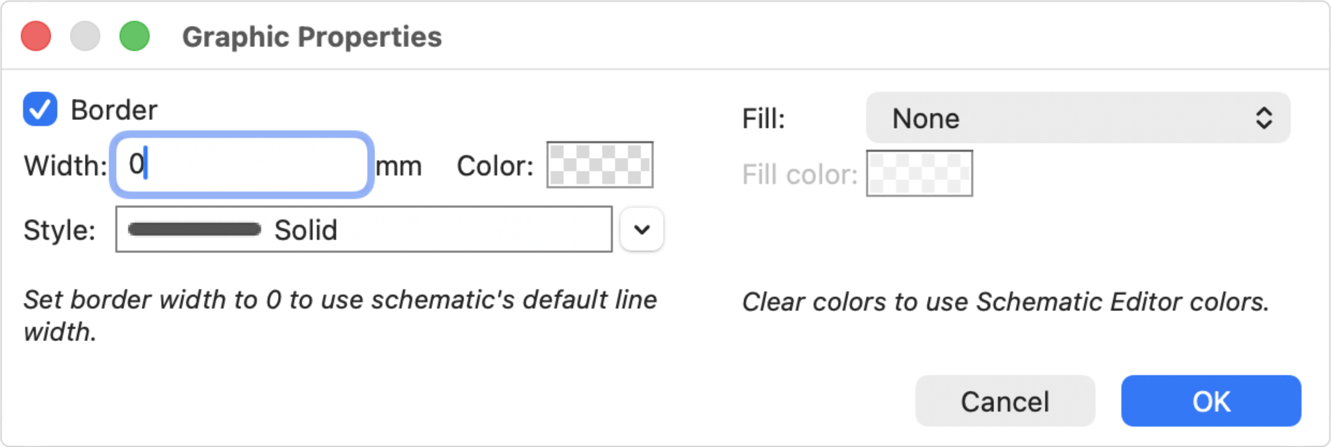
You can merge multiple cells into a single cell by selecting all the cells you want to merge, right clicking, and choosing **Merge Cells**. To unmerge them, select the merged cell, right click, and choose **Unmerge Cells**.

Graphic Shapes

Graphic rectangles (□), circles (○), arcs (◜), and lines (╱) can all be added using their respective buttons in the right toolbar.



Line width, color, and style (solid, dashed, or dotted) can be configured in the properties dialog for each shape (E). Rectangles, circles, and arcs can also have a fill color set and have their outlines removed.



Graphic Properties

☑ Border

Width: 0 mm Color:

Style: Solid

Set border width to 0 to use schematic's default line width.

Fill: None

Fill color:

Clear colors to use Schematic Editor colors.

Setting a shape’s line width to 0 uses the schematic default line width, which is configurable in Schematic Setup. Spacing for line dashes is also configurable there. Removing a line or fill color uses the color theme’s graphics color, which is configurable in Preferences.

Like wires, graphic lines obey the line drawing mode setting (90 degree, 45 degree, or free angle), which you can set using the toggle buttons on the left toolbar (, , and , respectively). `Shift`+`Space` cycles through the modes.

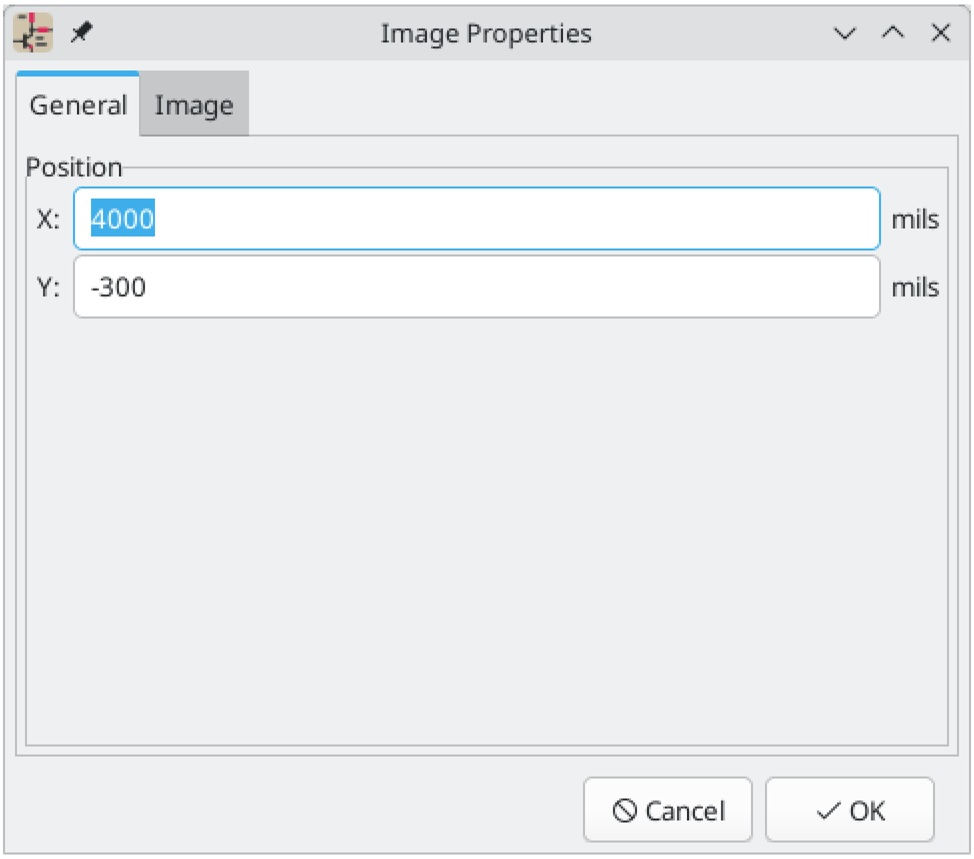
As with PCB tracks, the `/` hotkey switches line posture.

Bitmap Images

KiCad supports inserting images into the schematic. These are purely for reference during the design process and play no electrical role.

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 (`M`) 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.



Position

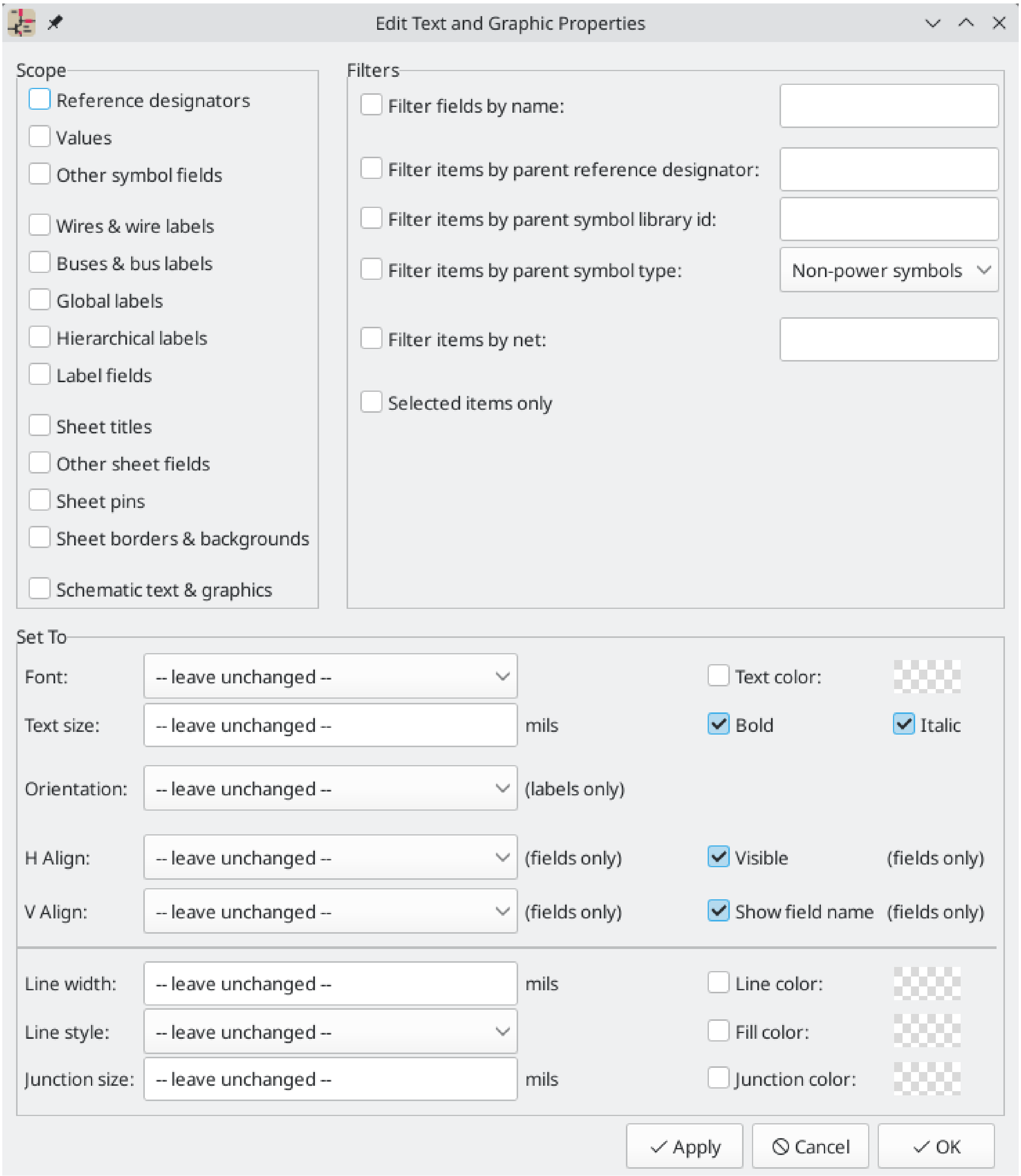
X: 4000 mils

Y: -300 mils

You can also reposition or scale the image in its properties dialog (E). 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.

Bulk editing text and graphics

Properties of text and graphics, including symbol fields, can be edited in bulk using the **Edit Text and Graphic Properties** dialog (**Edit** → **Edit Text and Graphic Properties...**). The tool can also modify visual properties of wires and buses.



Edit Text and Graphic Properties

Scope

□ Reference designators
□ Values
□ Other symbol fields

□ Wires & wire labels
□ Buses & bus labels
□ Global labels
□ Hierarchical labels
□ Label fields

□ Sheet titles
□ Other sheet fields
□ Sheet pins
□ Sheet borders & backgrounds

□ Schematic text & graphics

Filters

□ Filter fields by name:
□ Filter items by parent reference designator:
□ Filter items by parent symbol library id:
□ Filter items by parent symbol type: Non-power symbols
□ Filter items by net:
□ Selected items only

Set To

Font: -- leave unchanged --
□ Text color:
Text size: -- leave unchanged -- mils
☑ Bold
☑ Italic
Orientation: -- leave unchanged -- (labels only)
H Align: -- leave unchanged -- (fields only)
☑ Visible (fields only)
V Align: -- leave unchanged -- (fields only)
☑ Show field name (fields only)

Line width: -- leave unchanged -- mils
□ Line color:
Line style: -- leave unchanged --
□ Fill color:
Junction size: -- leave unchanged -- mils
□ Junction color:

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, symbol field filters do not apply to wires and are ignored for the purpose of changing wire 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 number of any characters, including none, and `?` matches any single character.

Filter fields by name filters to the specified symbol, label, or sheet field.

Filter items by parent reference designator filters to fields in the symbol with the specified reference designator. **Filter items by parent symbol library id** filters to fields in symbols with the specified library identifier. **Filter items by parent symbol type** filters to fields in symbols of the selected type (power or non-power).

Filter items by net filters to wires and labels on the specified net.

Only include selected items filters to the current selection.

Editable Properties

Properties for filtered objects can be set to new values in the bottom part of the dialog.

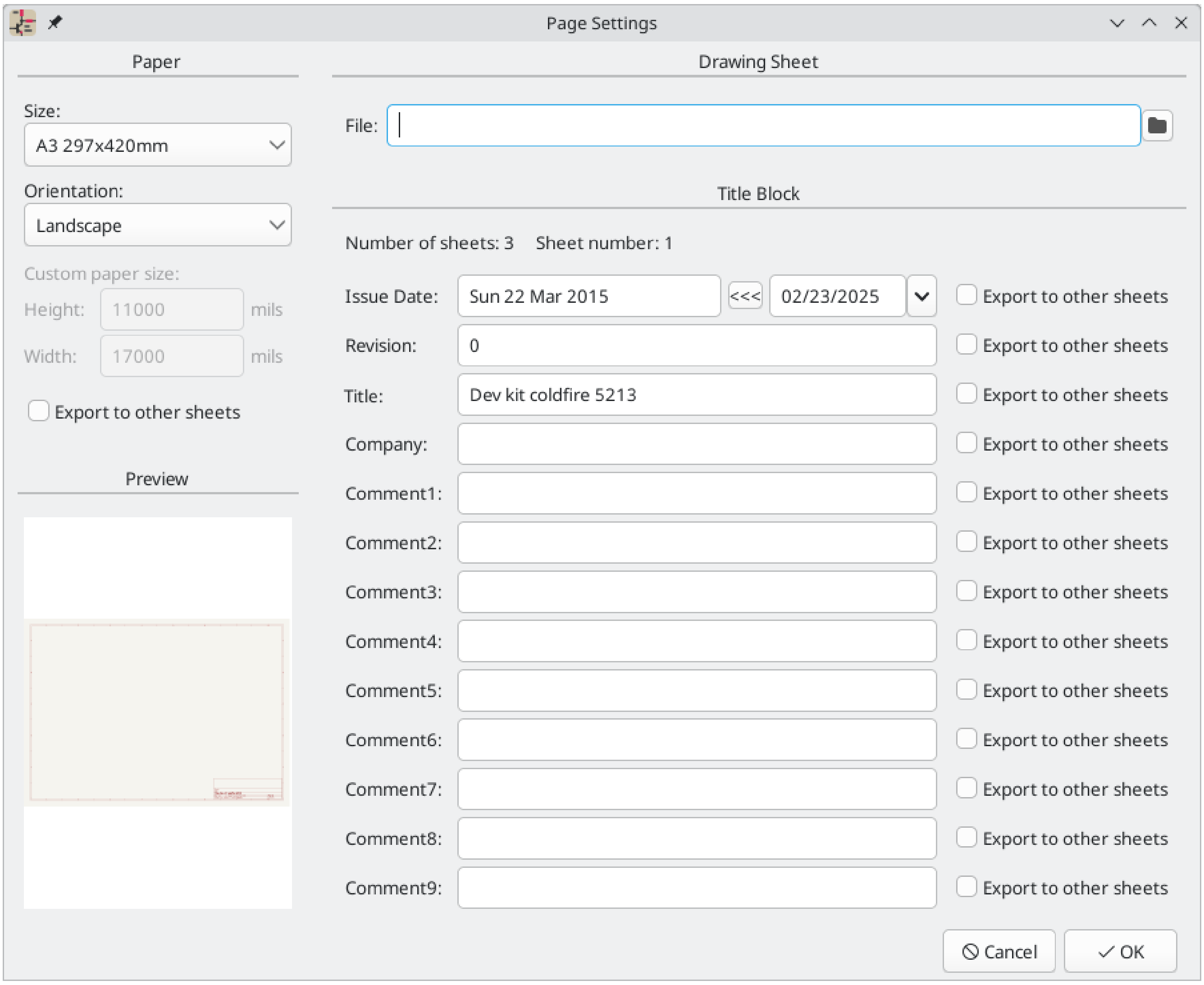
Drop-down lists and text boxes can be set to `-- leave unchanged --` to preserve existing values. Checkboxes can be checked or unchecked to enable or disable a change, but can also be toggled to a third "leave unchanged" state. Color properties must be checked to change the value; a checkerboard swatch indicates that the color will be inherited from the default value from the the schematic settings or net class properties.

Text properties that can be modified are **font**, **text size**, **text orientation** (right/up/leftdown), **horizontal** and **vertical alignment**, **text color**, emphasis (**bold** and **italic**), and **visibility** of fields and field names.

Graphic and wire properties that can be modified are **line width**, **line style** (solid, dashed, and dotted lines), **line color**, **fill color** for shapes, and **junction size** and **junction color** for wire junctions.

Blok Judul pada Lembar Kerja

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



Page Settings

Paper

Size: A3 297x420mm

Orientation: Landscape

Custom paper size:

Height: 11000 mils

Width: 17000 mils

□ Export to other sheets

Preview

Drawing Sheet

File:

Title Block

Number of sheets: 3 Sheet number: 1

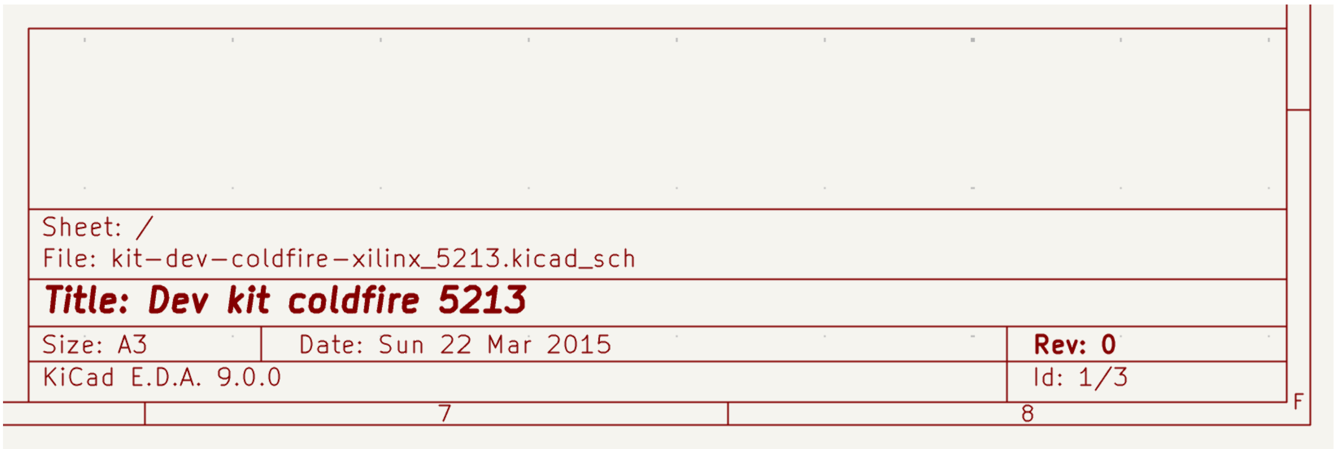
Issue Date:	Sun 22 Mar 2015 <<< 02/23/2025	□ Export to other sheets
Revision:	0	□ Export to other sheets
Title:	Dev kit coldfire 5213	□ Export to other sheets
Company:		□ Export to other sheets
Comment1:		□ Export to other sheets
Comment2:		□ Export to other sheets
Comment3:		□ Export to other sheets
Comment4:		□ Export to other sheets
Comment5:		□ Export to other sheets
Comment6:		□ Export to other sheets
Comment7:		□ Export to other sheets
Comment8:		□ Export to other sheets
Comment9:		□ Export to other sheets

Cancel OK

Each field in the title block can be edited, as well as the paper size and orientation. If the **Export to other sheets** option is checked for a field, that field will be updated in the title block of all sheets, rather than only the current sheet.

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 schematic 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 schematic instead of referencing an external file. This means the schematic 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.

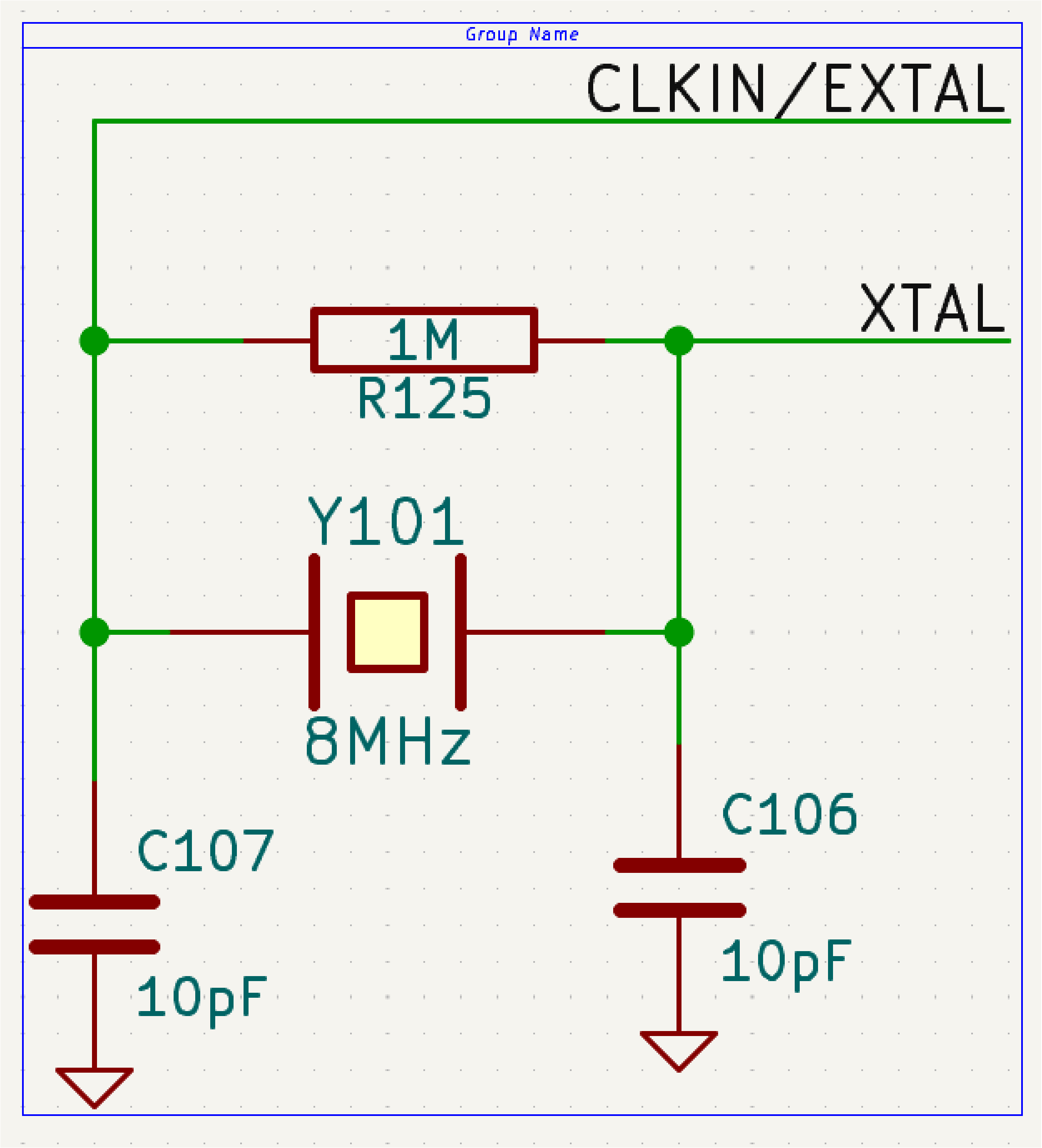


Sheet: /
File: kit-dev-coldfire-xilinx_5213.kicad_sch
Title: Dev kit coldfire 5213
Size: A3 Date: Sun 22 Mar 2015 **Rev: 0**
KiCad E.D.A. 9.0.0 Id: 1/3

The sheet number (Sheet X/Y) is automatically updated, but sheet page numbers can also be manually set using **Edit** → **Edit Sheet Page Number…**.

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.



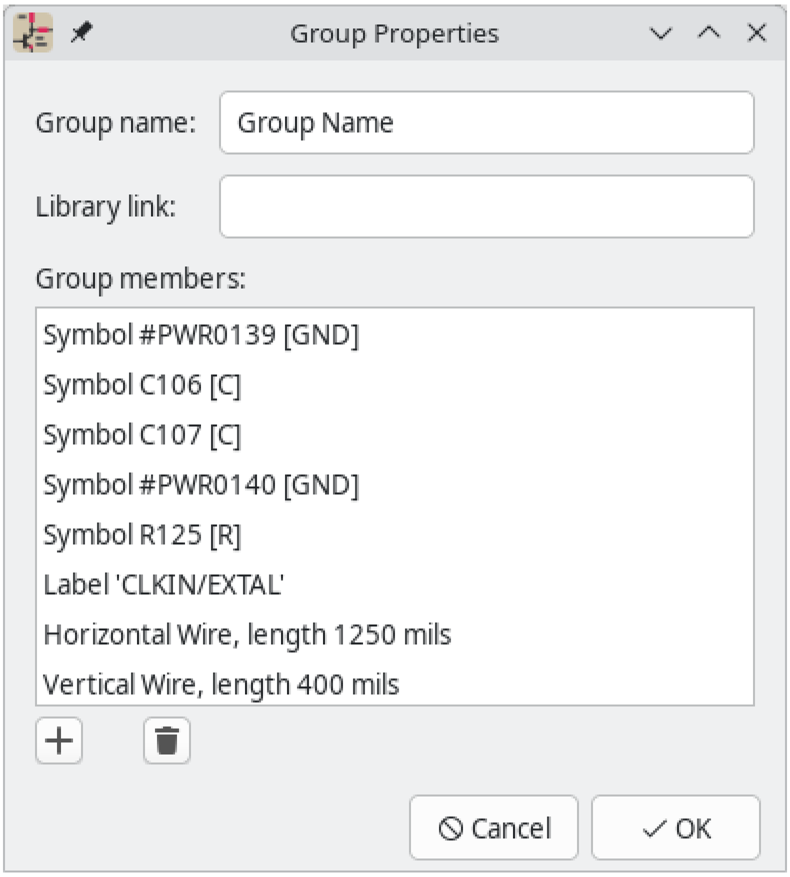
Most types of objects in the Schematic Editor can be grouped: symbols, wires, labels, graphic items, and even other groups. Groups can contain multiple different types of objects at once.

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 Esc.

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 E 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.



Group name: Group Name

Library link:

Group members:

Symbol #PWR0139 [GND]
Symbol C106 [C]
Symbol C107 [C]
Symbol #PWR0140 [GND]
Symbol R125 [R]
Label 'CLKIN/EXTAL'
Horizontal Wire, length 1250 mils
Vertical Wire, length 400 mils

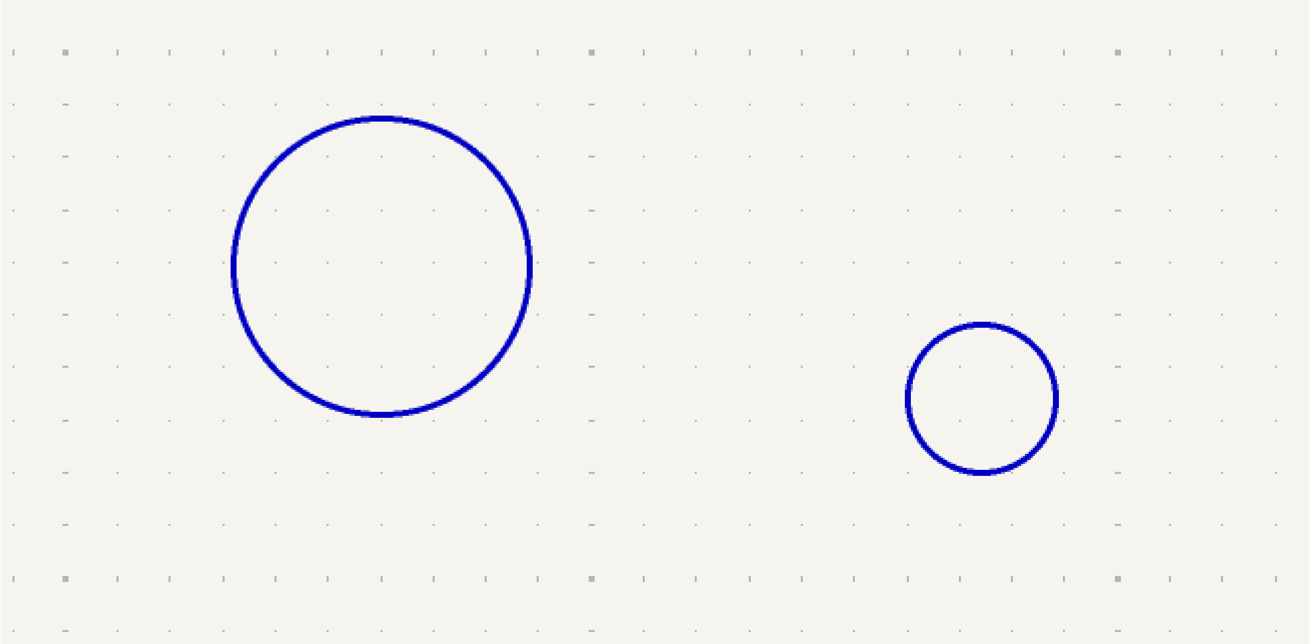
The **Library link** field is used for design blocks. It specifies the group's linked design block name and library in the format `<library>:<block>`. This field must be filled out correctly to link design block schematic groups to the corresponding block in the board and in the design block library.

You can optionally transfer groups from the schematic to the PCB using the Update PCB from Schematic tool. If the **Group footprints based on symbol group** option is enabled, footprints in the PCB will be grouped if their linked symbols are grouped in the schematic.

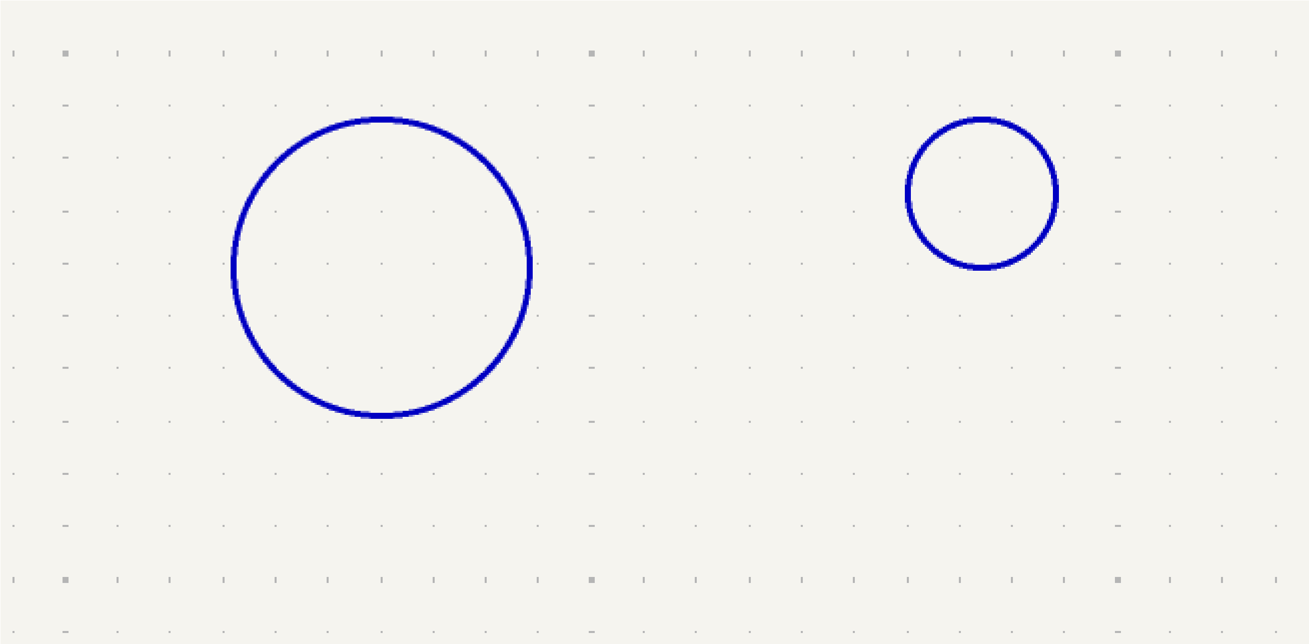
Aligning objects

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, two circles are vertically aligned by their top edges, with the large circle as the reference object. The first image shows them before alignment and the second image shows them after alignment. In this case, the large circle is the topmost object before alignment, so it is chosen as the reference object if the cursor is not over the other circle. After alignment, the top edges of the circles are at the same position, but the horizontal positions of the circles are unchanged.

NOTE

Symbols, labels, and other objects with pin connections are always moved so that their connection points remain on the grid, even if this results in imperfect alignment with the other objects.

Schematic editing convenience functions

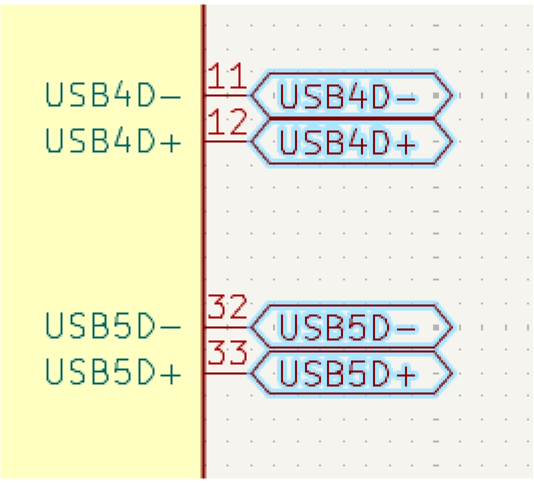
There are several convenience features in the Schematic Editor that make some common editing and connection operations faster.

Pin helpers

You can quickly add wires, labels, or no-connection markers to a selection of pins using the **Pin Helpers** tools in the right-click context menu. This can help you quickly break out unconnected pins from a symbol or hierarchical sheet. By selecting **Pin Helpers** → **Wire**, the wire tool will begin drawing a wire from all selected pins at once. If you select **No Connect**, no-connection markers will be added to the end of each selected pin. And if you choose **Net Label**, **Hierarchical Label**, or **Global Label**, a label of the respective type will be placed at the end of each selected pin. Each label's name will be set to the corresponding pin name. The new labels will remain selected, so you can easily move them away from the symbol using M or G, depending on whether you wish to maintain a wired connection between the pins and the labels.

NOTE

Pin helpers require you to select individual pins, not their parent symbol or sheet. Symbol pins cannot be individually selected if the **clicking on a pin selects the symbol** option is enabled in the Editing Options pane of the Schematic Editor preferences. Therefore, this option must be disabled to use the Pin Helper tools.



Converting between object types

Existing labels and text objects can be changed to another type of label or text by right clicking the object(s) and selecting the target object type from the **Change To** submenu. The allowed types for source and target objects are local labels, global labels, hierarchical labels, directive labels, text objects, and text boxes. The value of the original object is preserved in the resulting object: when a text object is converted to a label, the label's value (net name) will be the original text, and vice versa.

Swapping objects

You can swap the position of two selected objects using the Swap command (Alt + S; also available in the right-click context menu). This works on many schematic items, including symbols, symbol fields, labels, graphical items, and text. The first object is assigned the location and rotation of the second object, and vice versa. If there are more than two objects selected, the locations are cycled: the last object gets the position of the first object, the first object gets the location of the second, and so on.

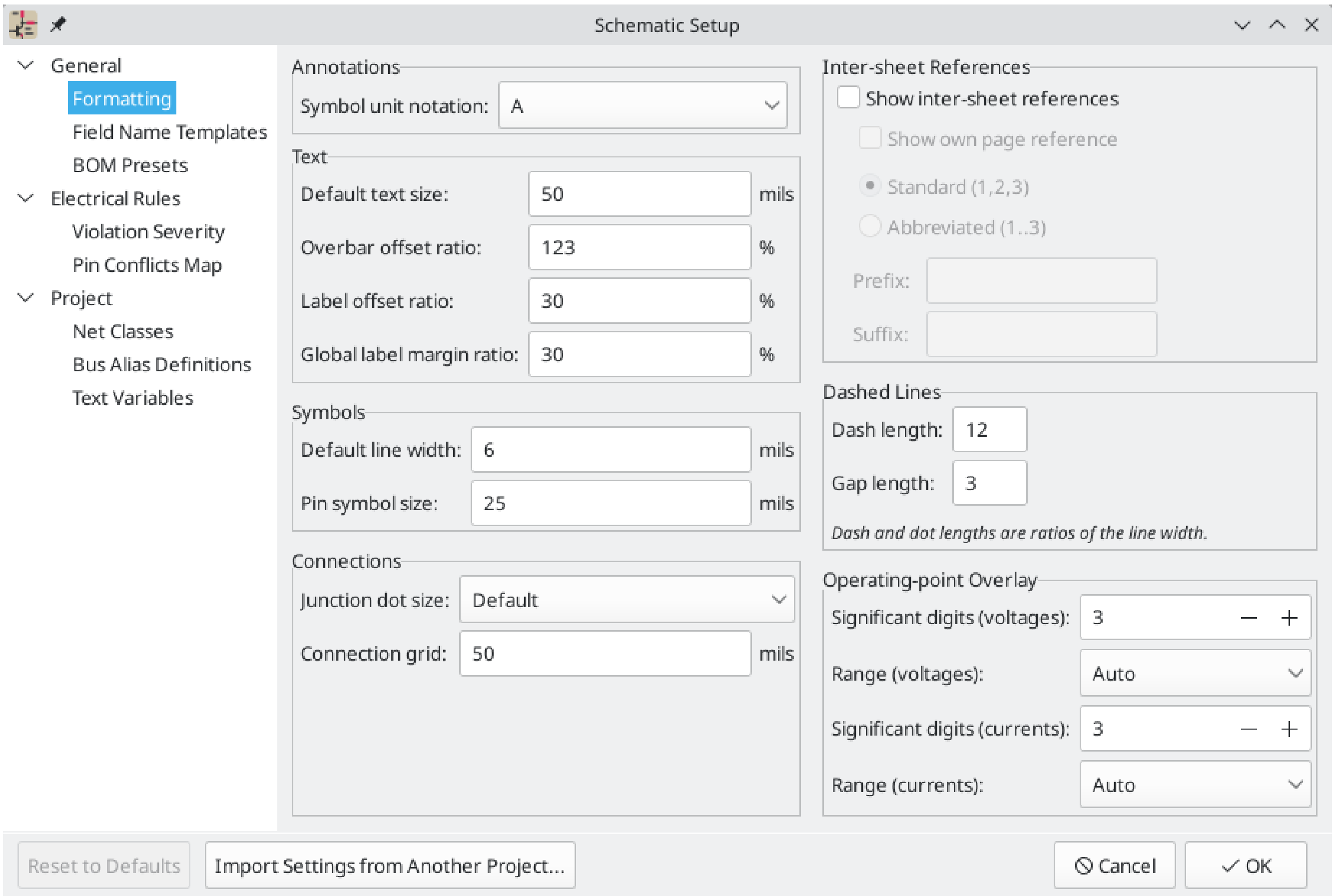
> **TIP**
>
> One possible use of the swap command is to exchange two units within a a symbol, for example the two amplifiers in a dual op-amp. You could also use swap with a selection of labels to quickly modify net assignments to symbol pins. In combination with cross-selection from the PCB, this can be a convenient way to make schematic changes for easier routing. This is sometimes known as pin or gate swapping.

Schematic Setup

The Schematic Setup window is used to set schematic options that are specific to the currently active schematic. For example, the Schematic Setup window contains formatting options, electrical rule configuration, net class setup, and schematic text variable setup.

To open Schematic Setup, use **File** → **Schematic Setup...** or press the button in the top toolbar.

Schematic formatting



Annotations

Symbol unit notation: A

Text

Default text size: 50 mils

Overbar offset ratio: 123 %

Label offset ratio: 30 %

Global label margin ratio: 30 %

Symbols

Default line width: 6 mils

Pin symbol size: 25 mils

Connections

Junction dot size: Default

Connection grid: 50 mils

Inter-sheet References

□ Show inter-sheet references

□ Show own page reference

● Standard (1,2,3)

○ Abbreviated (1..3)

Prefix:

Suffix:

Dashed Lines

Dash length: 12

Gap length: 3

Dash and dot lengths are ratios of the line width.

Operating-point Overlay

Significant digits (voltages): 3

Range (voltages): Auto

Significant digits (currents): 3

Range (currents): Auto

The formatting panel contains settings for the appearance of symbols, text, labels, graphics, and wires.

Symbol unit notation sets how each unit of a multi-unit symbol is referred to in its reference designator. By default, a different letter for each unit is appended to the reference designator with no separator, for example `U1B` for the second unit of symbol `U1`, but this can be changed. Numbers can be used instead of letters, and various separators can be used between the symbol designator and the unit identifier (`.`, `-`, `_`, or none).

Default text size sets the default text height used by the text, text box, and label tools. **Overbar offset ratio** controls the vertical spacing between text and an overbar (`~{}`) over that text, as a ratio of the text height. **Label offset ratio** controls the vertical spacing between a local label's text and the attached wire, relative to the label's text size. This also affects the spacing between symbol pins and their pin number. **Global label margin ratio** defines the size of the box around a global label, relative to the global label's text size. Increasing the margin may be useful to avoid overlapping text with overbars (`~{}`) or letters with descenders, but this may cause closely packed global labels to overlap with each other.

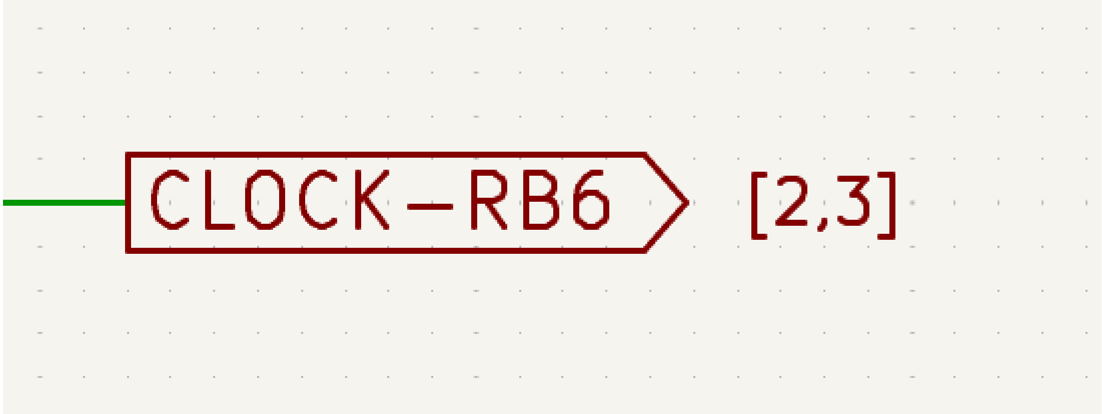
Default line width sets the default line width for symbol graphics, if the symbol does not override the default line width. **Pin symbol size** scales symbol pin graphic style annotations, such as the bubble on an inverted pin.

Hop-over size sets the size of wire hop-overs, which are optionally drawn when one wire crosses another wire without connecting. By default the size is set to **None**, which means hop-overs are not drawn.

Junction dot size sets the schematic's default wire junction dot size. The default size can be overridden by editing an individual junction dot's properties. **Connection width** specifies the grid size used for the **Symbol pin or wire end off connection grid** ERC check. Schematics typically use a 50 mil grid for electrical connections, so this should usually remain set at 50 mils.

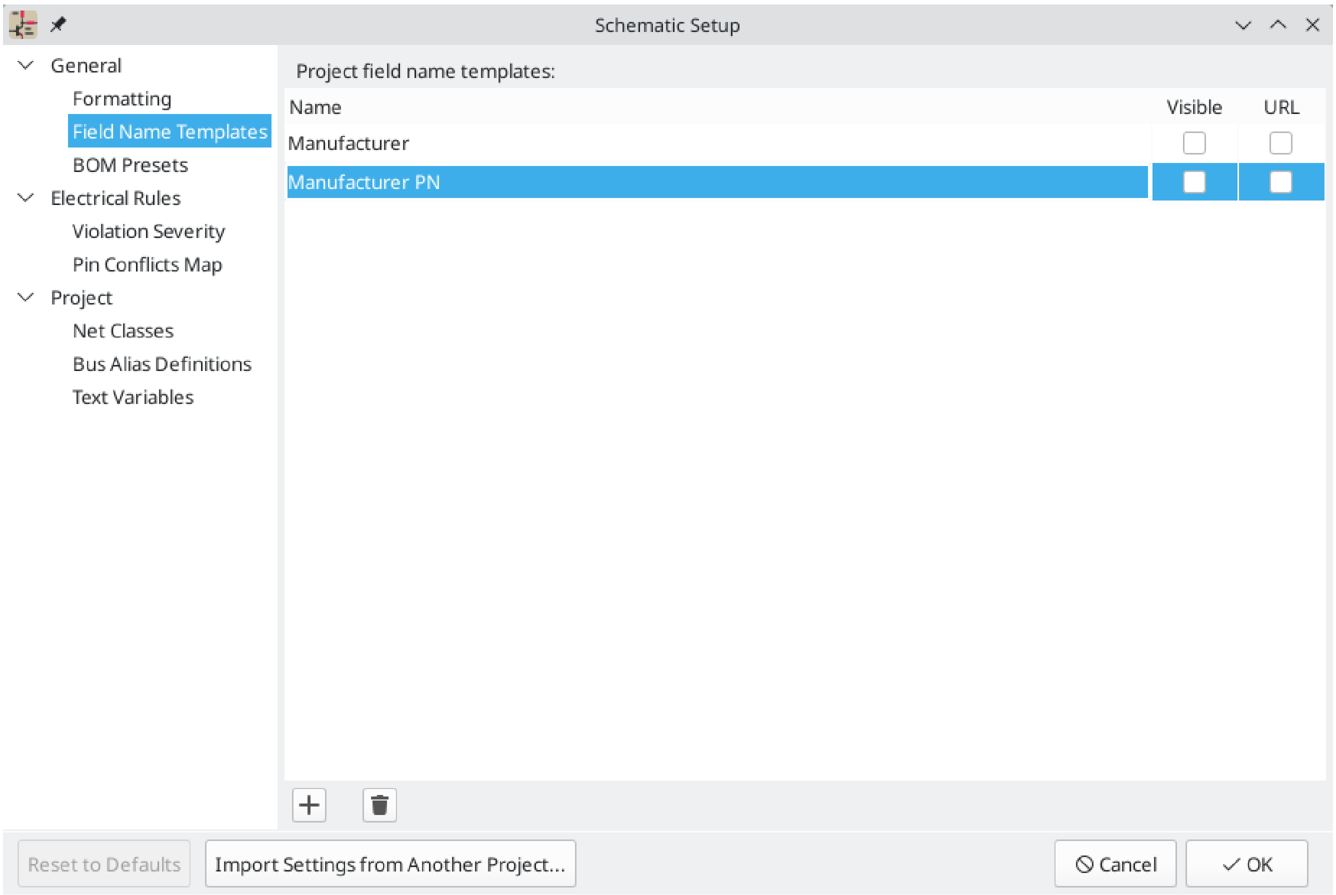
The Operating Point Overlay settings configure how operating point simulation annotations are displayed on the schematic canvas. The **significant digits** settings control the number of significant digits printed on voltage and current overlays. The **range** settings control the units used to display voltage and current measurements.

Show inter-sheet references enables or disables the display of inter-sheet references, which are a list of page numbers next to a global labels that link to other places in the schematic where the same global label appears. **Show own page reference** controls whether the current page is included in the list of page numbers. **Standard** and **abbreviated** determine whether to display the complete list of page numbers or only the first and last page numbers. The **prefix** and **suffix** fields add optional characters before and after the list of page numbers. In the image of an inter-sheet reference below, a prefix and suffix of `[` and `]`, respectively, have been added.



Dashed line appearance is controlled in the Formatting section. **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.

Field name templates



Project field name templates:

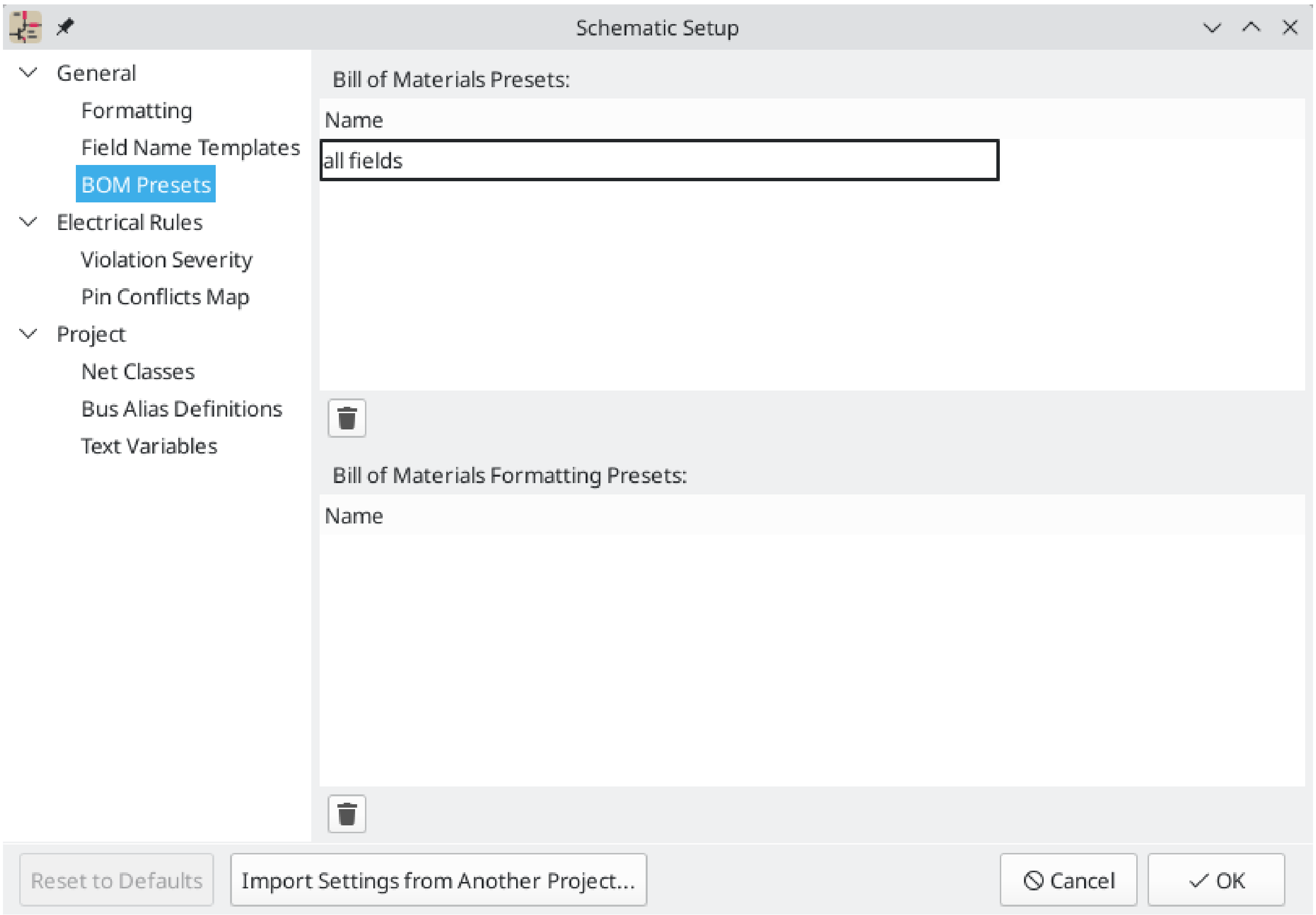
Name	Visible	URL
Manufacturer	□	□
Manufacturer PN	□	□

Field name templates are empty symbol fields that are automatically added to all symbols in the schematic. These can be useful when every symbol in the schematic needs additional fields beyond the fields that are defined in the library symbols, for example a field for the manufacturer's part number.

Template fields can be set as visible or invisible, and can also be set as URL fields.

Field name templates that are defined in schematic setup apply only to the current project. Field name templates can also be defined in Preferences, which apply to all projects edited on your computer.

BOM presets



Bill of Materials Presets:

Name
all fields

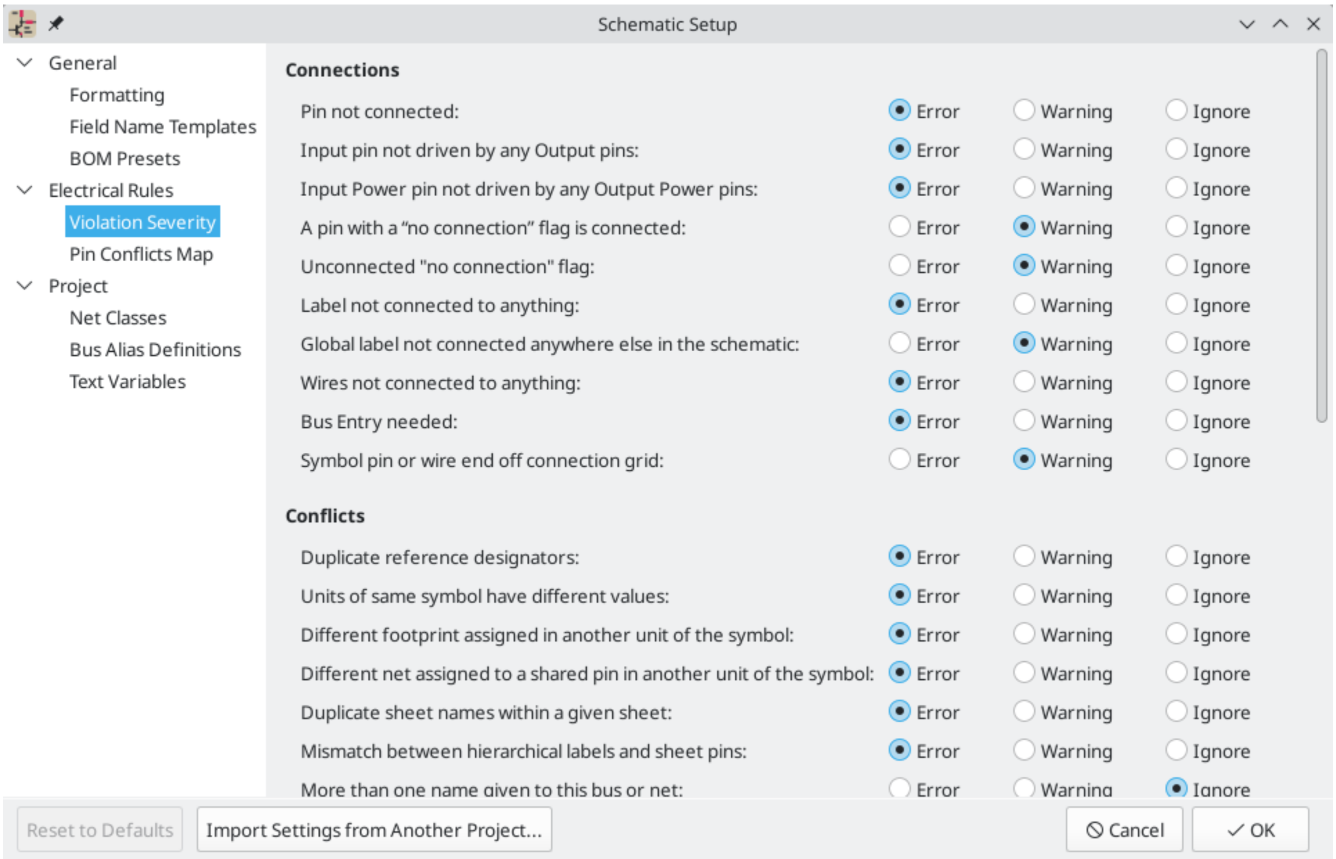
Bill of Materials Formatting Presets:

Name

BOM presets are saved configurations for the Symbol Fields Table and BOM export tool. There are two types of presets. **BOM presets** configure which fields are displayed in the symbol fields table, which order they are displayed in, and how they are used to group symbols. These fields are also directly used in the BOM output. **BOM formatting presets** configure the output BOM file format, including which separator characters are used to separate fields. Both types of presets are created in the Symbol Fields Table, but can are listed and can be deleted here.

ERC violation severity and pin conflicts map

The **Violation Severity** panel lets you configure what types of ERC messages should be reported as Errors, Warnings, or ignored.



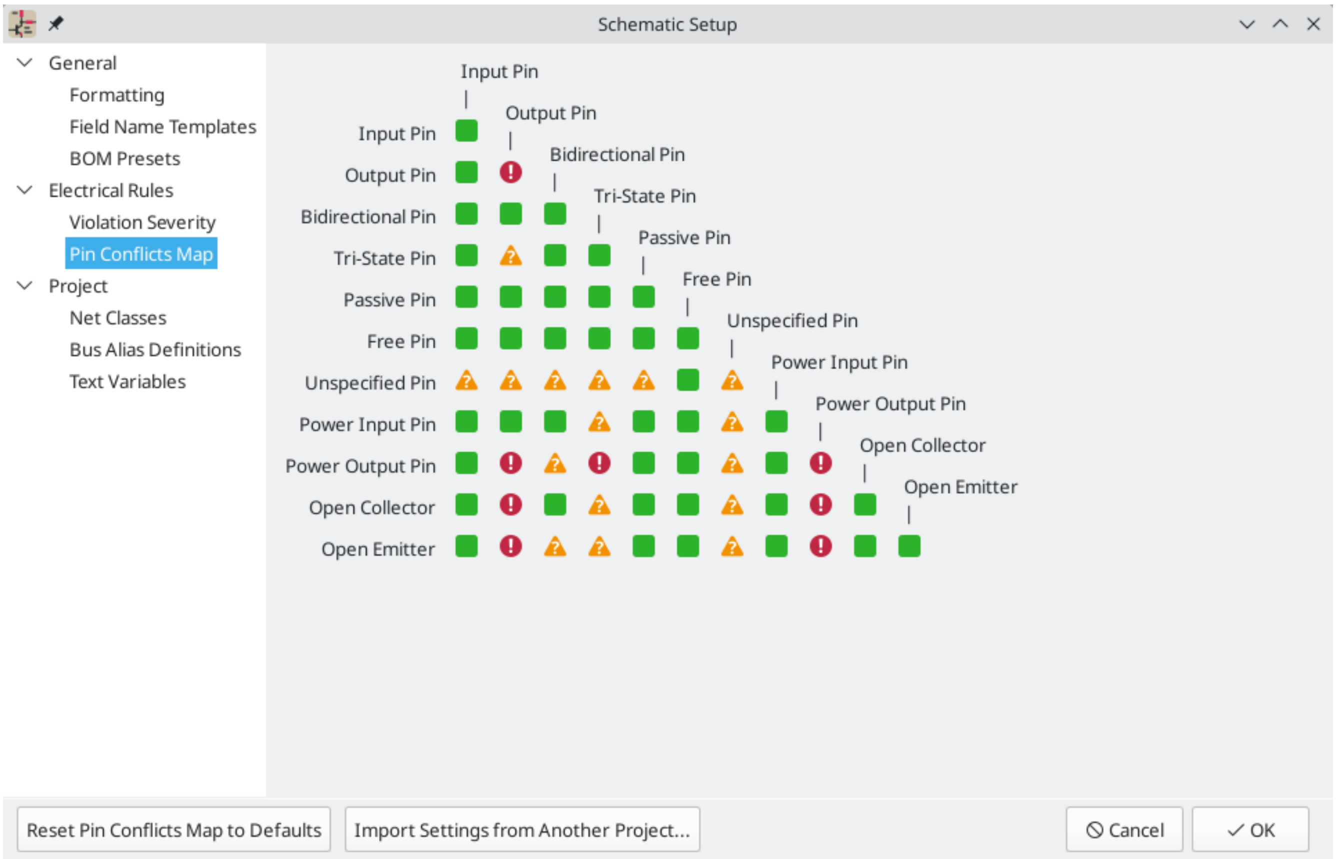
Connections

Pin not connected:	● Error	○ Warning	○ Ignore
Input pin not driven by any Output pins:	● Error	○ Warning	○ Ignore
Input Power pin not driven by any Output Power pins:	● Error	○ Warning	○ Ignore
A pin with a "no connection" flag is connected:	○ Error	● Warning	○ Ignore
Unconnected "no connection" flag:	○ Error	● Warning	○ Ignore
Label not connected to anything:	● Error	○ Warning	○ Ignore
Global label not connected anywhere else in the schematic:	○ Error	● Warning	○ Ignore
Wires not connected to anything:	● Error	○ Warning	○ Ignore
Bus Entry needed:	● Error	○ Warning	○ Ignore
Symbol pin or wire end off connection grid:	○ Error	● Warning	○ Ignore

Conflicts

Duplicate reference designators:	● Error	○ Warning	○ Ignore
Units of same symbol have different values:	● Error	○ Warning	○ Ignore
Different footprint assigned in another unit of the symbol:	● Error	○ Warning	○ Ignore
Different net assigned to a shared pin in another unit of the symbol:	● Error	○ Warning	○ Ignore
Duplicate sheet names within a given sheet:	● Error	○ Warning	○ Ignore
Mismatch between hierarchical labels and sheet pins:	● Error	○ Warning	○ Ignore
More than one name given to this bus or net:	○ Error	○ Warning	● Ignore

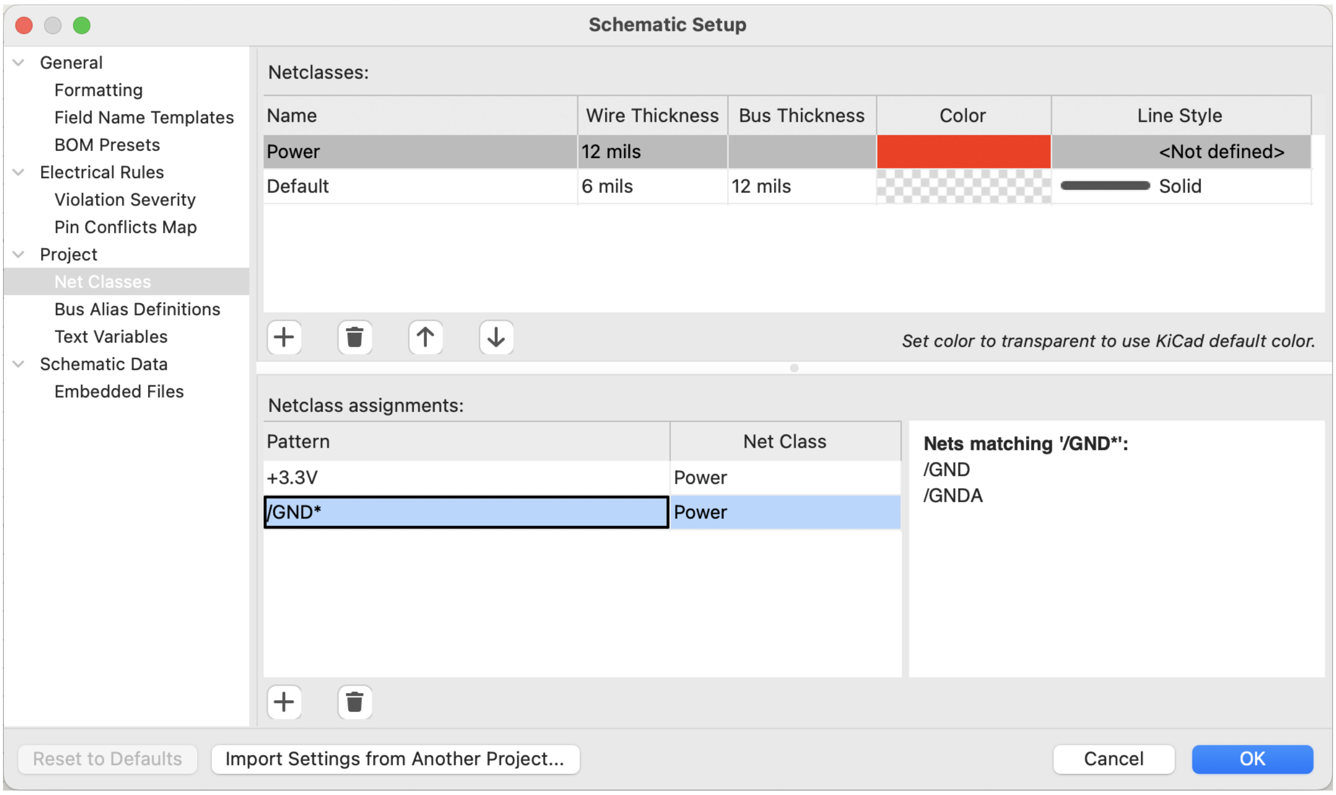
The **Pin Conflicts Map** allows you to configure connectivity rules to define electrical conditions for errors and warnings based on what types of pins are connected to each other. For example, by default an error is produced when an output pin is connected to another output pin.



	Input Pin	Output Pin	Bidirectional Pin	Tri-State Pin	Passive Pin	Free Pin	Unspecified Pin	Power Input Pin	Power Output Pin	Open Collector	Open Emitter
Input Pin	🟩										
Output Pin	🟩	❗									
Bidirectional Pin	🟩	🟩	🟩								
Tri-State Pin	🟩	⚠	🟩	🟩							
Passive Pin	🟩	🟩	🟩	🟩	🟩						
Free Pin	🟩	🟩	🟩	🟩	🟩	🟩					
Unspecified Pin	⚠	⚠	⚠	⚠	⚠	🟩	⚠				
Power Input Pin	🟩	🟩	🟩	⚠	🟩	🟩	⚠	🟩			
Power Output Pin	🟩	❗	⚠	❗	🟩	🟩	⚠	🟩	❗		
Open Collector	🟩	❗	🟩	⚠	🟩	🟩	⚠	🟩	❗	🟩	
Open Emitter	🟩	❗	⚠	⚠	🟩	🟩	⚠	🟩	❗	🟩	🟩

These panels are explained in more detail in the ERC section.

Net classes



Netclasses:

Name	Wire Thickness	Bus Thickness	Color	Line Style
Power	12 mils		🟥	<Not defined>
Default	6 mils	12 mils	▨	Solid

Set color to transparent to use KiCad default color.

Netclass assignments:

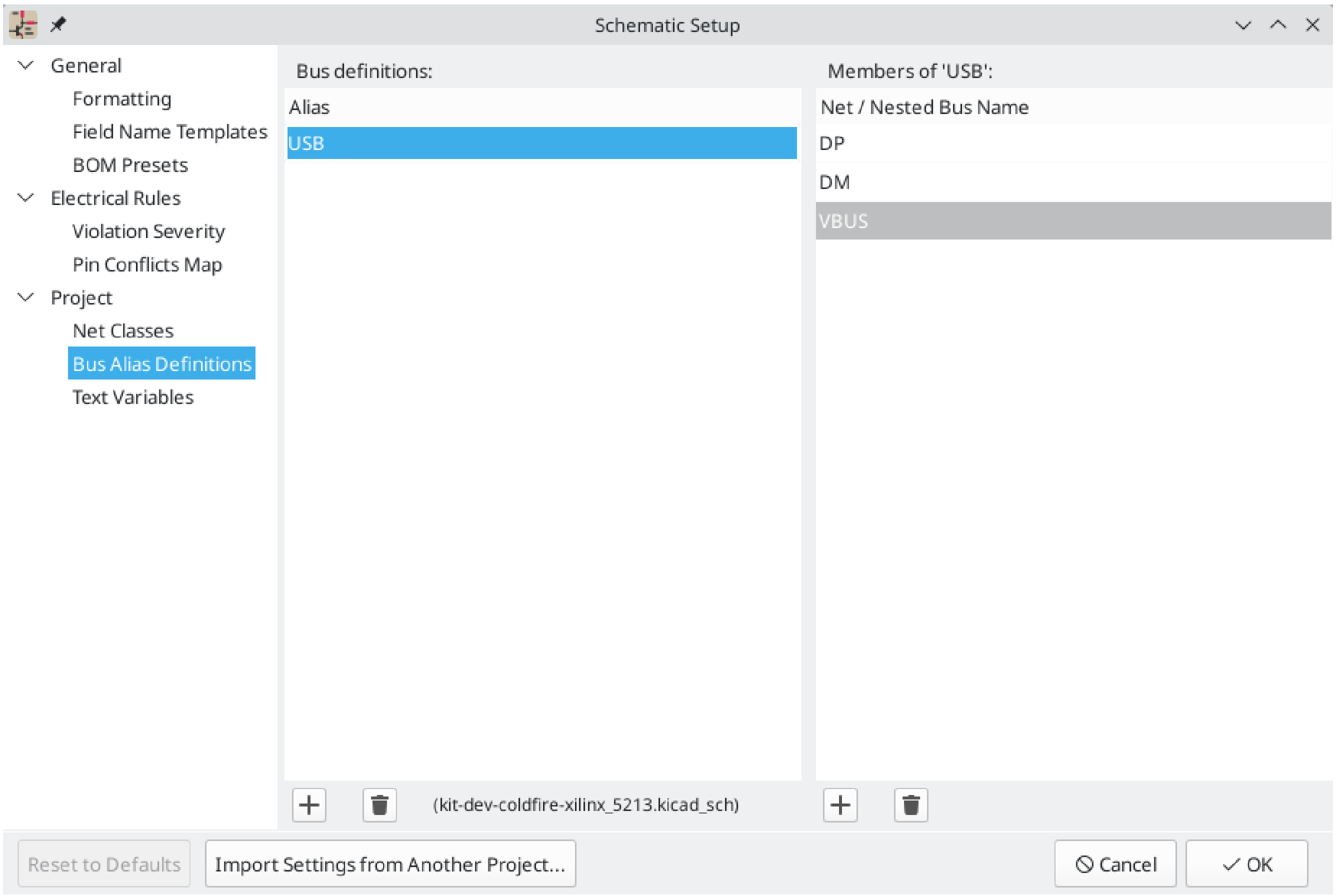
Pattern	Net Class
+3.3V	Power
/GND*	Power

Nets matching '/GND*':
/GND
/GNDA

The **Net Classes** panel allows you to manage net classes for the project and assign nets to net classes with patterns. Managing net classes in this panel is equivalent to managing them in the Board Setup dialog. Nets can also be assigned to net classes in the schematic using graphical assignments with net class directives or net labels.

Pattern-based net class assigment is explained in more detail in the net classes section.

Bus alias definitions



Bus definitions:

Alias
USB

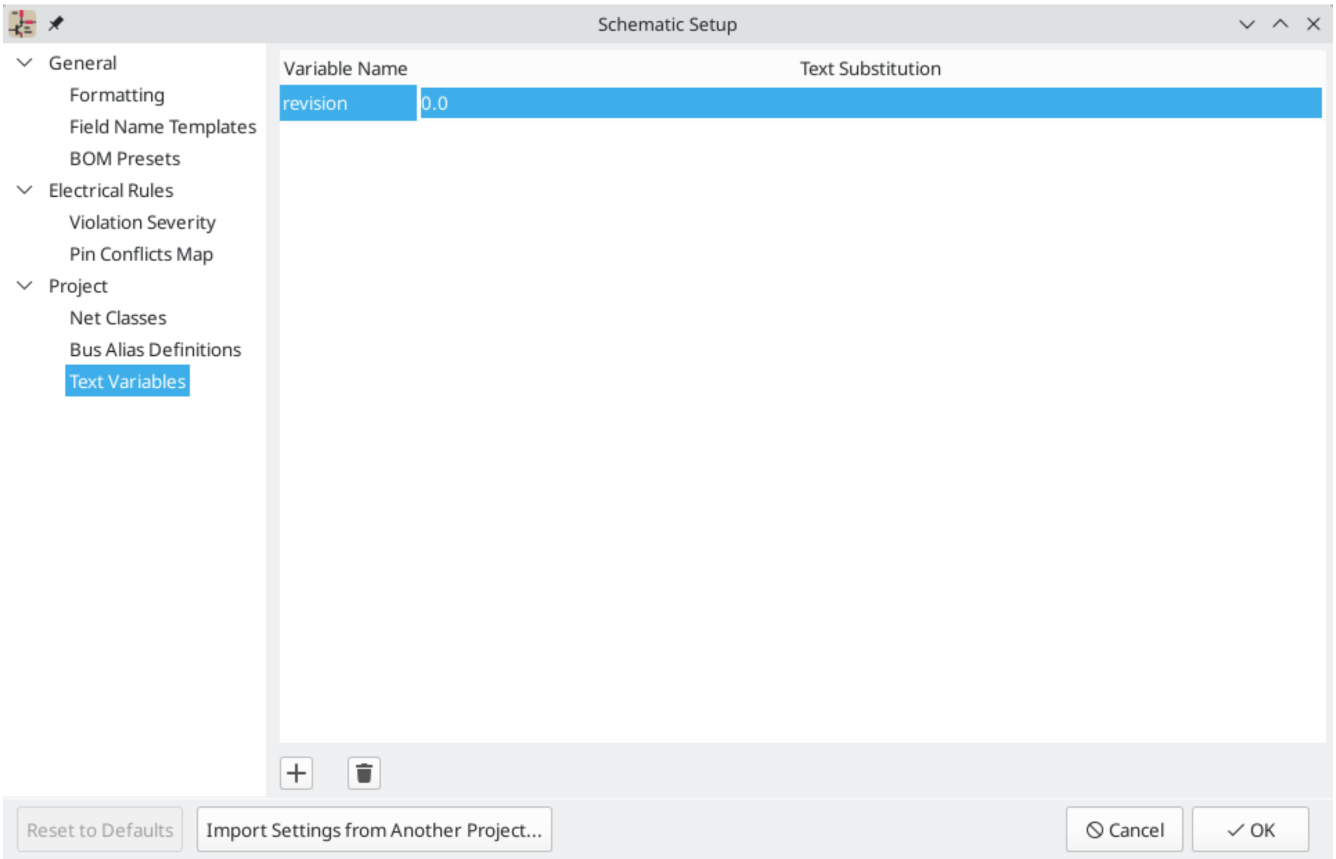
Members of 'USB':

Net / Nested Bus Name
DP
DM
VBUS

(kit-dev-coldfire-xilinx_5213.kicad_sch)

The **Bus Alias Definitions** panel allows you to create bus aliases, which are names for groups of signals in a bus. For more information about bus aliases, see the bus alias documentation.

Text variables



Variable Name	Text Substitution
revision	0.0

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}`.

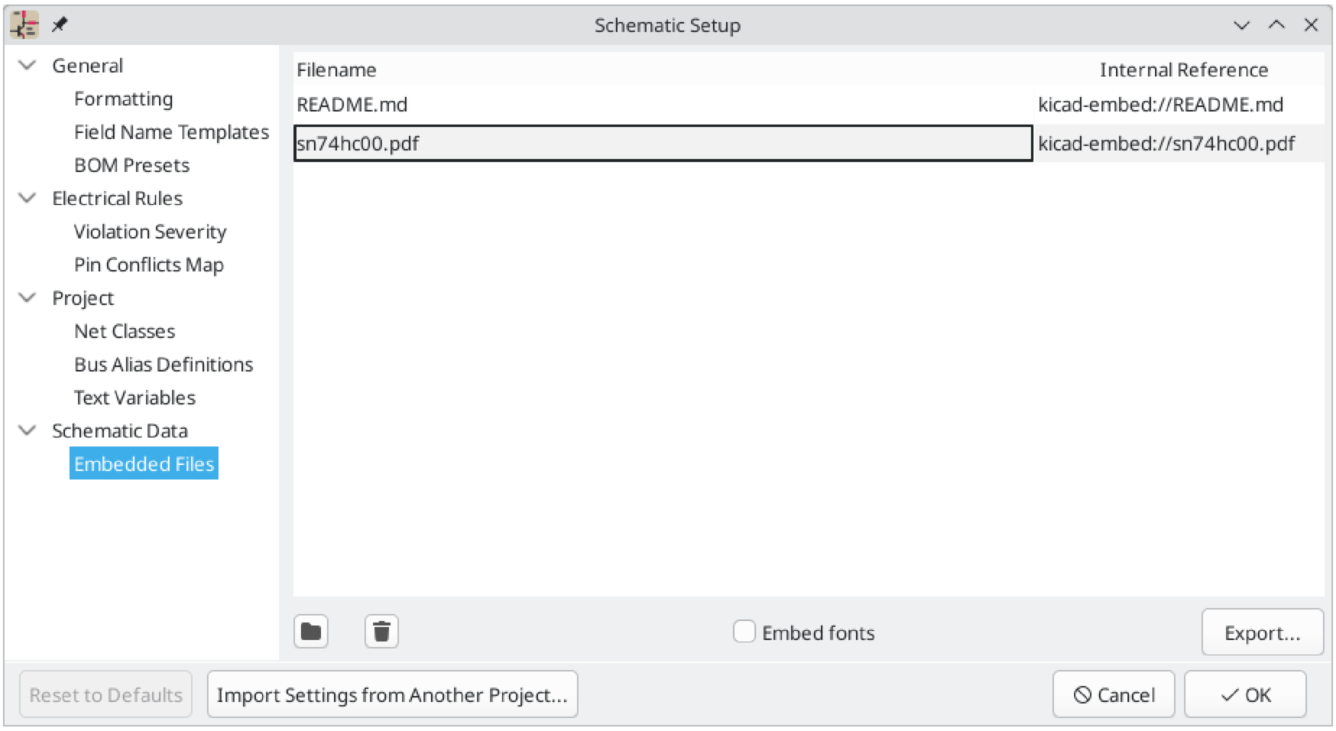
For example, you could create a variable named `VERSION` and set the text substitution to `1.0`. Now, in any text object in the schematic, you can enter `${VERSION}` and KiCad will display this as `1.0`. If you change the value to `2.0`, every text object that includes `${VERSION}` will be updated automatically. 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 Board 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.

Embedding files

External files can be embedded within a schematic. Embedding a file stores a copy of the file inside the schematic 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 schematic necessarily increase the schematic's file size, although files are compressed before being embedded to minimize the space required.



Filename	Internal Reference
README.md	kicad-embed://README.md
sn74hc00.pdf	kicad-embed://sn74hc00.pdf

Embedded files are managed in the Embedded Files section of Schematic Setup. All files embedded in a schematic are shown here. To embed a file inside a schematic, click the 📁 button and select the file. The file is then embedded inside the schematic 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 Schematic 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** Datasheets, SPICE models, and drawing sheets can be embedded directly using the file browser when you add them to a symbol (datasheets and SPICE models) or to a schematic (drawing sheets) by enabling the **Embed Files** option in the file browser. This is a single-step shortcut for adding the files in Schematic Setup and then referring to them by their embedded reference; the result is the same.

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

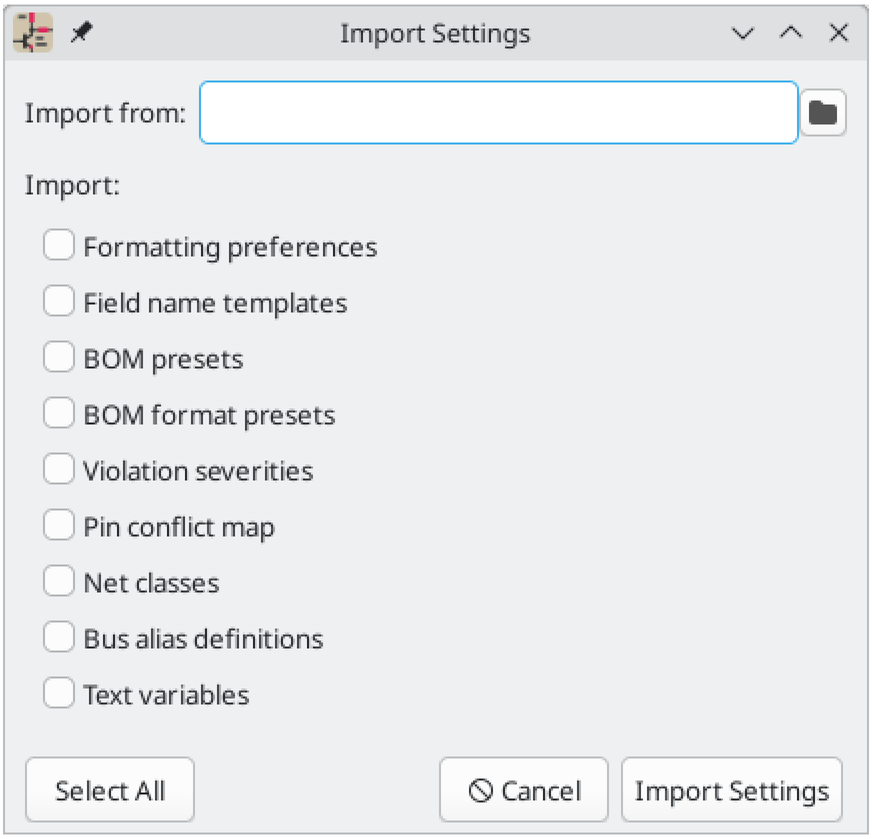
You can also embed files in a symbol. Such files will be available within the symbol, but not within the larger schematic or in other symbols. Files embedded in a symbol are deduplicated when the symbol is added to a schematic: if a file is embedded in a symbol, and multiple instances of that symbol are added to the schematic, only one copy of the file will be embedded, and all of the symbol instances will refer to the same embedded file.

As an example, to embed a datasheet in a project and use it within several symbols, you could embed the datasheet using the schematic setup dialog, copy the internal reference, and paste the internal reference into the datasheet field of each symbol that uses that datasheet. Each symbol would then have a portable reference to the embedded datasheet. Alternatively, you could embed the datasheet within the library symbol. In this case, each symbol will already have the datasheet embedded when the symbol is added to a schematic. A more convenient way to achieve the same thing, however, is to open the symbol's properties dialog, browse for a datasheet file, and enable the **Embed File** option in the file browser. Again, this could be done for a symbol in the schematic or for a symbol in the source symbol library.

Files can also be embedded in boards.

Importing settings

You can import some or all of the schematic setup from an existing schematic. This allows you to choose a schematic to use as a template and select which settings to import.



Import from:

Import:

□ Formatting preferences
□ Field name templates
□ BOM presets
□ BOM format presets
□ Violation severities
□ Pin conflict map
□ Net classes
□ Bus alias definitions
□ Text variables

To import settings, click the **Import Settings from Another Project...** button at the bottom of the Schematic Setup dialog and then choose the `.kicad_sch` 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 schematic.

The settings that are available to import are:

- Formatting preferences
- Field name templates
- BOM presets
- BOM format presets
- Violation severities
- Pin conflict map
- Net classes
- Bus alias definitions
- Text variables

Opening legacy schematics

Modern versions of KiCad can always open projects created in older versions of KiCad. However, schematics created in some older versions of KiCad have special considerations that must be observed when opening them in order to prevent any data loss.

Opening KiCad 5.0 and 5.1 schematics

Modern versions of KiCad can open schematics created in versions prior to KiCad 6.0, but the cache library file (`<projectname>-cache.lib`) must be present to load the schematic correctly.

Since version 6.0, KiCad stores all symbols used in a project in the schematic. This means that you can open a schematic made in KiCad 6.0 or later on any computer, even if the libraries used in the project are not installed or have changed. Modern KiCad schematic files use the `.kicad_sch` extension.

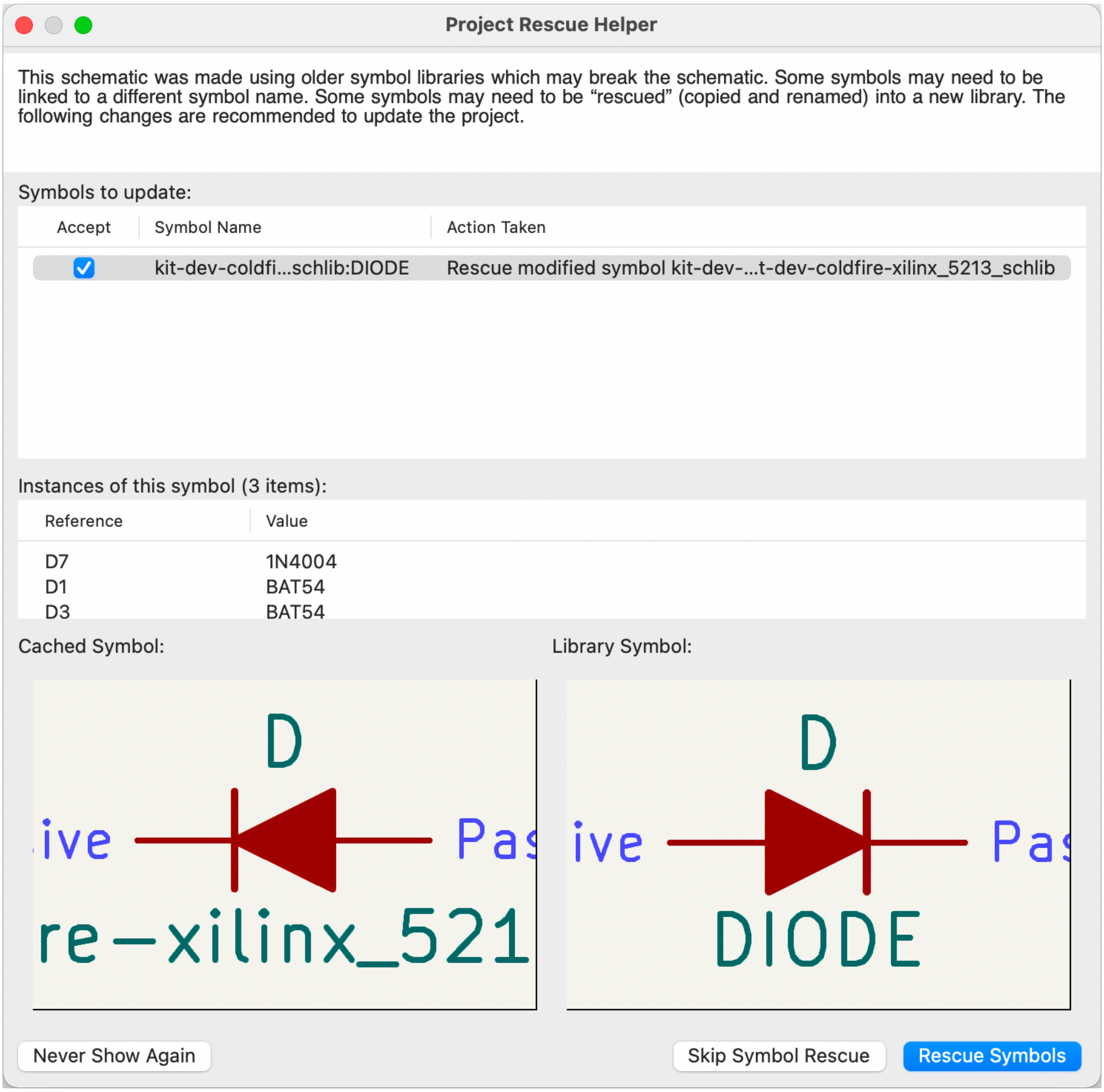
Prior to version 6.0, KiCad did not store symbols in the schematic. Instead, KiCad stored references to the symbols and their libraries. It also stored a copy of every symbol used by the project in a separate cache library file (`<projectname>-cache.lib`). As long as the cache library was included with the project, the project could be distributed without the system library files, because KiCad could load any needed symbols from the cache library as a fallback if the libraries referenced in the schematic were missing. Legacy KiCad schematic files use the `.sch` extension.

When you open a legacy schematic, KiCad will look in the cache library to find all of the symbols used in the schematic in the cache library. When you save the legacy schematic, KiCad will save it as a new file in the modern schematic format (`.kicad_sch`), with the necessary symbols embedded in the schematic itself. The original legacy schematic and the cache library will remain, unmodified, but they are no longer necessary once the schematic has been saved in the modern format.

NOTE

Projects created in KiCad prior to version 6.0 must have a cache library. If the cache library is missing, the schematic will lose symbol information if the system symbol libraries are modified, reorganized, moved, or deleted. The libraries included with legacy versions of KiCad are substantially different than the modern KiCad libraries, so in practice KiCad will almost always fail to open legacy projects unless the cache library is present.

When you open a legacy schematic, KiCad may display the **Project Rescue Helper** dialog. This means that one or more symbols in the cache library do not match the corresponding symbol in the external library. The dialog helps you "rescue" symbols from the cache library into your schematic, if desired. You can also open the rescue dialog at any time using **Tools** → **Rescue Symbols…**. The cache library file must be present in order to use the rescue tool.



Project Rescue Helper

This schematic was made using older symbol libraries which may break the schematic. Some symbols may need to be linked to a different symbol name. Some symbols may need to be "rescued" (copied and renamed) into a new library. The following changes are recommended to update the project.

Symbols to update:

Accept	Symbol Name	Action Taken
☑	kit-dev-coldfi...schlib:DIODE	Rescue modified symbol kit-dev-...t-dev-coldfire-xilinx_5213_schlib

Instances of this symbol (3 items):

Reference	Value
D7	1N4004
D1	BAT54
D3	BAT54

Cached Symbol:

Library Symbol:

The rescue dialog lists all symbols that don't match between the cache library and the external symbol library. The discrepancy can be because:

- the cached symbol or the library symbol has been modified, so the two symbols no longer match, or
- the cached symbol does not have a corresponding symbol in the symbol library, because the symbol or library was moved, renamed, deleted, or is not present on the current computer.

For each symbol in the list, selecting the symbol displays the reference designator and value for each instance of the symbol, and shows a visual preview of the symbol. If a corresponding symbol exists in the system symbol library, the dialog shows both copies of the symbol for comparison. If the symbol only exists in the cache library, the dialog only shows the cached symbol.

In this example, the project originally used a diode with the cathode facing left, but the library now contains one with the cathode facing right. This change would break the design, so it would be important to use the cached symbol as the original designer intended.

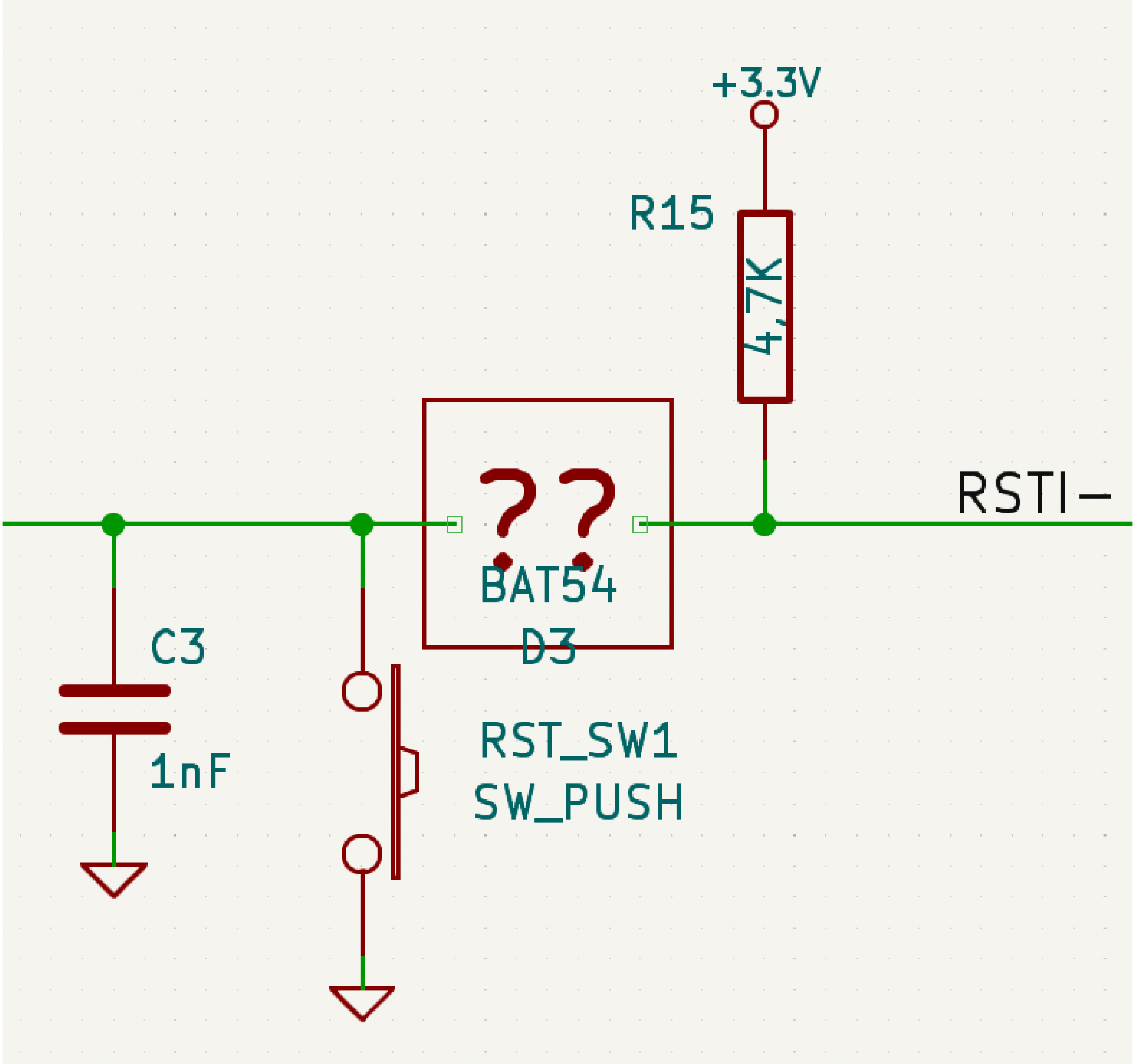
Pressing **Rescue Symbols** here will cause the selected symbols from the cache library to be saved into a special `rescue` library (`<projectname>-rescue.kicad_sym`). The corresponding symbols in the schematic will be updated to use the newly rescued symbols. Any unselected symbols will not be rescued, but their symbol linkage can be updated in the schematic later.

Alternatively, pressing **Skip Symbol Rescue** will exit the dialog without rescuing any symbols. KiCad will use the versions of the symbols found in the external libraries. You can run the rescue function again with **Tools** → **Rescue Symbols...**, or manually edit symbol linkage in the symbol's properties.

If you would prefer not to see this dialog, you can press **Never Show Again**. This has the same effect as pressing **Skip Symbol Rescue** for the current schematic and all future schematics.

If a symbol in a legacy schematic cannot be found in either the cache library or the external library, KiCad cannot rescue that symbol. A placeholder symbol is inserted into the schematic in its place, as shown below.

You can attempt to remap these orphaned symbols using the **Change Symbols** or **Edit Symbol Library Links** dialogs, but either option may require manual corrections to the schematic. These tools are explained in more detail in the Updating and exchanging symbols section.



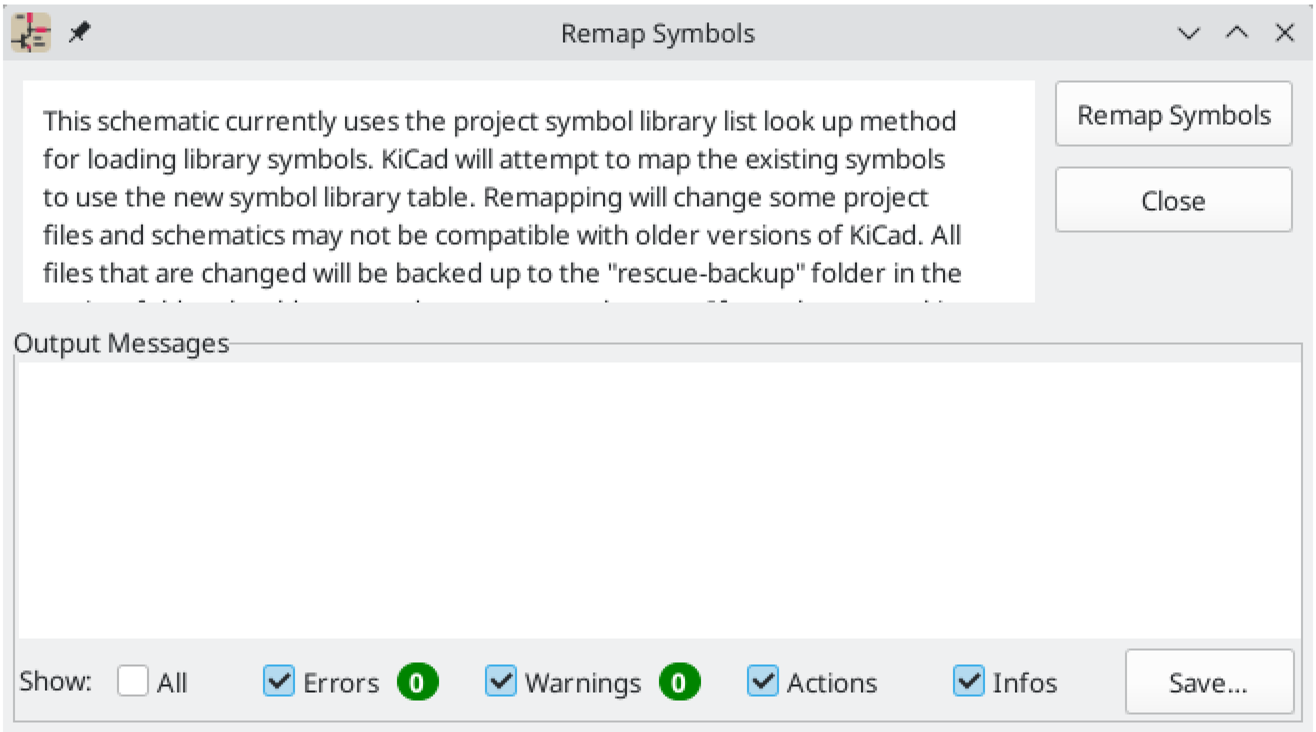
Opening pre-5.0 schematics

Modern versions of KiCad can open schematics created in versions prior to KiCad 5.0, but you will need to go through a symbol remapping process to open the schematic without losing symbol information.

Since version 5.0, KiCad schematics refer to specific symbols using both the symbol and library name. Even if multiple libraries each contain a symbol with the same name, the designer's intended symbol is unambiguously specified.

Prior to version 5.0, KiCad schematics stored only the symbol name, not the library name. Symbols in the schematic were indirectly mapped back to the original library by searching through the project's library list for a matching symbol. When you open a pre-5.0 schematic, KiCad will attempt to automatically "remap" the symbols so that each bare symbol name is replaced with a fully-specified symbol library and symbol name pair. The original schematics will be backed up in a `rescue-backup` folder.

You can skip the automatic remapping, but you will need to remap the symbols yourself using the Change Symbols dialog. You can also re-run the Remap Symbols tool using **Tools → Remap Legacy Library Symbols...**.


This schematic currently uses the project symbol library list look up method for loading library symbols. KiCad will attempt to map the existing symbols to use the new symbol library table. Remapping will change some project files and schematics may not be compatible with older versions of KiCad. All files that are changed will be backed up to the "rescue-backup" folder in the

Show: ☐ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

Multiple sheets and hierarchical schematics

Schematics can contain multiple sheets, which can be arranged and connected to each other in several ways. Schematic sheets are organized in a hierarchy, with some sheets organized at the top level of the hierarchy ("top-level sheets"), and others organized as subsheets of top-level sheets or other subsheets ("hierarchical sheets"). Hierarchical sheets are drawn inside another sheet and are organized under that sheet in the schematic hierarchy. Hierarchical sheets can be included in a hierarchy multiple times, if desired.

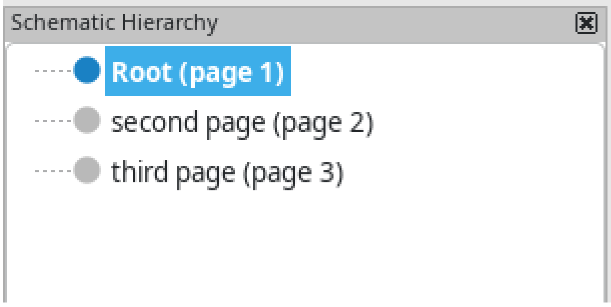
> **NOTE**
>
> Before KiCad 10.0, all multi-sheet schematics in KiCad were hierarchical. Schematics could only have a single top-level sheet, and all other sheets were created as subsheets of the root sheet or other subsheets. Beginning in KiCad 10.0, schematics can have multiple top-level sheets.

Creating a multi-sheet schematic starts from the first sheet. The process is to create a new sheet, either as a top-level sheet or a hierarchical sheet, then draw the circuit in the subsheet and make the necessary electrical connections between sheets. Connections can be made between nets in different sheets using global labels. For hierarchical sheets, you can also make connections from a subsheet to the parent sheet using hierarchical labels and hierarchical sheet pins.

Carefully considering the best way to organize sheets in a schematic, including whether to draw the schematic hierarchically, improves schematic legibility and reduces repetitive drawing.

Adding top-level sheets to a design

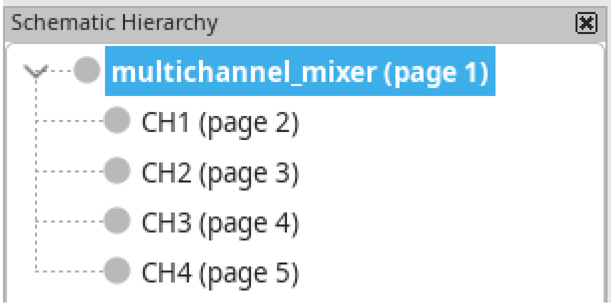
The most basic way to add new sheets to a design in KiCad is to add new top-level sheets. This puts new sheets at the same level of the hierarchy as the existing root sheet, creating a flat schematic where sheets are at the same level rather than nested. Unlike hierarchical subsheets, top-level sheets are not subsheets and are not displayed as sheet symbols in another sheet. You cannot make hierarchical connections between top-level sheets; net connections between top-level sheets can only be made using global labels and global power symbols. Each top-level sheet is saved as a separate `.kicad_sch` file.



To create a new top-level sheet, right click in the Schematic Hierarchy panel and choose **New Top-Level Sheet** (if the Schematic Hierarchy panel is not shown, you can show it with **View** → **Panels** → **Hierarchy Navigator**). After choosing a sheet name for the new sheet, the sheet is added to the hierarchy. You can choose a new page number for the sheet by right clicking it in the hierarchy and choosing **Edit Page Number** or rename the sheet by right clicking and choosing **Rename**.

Adding hierarchical subsheets to a design

Another way to add additional sheets to a design is using hierarchical subsheets. Subsheets are drawn as a sheet symbol inside of another sheet. The new subsheet is placed beneath the parent sheet in the sheet hierarchy. A net in a sheet can be connected to a subsheet, and vice versa, through a hierarchical sheet pin at the parent sheet level and a matching hierarchical label in the subsheet. Like top-level sheets, each new hierarchical subsheet is saved as a separate `.kicad_sch` file.



Schematic Hierarchy

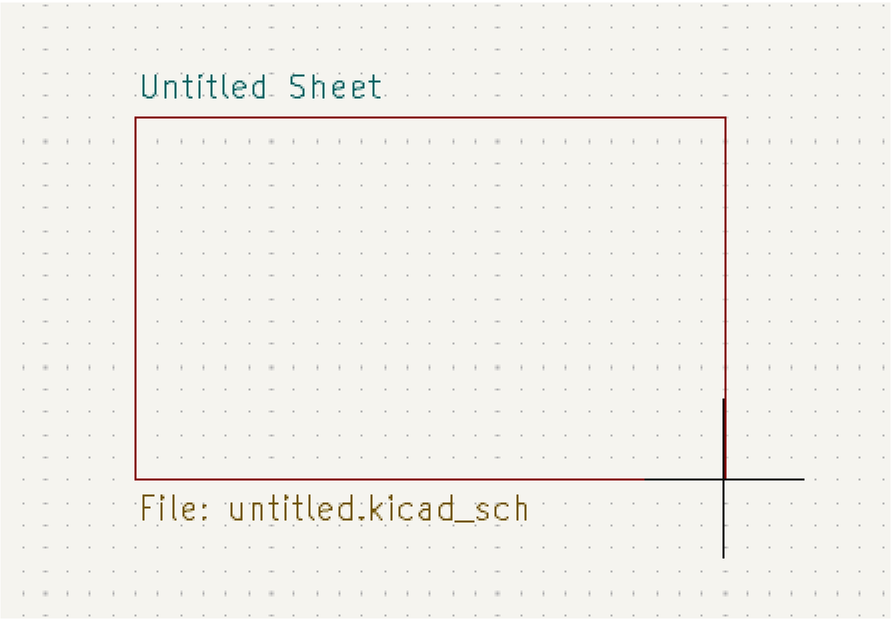
- multichannel_mixer (page 1)
  - CH1 (page 2)
  - CH2 (page 3)
  - CH3 (page 4)
  - CH4 (page 5)

When you add a hierarchical subsheet, you can either create a new sheet or reuse an existing sheet. By reusing an existing sheet, you can use the sheet multiple times in a design. The existing sheet file is used for each instantiation; no new sheet file is created when it is reused. Any edits to the contents of one hierarchical sheet instantiation also affect the other instantiations. The only thing that varies between instantiations of a hierarchical sheet are the reference designators of the symbols in the sheet, which are unique between instantiations.

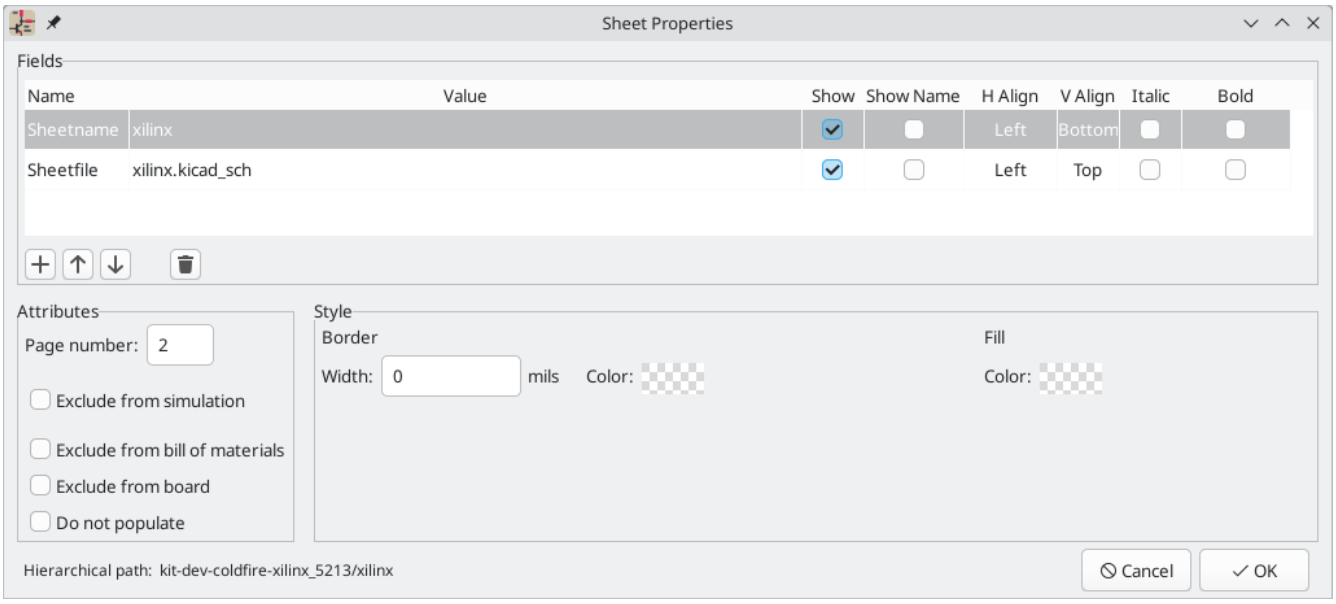
NOTE

Sheet files can be shared between multiple projects to allow design reuse between projects. However, this is not recommended due to path portability concerns and the risk of unintentionally changing other projects while editing a shared sheet.

You can add a hierarchical subsheet to a design with the Add Hierarchical Sheet tool (S hotkey, or the button in the right toolbar). Launch the tool, then click twice in the canvas to draw the upper left and lower right corners of the subsheet symbol (you can also click and drag). Make the sheet outline large enough to fit the hierarchical pins you will add later.



The Sheet Properties dialog will appear and prompt you for a sheet name and filename.



Fields

Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Sheetname	xilinx	☑	□	Left	Bottom	□	□
Sheetfile	xilinx.kicad_sch	☑	□	Left	Top	□	□

Attributes

Page number: 2

□ Exclude from simulation

□ Exclude from bill of materials

□ Exclude from board

□ Do not populate

Style

Border

Width: 0 mils Color: ▨

Fill

Color: ▨

Hierarchical path: kit-dev-coldfire-xilinx_5213/xilinx

The **sheet name** must be unique, as it is used in the full net name for any nets in the subsheet. For example, a net with the local label `net1` in the sheet `sheet1` would have a full net name of `/sheet1/net1`. The sheet name is also used to refer to the sheet in various places in the GUI, including the title block and the hierarchy navigator. The sheet name can also be changed in the hierarchy navigator.

The **sheet file** specifies the file that the new sheet will be saved to or loaded from. The path to the sheet file can be relative or absolute. It is usually preferable to save subsheet files in the project directory and use a relative path so that the project is portable.

A single sheet file can be used more than once in a project by specifying the same filename for each repeated sheet; the circuit drawn in the sheet will be instantiated once per usage, and any edits in once instance will be reflected in the other instances.

The sheet's **page number** is configurable here. The page number is displayed in the sheet title block and the hierarchy navigator, and sheets are sorted by page number in the hierarchy navigator and when printing or plotting. The sheet number can also be changed in the hierarchy navigator or for the current page with **Edit → Edit Sheet Page Number...**.

Other attributes for sheets are **Exclude from simulation**, **Exclude from bill of materials**, **Exclude from board**, and **Do not populate**. When any of these attributes are set for a sheet, they are inherited by all symbols in that sheet, as if they were set on the symbols themselves.

Several graphical options are also available. **Border width** sets the width of the border around the sheet shape. **Border color** and **Background fill** set the color for the border and fill of the sheet shape, respectively. If no color is set, a checkerboard swatch is shown and the default values from the color theme are used.

Sheets support arbitrary custom fields, which can be added and removed with the + and 🗑 buttons, respectively. Sheet fields can be optionally displayed on the schematic by checking their **Show** box, and they can be accessed from inside the sheet or in other sheet fields using text variables.

The Sheet Properties dialog can be accessed at any time by selecting a sheet symbol and using the E hotkey, or by right-clicking on a sheet symbol and selecting **Properties...**.

Once a new sheet has been added, you can open it to draw a circuit in it as you would in any other schematic sheet. You can also drag and drop a selection of existing circuit elements onto the child sheet symbol, which moves the selection into the child sheet.

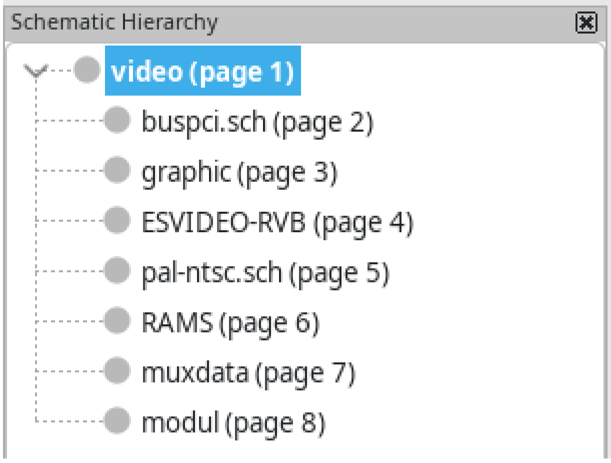
Navigating between sheets

You can enter a hierarchical sheet from the parent sheet by double-clicking the child sheet's shape, or right-clicking the child sheet and selecting **Enter Sheet**.

Return to the parent sheet by using the ⬆ button in the top toolbar, or by right-clicking in an empty part of the schematic and clicking **Leave Sheet**.

You can jump to the next sheet with the ➡ button, or to the previous sheet with the ⬅ button.

Alternatively, you can jump to any sheet with the hierarchy navigator. To open the hierarchy navigator, click the ☰ button in the left toolbar. The hierarchy navigator docks at the left of the screen. Each sheet in the design is displayed as an item in the tree. Clicking a sheet name opens that sheet in the editing canvas. You can also use the hierarchy navigator to rename or renumber a sheet by right clicking on the sheet name and selecting **Edit page number** or **Rename**.



Schematic Hierarchy

- video (page 1)
  - buspci.sch (page 2)
  - graphic (page 3)
  - ESVIDEO-RVB (page 4)
  - pal-ntsc.sch (page 5)
  - RAMS (page 6)
  - muxdata (page 7)
  - modul (page 8)

Electrical connections between sheets

Electrical connections between sheets are made with labels. There are several kinds of labels in KiCad, each with a different connection scope.

- **Local labels** only make connections within a sheet. Therefore local labels cannot be used to connect between sheets. Local labels are added with the A button.
- **Global labels** make connections anywhere in a schematic, regardless of sheet. Therefore global labels can be used to make connections between sheets whether the sheets are top-level or hierarchical. Global labels are added with the A▷ button.
- **Hierarchical labels** in a hierarchical subsheet connect to **hierarchical sheet pins** accessible in the parent sheet. Hierarchical designs rely on hierarchical labels and pins to make connections between parent sheets and child sheets. You can think of sheet pins as defining the interface for a subsheet; hierarchical labels within the child sheet connect to corresponding sheet pins which are visible in the parent sheet. Hierarchical labels are added inside a child sheet using the A◇ button.

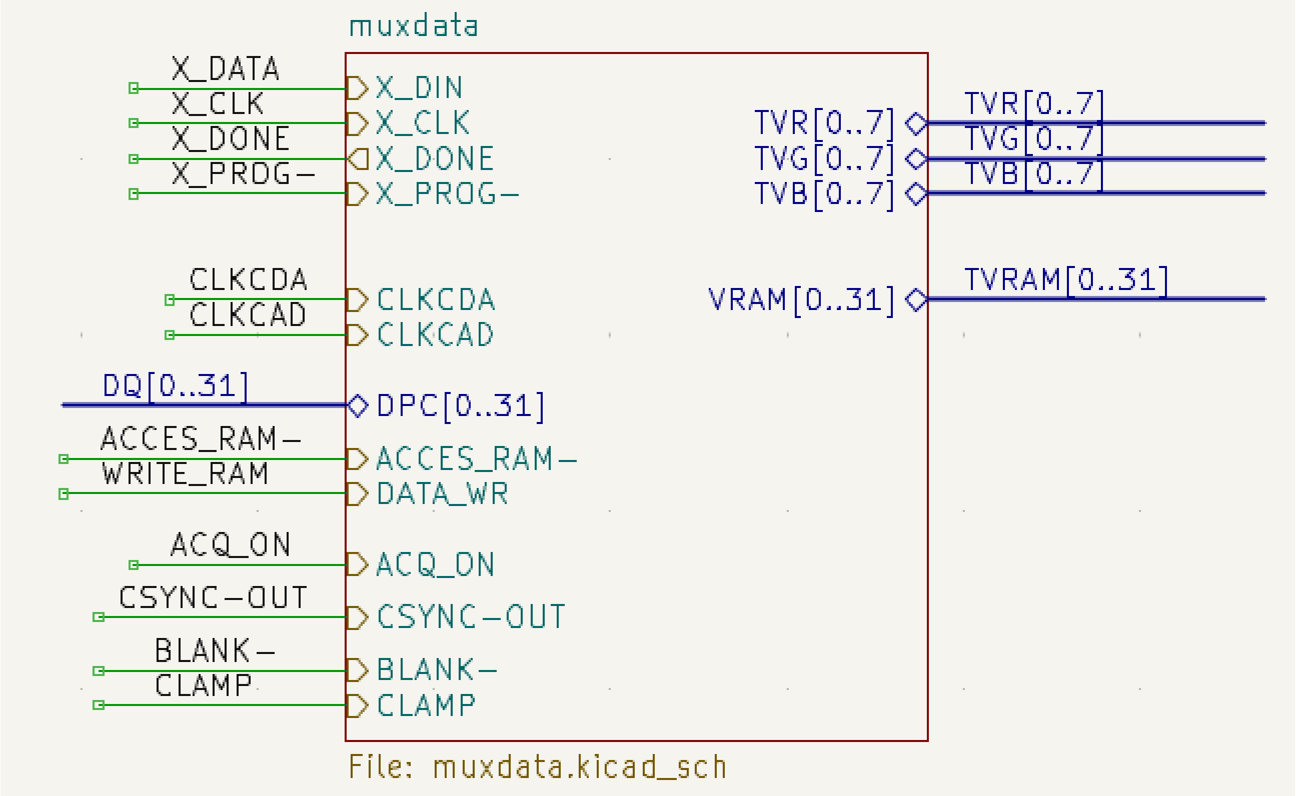
NOTE Labels that have the same name will connect, regardless of the label type, if they are in the same sheet.

NOTE Hidden power pins can also be considered global labels, because they connect anywhere in the schematic hierarchy.

Hierarchical sheet pins

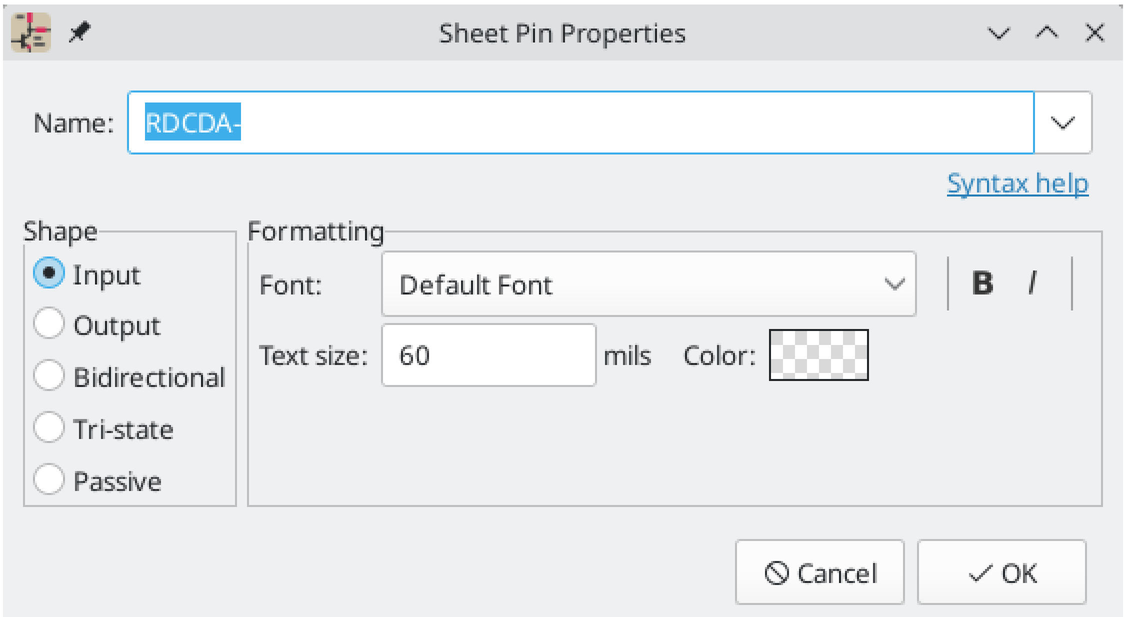
After placing hierarchical labels within a subsheet, matching hierarchical sheet pins can be added to the subsheet symbol in the parent sheet. You can then make connections to the hierarchical pins with wires, labels, and buses. Hierarchical sheet pins in a subsheet symbol are connected to the matching hierarchical labels in the subsheet itself.

NOTE Hierarchical labels must be defined in the subsheet before the corresponding hierarchical sheet pin can be imported in the sheet symbol.



For every hierarchical label in the subsheet, add the corresponding hierarchical pin onto the sheet symbol by clicking the button in the right toolbar, then clicking on the sheet symbol. A sheet pin for the first unmatched hierarchical label will be attached to the cursor, where it can be placed anywhere along the border of the sheet symbol. Clicking again with the tool will continue to add additional sheet pins until all of the hierarchical labels in the subsheet have a matching sheet pin on the sheet symbol. Sheet pins can also be imported by selecting **Place Sheet Pin** in a sheet symbol's right-click context menu.

You can edit the properties of a sheet pin in the Sheet Pin Properties dialog. Open this dialog by double-clicking a sheet pin, selecting a sheet pin and using the E hotkey, or right-clicking a sheet pin and selecting **Properties...**.


Sheet Pin Properties

Name: RDCDA-

Shape

● Input

○ Output

○ Bidirectional

○ Tri-state

○ Passive

Formatting

Font: Default Font

Text size: 60 mils

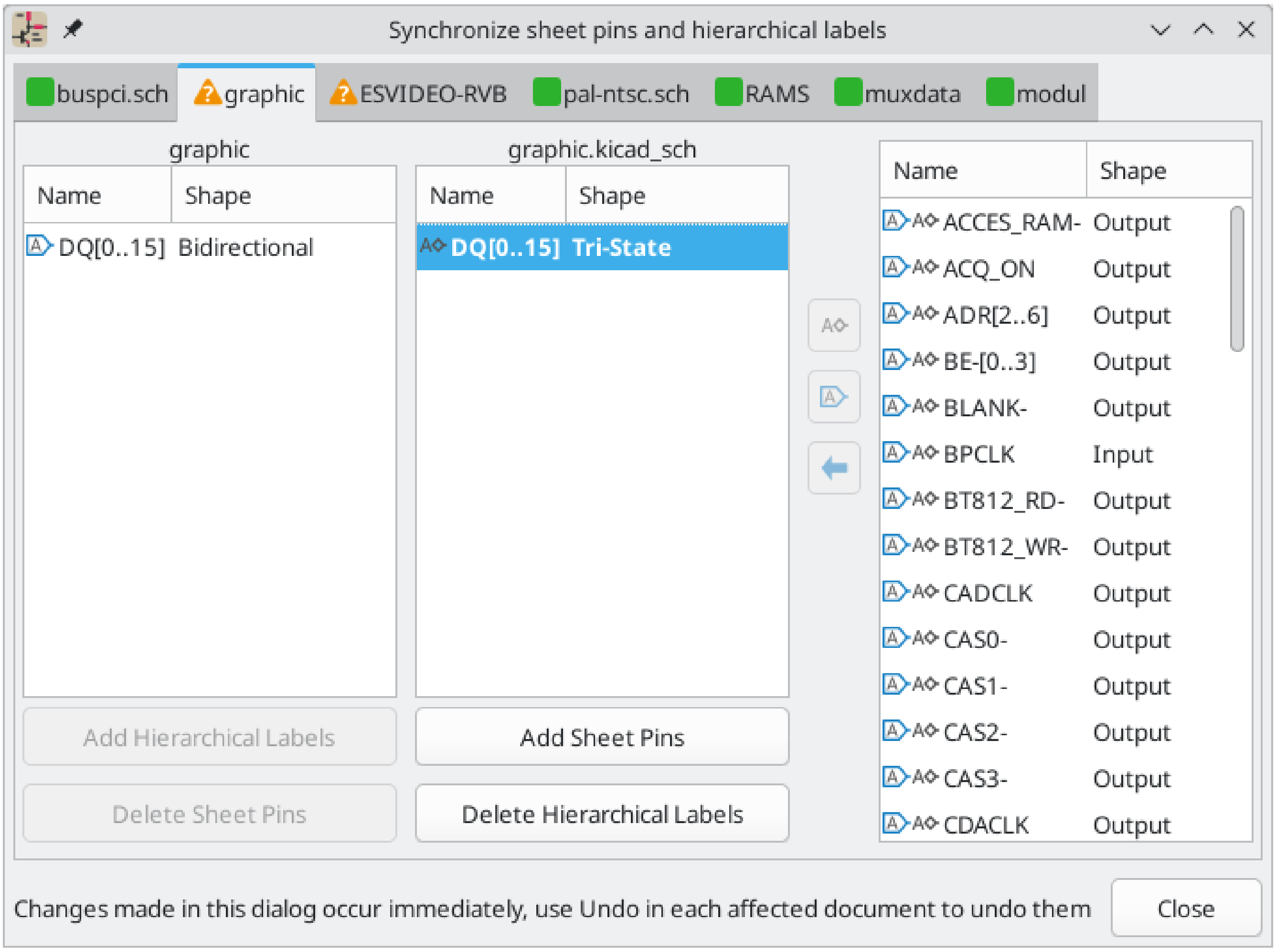
Color: ▨

The sheet pin's **name** can be edited in the textbox or by selecting from the dropdown list of hierarchical labels in the subsheet. A sheet pin's name has to match the corresponding hierarchical label in the subsheet, so if you change a pin name you must change the label name as well.

Shape changes the shape of the sheet pin, and has no electrical effect. It can be set to Input, Output, Bidirectional, Tri-state, or Passive. The pin's **font**, **text size**, **color**, and emphasis (bold or italic) can also be changed.

Syncing sheet pins

Another way to manage the connections between hierarchical labels and sheet pins is to use the Sync Sheet Pins tool. Launch this tool using the button in the right toolbar or with **Sync Sheet Pins** in a sheet symbol's right click context menu.



graphic

Name	Shape
DQ[0..15]	Bidirectional

graphic.kicad_sch

Name	Shape
DQ[0..15]	Tri-State

Name	Shape
ACCES_RAM-	Output
ACQ_ON	Output
ADR[2..6]	Output
BE-[0..3]	Output
BLANK-	Output
BPCLK	Input
BT812_RD-	Output
BT812_WR-	Output
CADCLK	Output
CAS0-	Output
CAS1-	Output
CAS2-	Output
CAS3-	Output
CDACLK	Output

Changes made in this dialog occur immediately, use Undo in each affected document to undo them

This dialog shows the hierarchical labels and hierarchical sheet pins for each hierarchical sheet. If the tool was launched from the context menu of a sheet symbol, only one tab will be available, with the labels and sheet pins for that specific sheet. If the tool was started globally, i.e. with the button or with **Place → Sync Sheet Pins**, a tab will be shown for each hierarchical sheet. If a sheet was selected in the editing canvas when the tool was launched, the dialog will open to that sheet's tab.

The icon in each tab indicates whether the hierarchical sheet pins on the sheet symbol are correctly matched with the hierarchical labels inside the sheet. If the tab has a ⚠ icon, then there is a hierarchical label in the sheet without a matching sheet pin, or a sheet pin without a corresponding hierarchical label, or both. If the tab has a 🟩 icon, then the hierarchical labels and hierarchical sheet pins are matched up correctly. Sheet pins and labels are considered matching if they have the same name and the same graphic shape (input, output, bidirectional, tri-state, or passive).

The column on the left lists sheet pins for the current sheet that do not have a corresponding hierarchical label in the sheet. The middle column lists hierarchical labels in the current sheet that do not have a corresponding hierarchical sheet pin on the sheet symbol. The right column lists pairs of matching sheet pins and hierarchical labels. The name of each pin or label is shown along with its graphic shape.

If you click the **Add Hierarchical Labels** button, new hierarchical labels corresponding to the selected sheet pins will be created for you to place sequentially in the sheet. The selected sheet pins are then removed from the left column and added to to the right column for matching sheet pins and labels. Clicking **Delete Sheet Pins** will delete the selected sheet pins from the sheet symbol.

If you click the **Add Sheet Pins** button, new sheet pins corresponding to the selected hierarchical labels will be created for you to place on the sheet symbol. The hierarchical labels are then removed from the middle column and added to the right column for matching sheet pins and labels. Clicking **Delete Hierarchical Labels** will delete the selected hierarchical labels from inside the sheet.

Clicking the A◇ button will match the selected sheet pin and hierarchical label by renaming the sheet pin to match the hierarchical label's name. Clicking the A▷ button will do the opposite, matching the selected sheet pin and hierarchical label by renaming the label to match the sheet pin.

Clicking the ⬅ button will unmatch a matched pair, moving both the sheet pin and the hierarchical label back to their respective unmatched columns. The unmatched sheet pin and hierarchical label can then be edited or rematched as desired.

Any changes made in the Sync Sheet Pins dialog are applied immediately, before the dialog is closed. To cancel a change made in the Sync Sheet Pins dialog, use Undo.

Multipage and hierarchical design examples

Multipage schematics in KiCad can be arranged in several ways. Which schematic organization is most appropriate depends on the design.

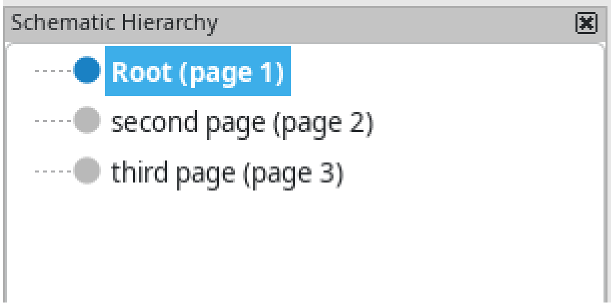
A design that conceptually contains nested blocks lends itself well to a hierarchical organization, especially if some of the blocks are repeated, because each block can be represented as a hierarchical subsheet. On the other hand, designs that conceptually don't contain a hierarchy might be better represented using top-level sheets. You can also both strategies in the same schematic, using multiple top-level sheets while also using hierarchical subsheets.

Some examples of different schematic organizations are shown below.

Flat schematics

Flat schematics contain multiple top-level sheets; each top-level sheet follows the previous sheet, and all top-level sheets are at the same level of the hierarchy. None of the top-level sheets are nested under another sheet.

In the example below there are three top-level sheets, each with a name and a page number. They are not hierarchically nested, so hierarchical connections with hierarchical labels and sheet pins are impossible. Any net connections between sheets must be made with global labels (and global power symbols).

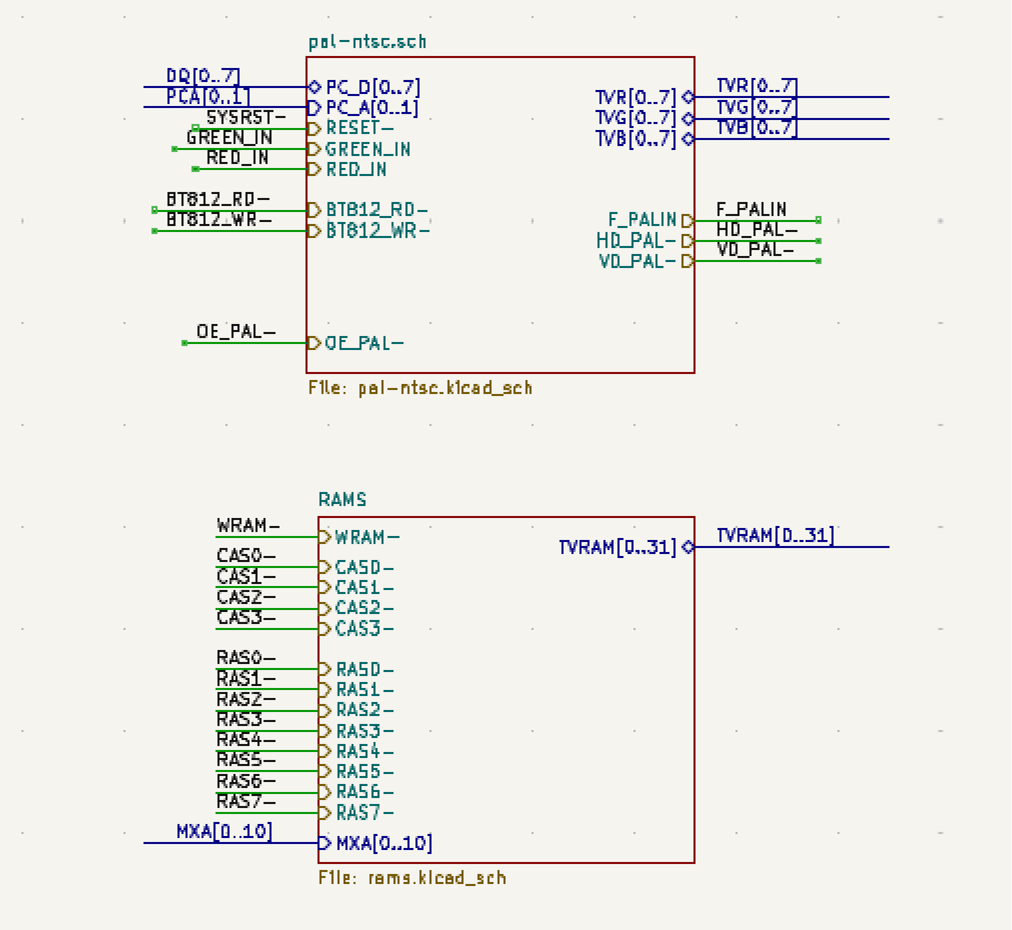


Schematic Hierarchy

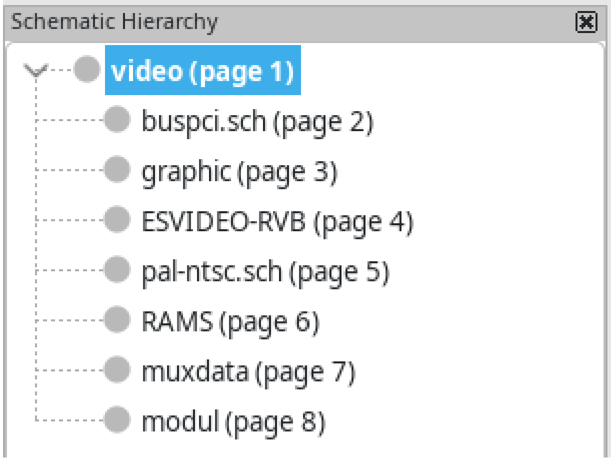
- Root (page 1)
- second page (page 2)
- third page (page 3)

Skematik Hirarkis

An example of a hierarchical design is the `video` demo project included with KiCad. The root sheet contains seven unique subsheets, each with hierarchical labels and sheet pins linking the sheets to each other in the root sheet. Each of the seven subsheets is used once; none are reused (see the next section for an example of reusing subsheets). Two of the subsheet symbols are shown below.



The subsheets are drawn as sheet symbols in the root sheet, meaning they are placed under the root sheet in the schematic hierarchy.

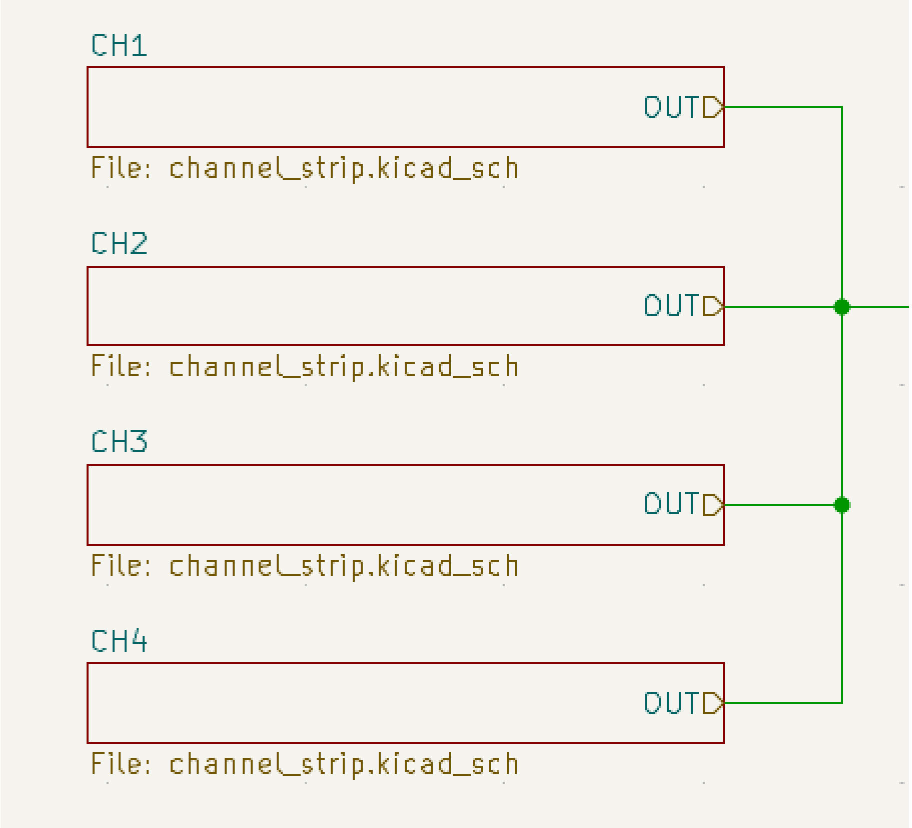


Schematic Hierarchy

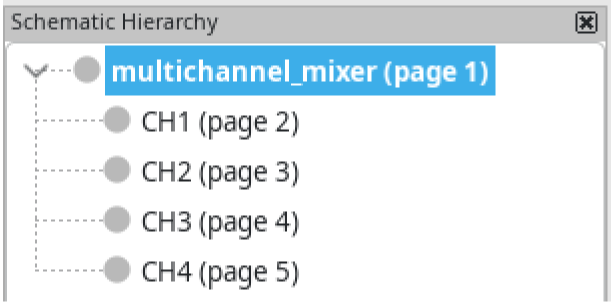
- video (page 1)
  - buspci.sch (page 2)
  - graphic (page 3)
  - ESVIDEO-RVB (page 4)
  - pal-ntsc.sch (page 5)
  - RAMS (page 6)
  - muxdata (page 7)
  - modul (page 8)

Reusing hierarchical subsheets

The `multichannel` demo project is an example of using a subsheet multiple times. The root sheet contains four subsheet symbols, which both refer to the same sheet file (`channel_strip.kicad_sch`). This allows the design to include four copies of the same audio mixer channel circuit. Although the four sheet symbols all refer to the same filename, the sheet names are unique (`CH1` , `CH2` , `CH3` , and `CH4`). Inside each subsheet the circuits are identical except for the reference designators, which as always are unique. The subsheets each have a single hierarchical label/pin connection to the root sheet, which contains the summing amplifier circuit that combines the output of each channel.


CH1
OUT
File: channel_strip.kicad_sch
CH2
OUT
File: channel_strip.kicad_sch
CH3
OUT
File: channel_strip.kicad_sch
CH4
OUT
File: channel_strip.kicad_sch

The subsheets are drawn as sheet symbols in the root sheet, meaning they are placed under the root sheet in the schematic hierarchy.

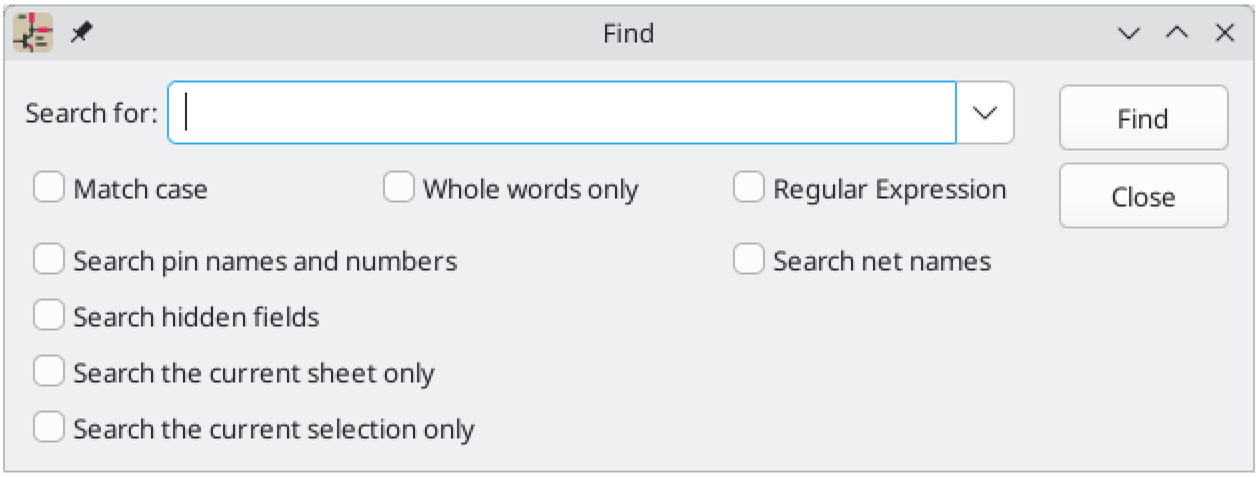

Schematic Hierarchy

- multichannel_mixer (page 1)
  - CH1 (page 2)
  - CH2 (page 3)
  - CH3 (page 4)
  - CH4 (page 5)

Inspecting a schematic

Find tool

The Find tool searches for text in the schematic, including reference designators, pin names, symbol 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.



Find

Search for:

☐ Match case ☐ Whole words only ☐ Regular Expression

☐ Search pin names and numbers ☐ Search net names

☐ Search hidden fields

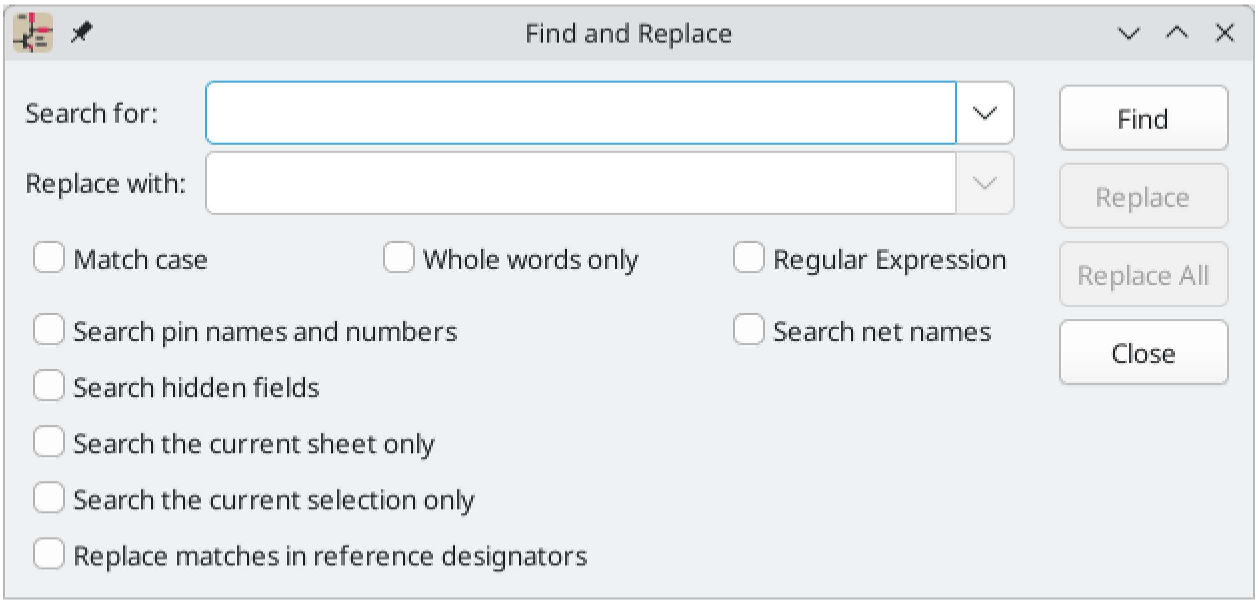
☐ Search the current sheet only

☐ Search the current selection only

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 schematic. When unselected, the search will match if the search term is part of a larger word in the schematic.
- **Regular Expression**: When selected, regular expressions can be used in the search terms.
- **Search pin names and numbers**: Selects whether the search should apply to pin names and numbers.
- **Search net names**: Selects whether the search should apply to net names (labels, symbol pins, sheet pins, and bus members).
- **Search hidden fields**: Selects whether the search should apply only to visible fields or if it should include hidden symbol fields.
- **Search the current sheet only**: Selects whether the search should be limited to the current schematic sheet.
- **Search the current selection only**: Selects whether the search should be limited to the current selection.

There is also a Find and Replace tool which is activated with the button in the top toolbar. This tool behaves the same as the Find tool, but additionally can replace some or all matches with different text.



☐ Match case ☐ Whole words only ☐ Regular Expression

☐ Search pin names and numbers ☐ Search net names

☐ Search hidden fields

☐ Search the current sheet only

☐ Search the current selection only

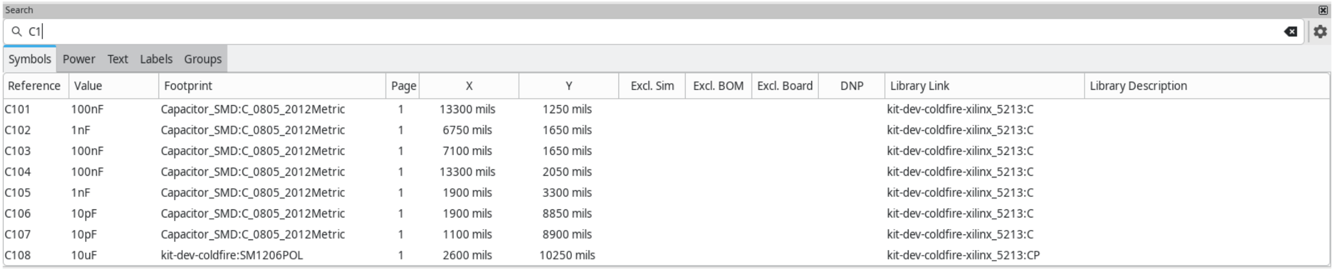
☐ Replace matches in reference designators

The Find and Replace tool has the same options as the Find tool, with one addition:

- **Replace matches in reference designators**: When selected, reference designators will be modified if they contain matching text. Otherwise reference designators will not be affected.

Search panel

The search panel is a docked panel that lists information about symbols, text, labels, and groups from the schematic. Show or hide the search panel with **View** → **Panels** → **Search** or use the `Ctrl` + `G` shortcut.



C1

Reference	Value	Footprint	Page	X	Y	Excl. Sim	Excl. BOM	Excl. Board	DNP	Library Link	Library Description
C101	100nF	Capacitor_SMD:C_0805_2012Metric	1	13300 mils	1250 mils					kit-dev-coldfire-xilinx_5213:C	
C102	1nF	Capacitor_SMD:C_0805_2012Metric	1	6750 mils	1650 mils					kit-dev-coldfire-xilinx_5213:C	
C103	100nF	Capacitor_SMD:C_0805_2012Metric	1	7100 mils	1650 mils					kit-dev-coldfire-xilinx_5213:C	
C104	100nF	Capacitor_SMD:C_0805_2012Metric	1	13300 mils	2050 mils					kit-dev-coldfire-xilinx_5213:C	
C105	1nF	Capacitor_SMD:C_0805_2012Metric	1	1900 mils	3300 mils					kit-dev-coldfire-xilinx_5213:C	
C106	10pF	Capacitor_SMD:C_0805_2012Metric	1	1900 mils	8850 mils					kit-dev-coldfire-xilinx_5213:C	
C107	10pF	Capacitor_SMD:C_0805_2012Metric	1	1100 mils	8900 mils					kit-dev-coldfire-xilinx_5213:C	
C108	10uF	kit-dev-coldfire:SM1206POL	1	2600 mils	10250 mils					kit-dev-coldfire-xilinx_5213:CP	

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 from the entire schematic are listed, not just items in the current sheet. Items are filtered based on their properties:

- Symbols and power symbols 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. Symbols are also filtered by their metadata (library link, description, and keywords) if **Search Metadata** is enabled in the ⚙ menu.
- Text (text and textboxes) is filtered by the text content.
- Labels are filtered by their netnames.
- Groups are filtered by the group name.

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 `/symbol value/`.

The displayed information depends on the item type:

- All items list their name and/or value, page number, and X/Y location in the sheet as applicable.
- Symbols additionally list their reference designator, footprint, attributes (Exclude from Simulation, Exclude from BOM, Exclude from Board, and Do Not Populate), library link (library name and symbol name), and description.
- Power symbols additionally list their reference designator.
- Text and labels additionally list their type, e.g. textbox or hierarchical.

When you click an item in the search panel, the schematic editor switches to the item's schematic sheet, and the item is selected in the editing canvas. Depending on what is configured in the ⚙ menu, the schematic 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.

Net highlighting

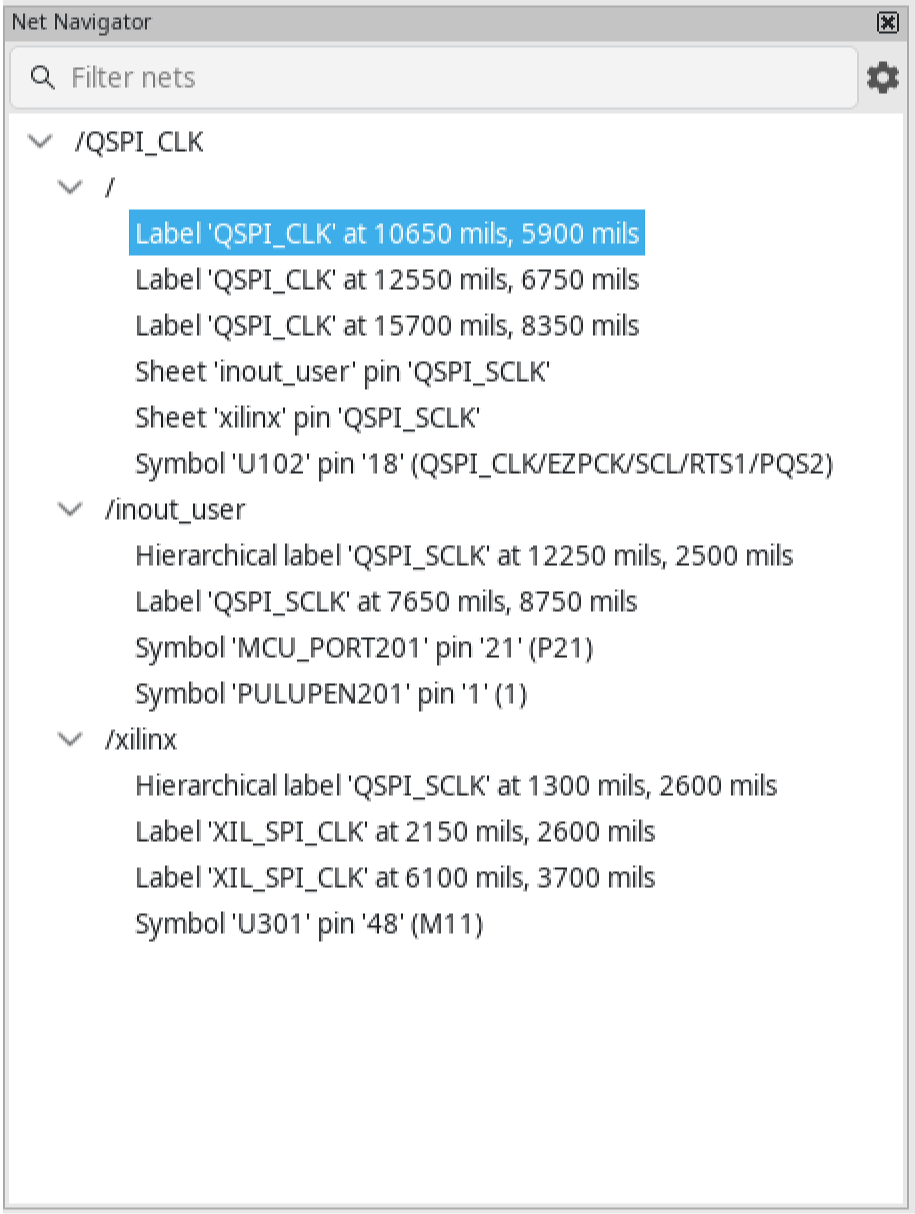
An electrical net can be highlighted in the schematic editor to visualize all of the places it appears in the schematic. Net highlighting can be activated in the Schematic Editor or by highlighting the corresponding net in the PCB editor when cross-probe highlighting is enabled (see below). When net highlighting is active, the highlighted net will be shown in a different color. By default this color is pink, but it is configurable in the Color section of the Preferences dialog.

Nets can be highlighted by clicking on a wire or pin using the Highlight Net tool in the right toolbar (). Alternatively, the Highlight Net hotkey (`) highlights the net under the cursor.

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 schematic. By default, Esc also clears net highlighting, but this can be disabled if desired in **Preferences** → **Schematic Editor** → **Editing Options**.

Net navigator

The net navigator is a docked panel that shows the location of every occurrence of a highlighted net in a schematic. Show or hide the net navigator with **View** → **Panels** → **Net Navigator**.



Net Navigator

Filter nets

- /QSPI_CLK
  - /
    - Label 'QSPI_CLK' at 10650 mils, 5900 mils
    - Label 'QSPI_CLK' at 12550 mils, 6750 mils
    - Label 'QSPI_CLK' at 15700 mils, 8350 mils
    - Sheet 'inout_user' pin 'QSPI_SCLK'
    - Sheet 'xilinx' pin 'QSPI_SCLK'
    - Symbol 'U102' pin '18' (QSPI_CLK/EZPCK/SCL/RTS1/PQS2)
  - /inout_user
    - Hierarchical label 'QSPI_SCLK' at 12250 mils, 2500 mils
    - Label 'QSPI_SCLK' at 7650 mils, 8750 mils
    - Symbol 'MCU_PORT201' pin '21' (P21)
    - Symbol 'PULUPEN201' pin '1' (1)
  - /xilinx
    - Hierarchical label 'QSPI_SCLK' at 1300 mils, 2600 mils
    - Label 'XIL_SPI_CLK' at 2150 mils, 2600 mils
    - Label 'XIL_SPI_CLK' at 6100 mils, 3700 mils
    - Symbol 'U301' pin '48' (M11)

When you highlight a net in the schematic, every place where that net is shown in the schematic is listed in the net navigator panel. All labels, symbol pins, and sheet pins connected to the net are listed. Each occurrence is sorted under its schematic sheet. Clicking on an occurrence displays that item in the editing canvas.

When no net is highlighted, the net navigator displays this information for all nets in the schematic.

NOTE The net navigator displays highlighted nets, not selected nets.

You can filter the displayed nets by typing part of a net name in the **Filter nets** textbox. If **Wildcard Search** is enabled in the ⚙ menu, you can use wildcards in the filter string: `*` matches any characters, and `?` matches any single character. If **Regex Search** is enabled instead, you can use regular expressions.

With the net navigator open and a net highlighted, you can quickly select various items on that net (net labels, sheet pins, and symbol pins) by selecting one of the items on the highlighted net in the schematic canvas and pressing `Tab`/`Shift`+`Tab` to cycle through the net items. Pressing `Tab` selects the next item on the net, while `Shift`+`Tab` selects the previous item.

Cross-probing from the PCB

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***.

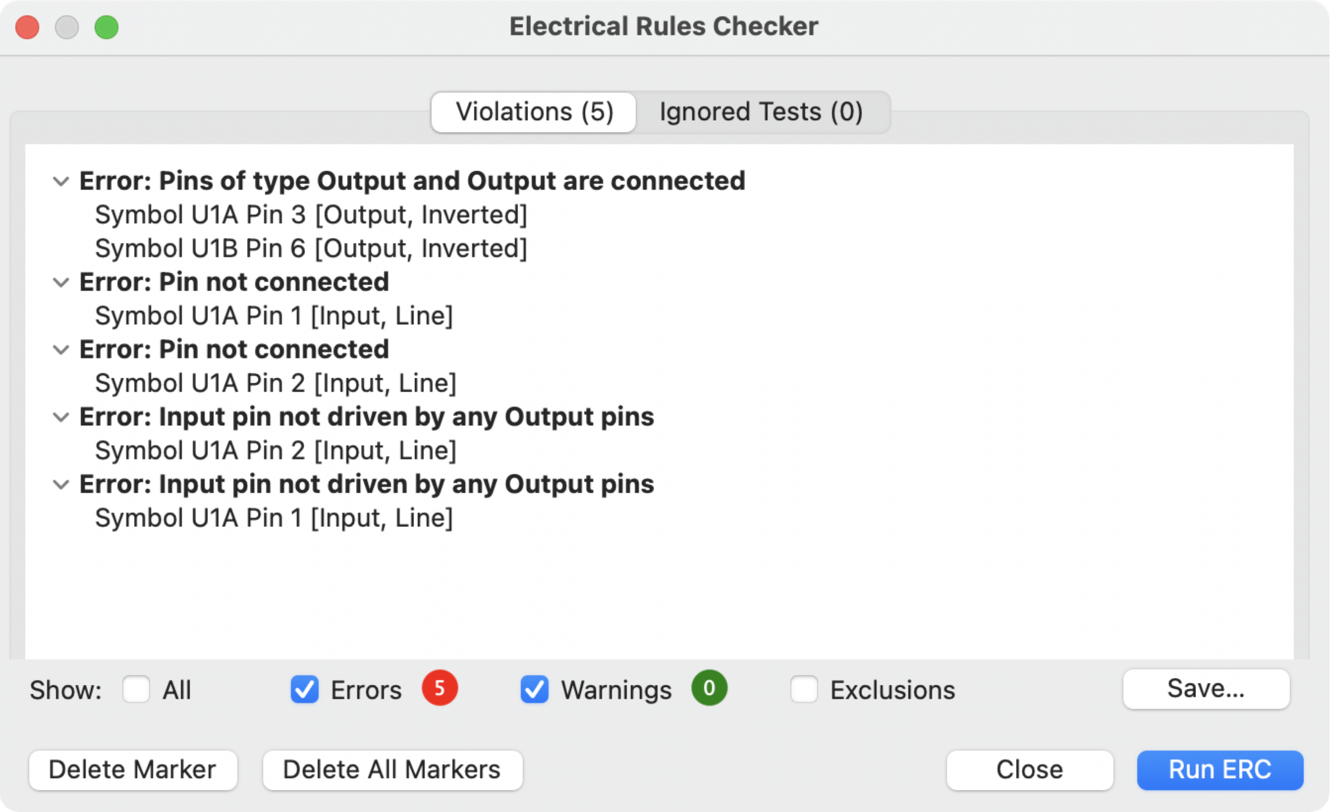
Highlight cross-probing allows you to highlight a net in the schematic and PCB at the same time. If the option "Highlight cross-probed nets" is enabled in the Display Options section of the Preferences dialog, highlighting a net or bus in the schematic editor will cause the corresponding net or nets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

Electrical rules checking

The Electrical Rules Checker (ERC) tool checks for certain errors in your schematic, such as unconnected pins, unconnected hierarchical symbols, shorted outputs or other illegal connections, etc. ERC violations are reported as errors or warnings depending on the severity of the issue detected.

ERC is imperfect and cannot detect all errors, but it can detect many common issues and oversights. All detected issues should be checked and addressed before proceeding. The quality of the ERC is directly related to the care taken in declaring electrical pin properties during symbol creation. If symbols are designed incorrectly, ERC will not report accurate information.

ERC can be started by clicking on the ☑ button in the top toolbar and clicking the **Run ERC** button.



Electrical Rules Checker

Violations (5) | Ignored Tests (0)

- Error: Pins of type Output and Output are connected
  - Symbol U1A Pin 3 [Output, Inverted]
  - Symbol U1B Pin 6 [Output, Inverted]
- Error: Pin not connected
  - Symbol U1A Pin 1 [Input, Line]
- Error: Pin not connected
  - Symbol U1A Pin 2 [Input, Line]
- Error: Input pin not driven by any Output pins
  - Symbol U1A Pin 2 [Input, Line]
- Error: Input pin not driven by any Output pins
  - Symbol U1A Pin 1 [Input, Line]

Show: □ All ☑ Errors 5 ☑ Warnings 0 □ Exclusions

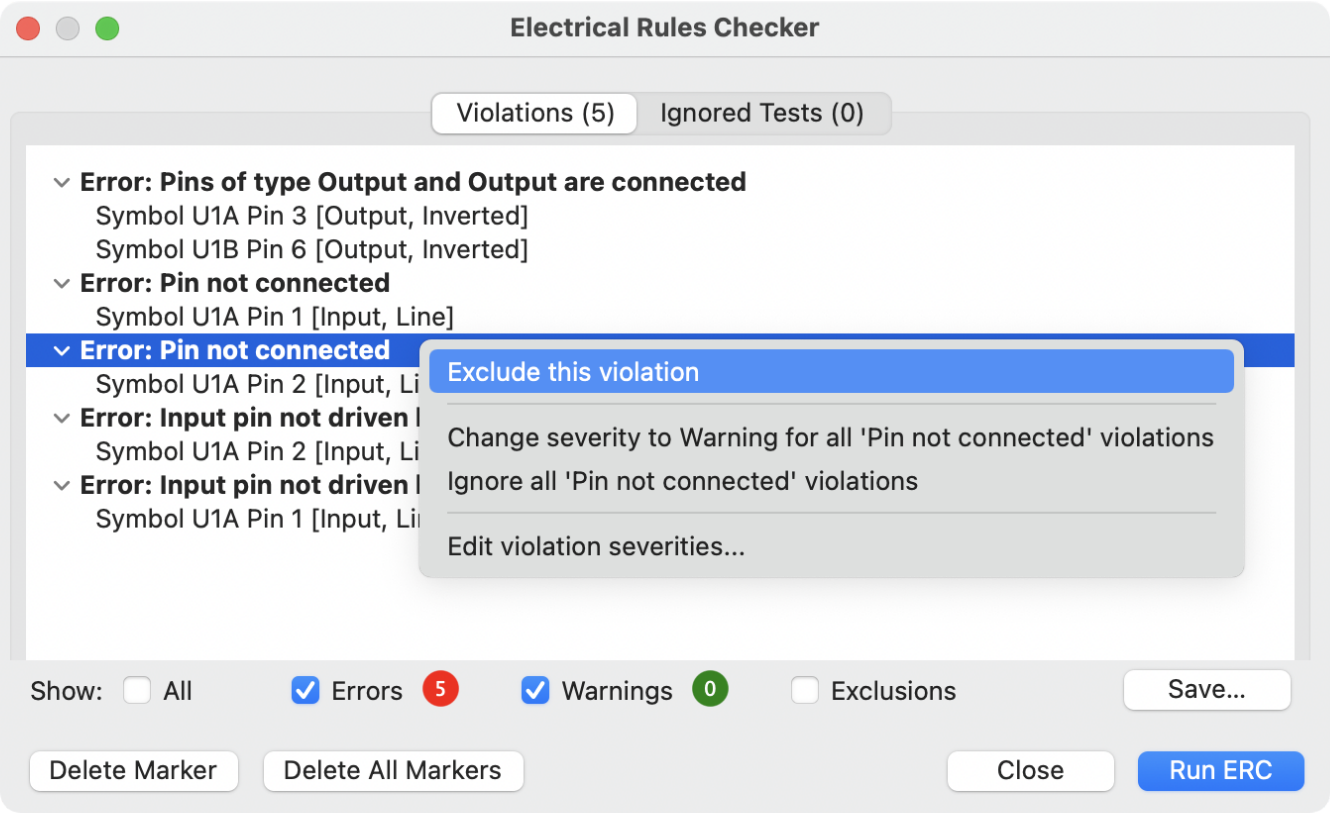
Any warnings or errors are reported in the **Violations** tab, and markers for each violation are placed in the schematic so that they point to the relevant part of the schematic. Warnings are indicated by yellow arrows, and errors have red arrows. Excluded violations are shown as green arrows. A list of the ignored tests are shown in the **Ignored Tests** tab. A report file in plain text format can be created after running DRC using the **Save...** button.

NOTE	Selecting a violation in the ERC window jumps to the selected violation marker in the schematic.

Certain types of violations have contextual actions in the context menu. For example, unannotated symbol violations have an action to run the annotation tool on the violating items. For schematic vs. library symbol mismatch violations, there is an action to run the Compare Symbol with Library tool and another action to update the symbol from the library. These actions can help to quickly fix or 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 ERC is run again, while clicking **Delete All Markers** will clear all violations until the next ERC run.

Violations can be right-clicked in the dialog to ignore them or change their severity:



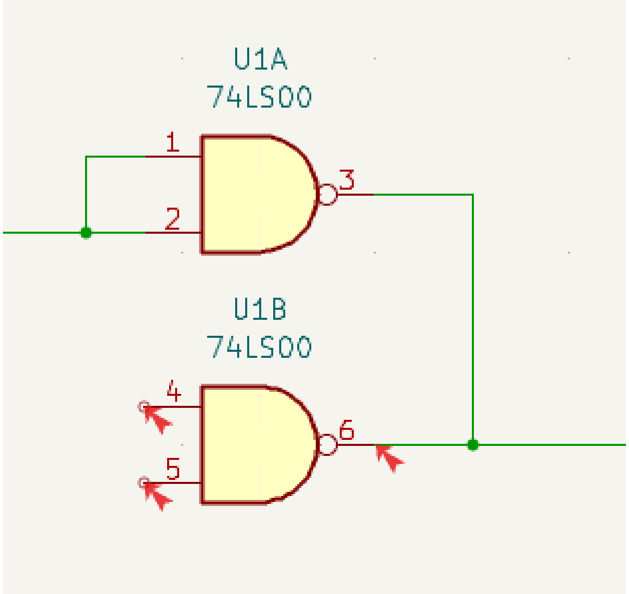
Error: Pins of type Output and Output are connected
Symbol U1A Pin 3 [Output, Inverted]
Symbol U1B Pin 6 [Output, Inverted]
Error: Pin not connected
Symbol U1A Pin 1 [Input, Line]
Error: Pin not connected
Symbol U1A Pin 2 [Input, Li
Error: Input pin not driven I
Symbol U1A Pin 2 [Input, Li
Error: Input pin not driven I
Symbol U1A Pin 1 [Input, Li

- **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...**.
- **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 Schematic Setup.
- **Edit violation severities...:** opens the Violation Severity panel in Schematic Setup, for editing the severities of all DRC violation types.

You can also exclude the selected marker with **Inspect** → **Exclude Marker**, and show or hide each category of marker (errors, warnings, and exclusions) with the **View** menu.

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.

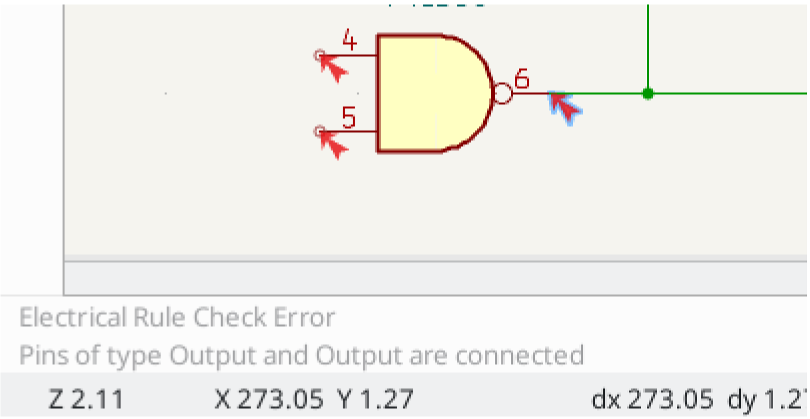
ERC example



There are three errors in the screenshot above.

- Two outputs have been connected together (red arrow at right).
- Two inputs have been left unconnected (red arrows at left). This is actually two errors per pin: each pin is unconnected, and each pin is an input pin that is not driven by an output pin.

Selecting an ERC marker displays a description of the violation in the message pane at the bottom of the window.



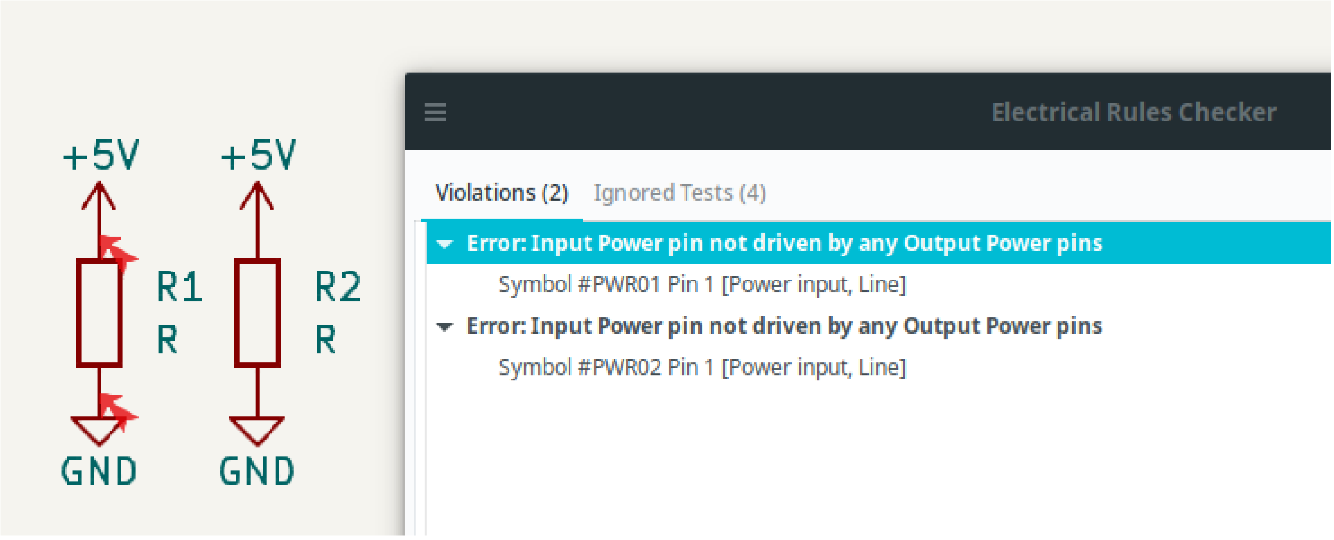
Electrical Rule Check Error
Pins of type Output and Output are connected

Z 2.11 X 273.05 Y 1.27 dx 273.05 dy 1.2

Per-net ERC markers

Some violations are reported only once per net. For example, the "Input Power pin not driven by any Output Power pins" error technically applies to each Input Power pin on an un-driven net, but only one marker is shown per net. This is to avoid producing a large number of markers for a single root cause. In this case, exactly where the marker is placed in the schematic is arbitrary and does not necessarily indicate the ideal location for fixing the issue.

In the example below, there are four un-driven Input Power pins (one per power symbol), but only two ERC markers. Adding two `PWR_FLAG` symbols, one to each un-driven net, will clear the errors. The "correct" location for the `PWR_FLAG` symbol depends on the schematic and may not be near the particular power symbol where the marker was placed, or even in the same schematic sheet.

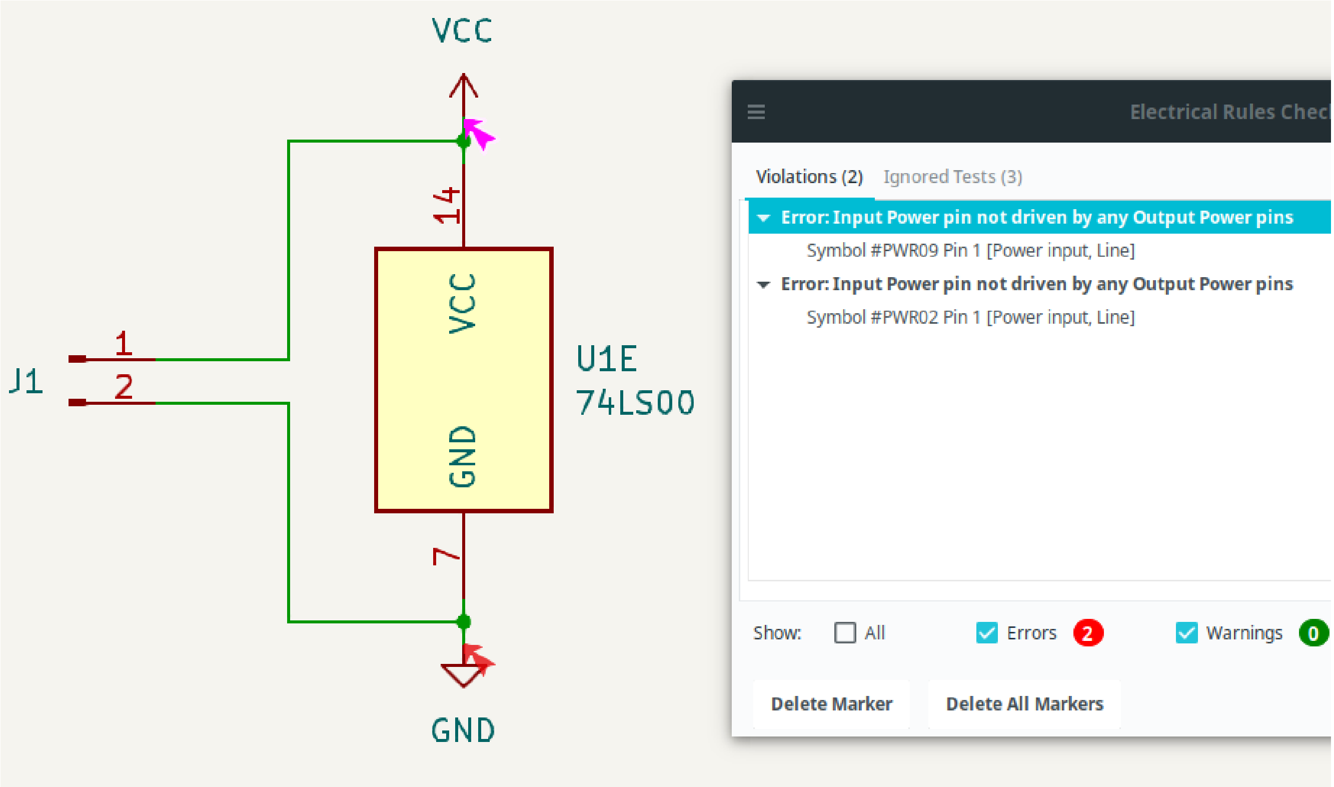


Error: Input Power pin not driven by any Output Power pins
Symbol #PWR01 Pin 1 [Power input, Line]
Error: Input Power pin not driven by any Output Power pins
Symbol #PWR02 Pin 1 [Power input, Line]

Power pins and power flags

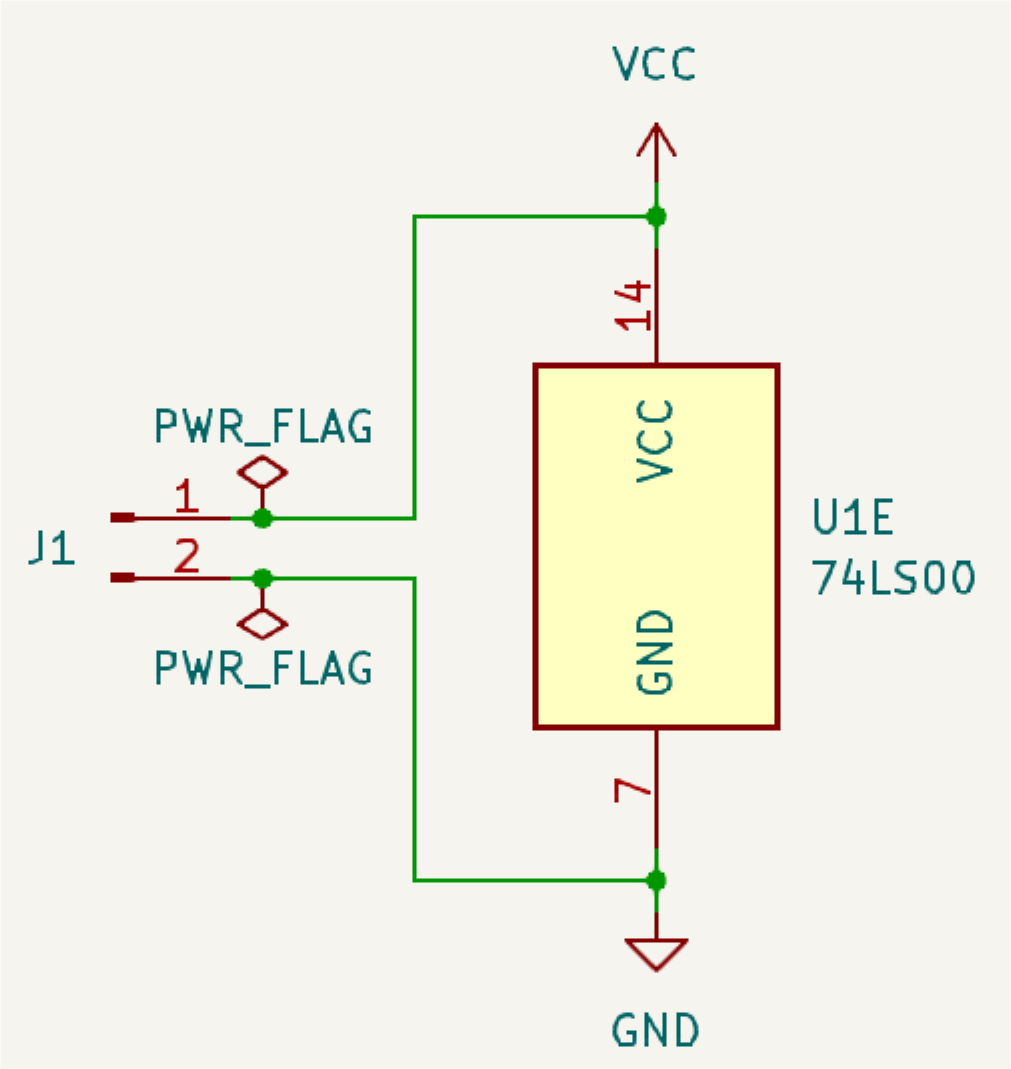
It is common to have an "Input Power pin not driven by any Output Power pins" error on power pins, as shown in the example below, even though the power pins seem to be properly connected to a power rail. This happens in designs where the power is provided through connectors or other components that are not marked as power outputs. In these cases ERC won't detect any Output Power pins connected to the net and will determine the Input Power pin is not driven by a power source.

For example, in the below schematic, `VCC` and `GND` are connected to a connector with passive pins, so the Input Power pins of `U1` are not considered driven by a power source.



Error: Input Power pin not driven by any Output Power pins
Symbol #PWR09 Pin 1 [Power input, Line]
Error: Input Power pin not driven by any Output Power pins
Symbol #PWR02 Pin 1 [Power input, Line]

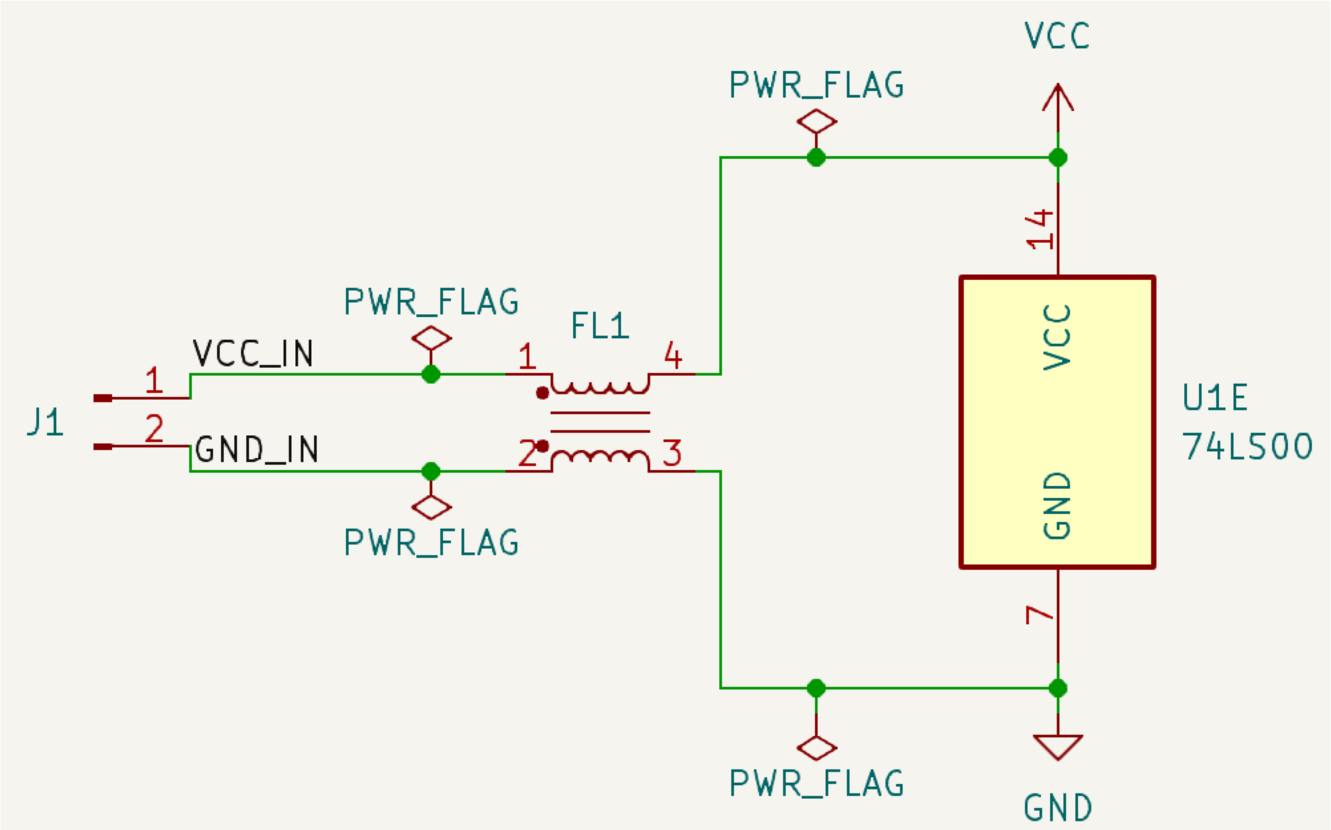
To avoid this warning, connect the net to `PWR_FLAG` symbol on such a power net as shown in the following example. The `PWR_FLAG` symbol is found in the `power` symbol library. Alternatively, connect any Output Power pin to the net; `PWR_FLAG` is simply a symbol with a single Output Power pin.



Ground nets often need a `PWR_FLAG` to be manually added, even when power rails do not, because voltage regulators have outputs declared as Output Power, but their ground pins are typically marked as Power Inputs. Therefore grounds are not considered driven by these voltage regulators. This is so multiple regulators can share a common ground without errors caused by two Output Power pins being connected together.

Power nets do not "jump" across components, so, components like passive inline filtering elements may need a `PWR_FLAG` on their downstream side to indicate that the net is still connected to a power source.

In the below schematic, although there is a `PWR_FLAG` on the nets connected to the connectors (`VCC_IN` and `GND_IN`), a `PWR_FLAG` is also needed to mark each of `VCC` and `GND` as power nets, because the filter `FL1` has only Passive pins.

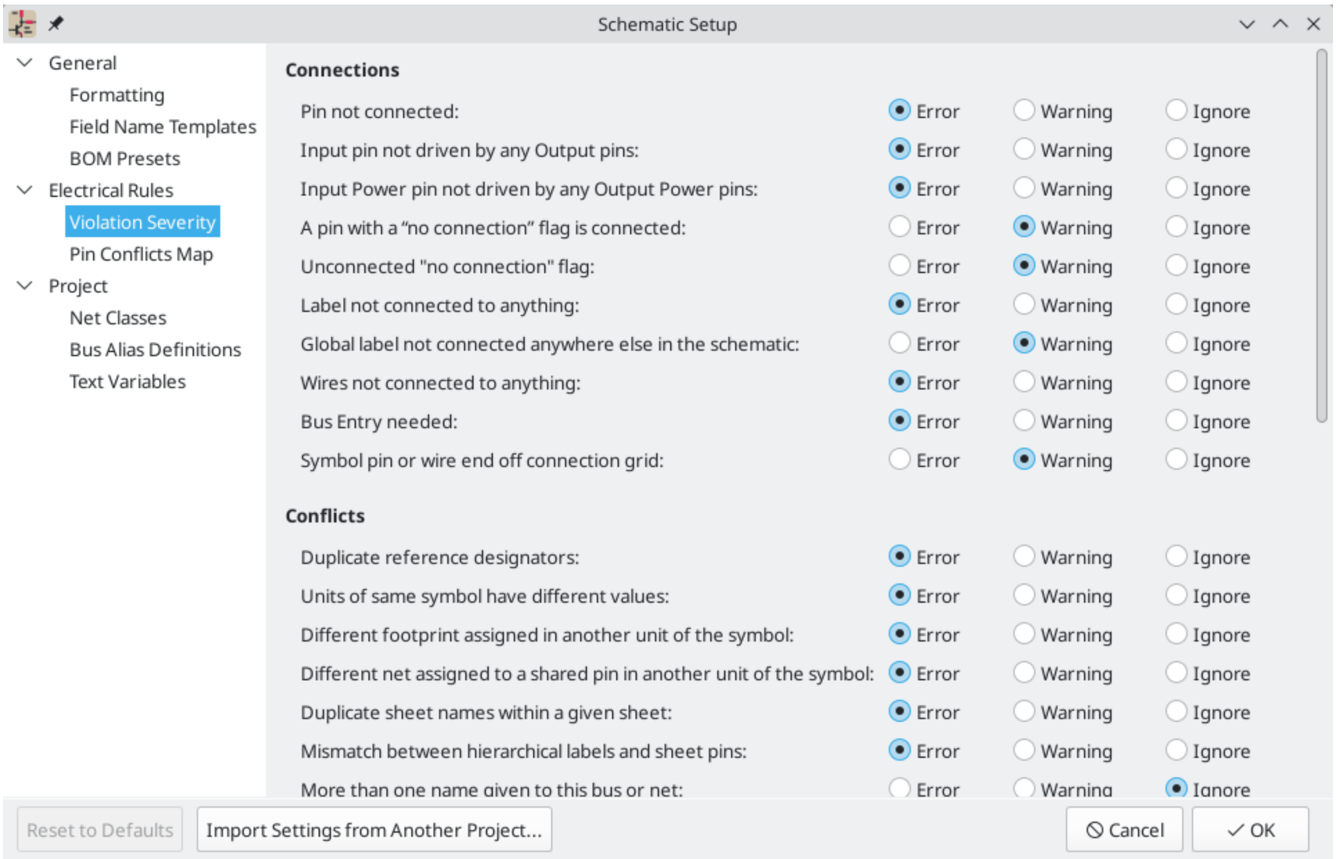


This would be the case also if the `VCC_IN` and `GND_IN` nets were connected to a symbol with Output Power pins. In fact, in this schematic, the `PWR_FLAG` symbols are on `VCC_IN` and `GND_IN` are not required, as there are no Input Power pins on those nets.

For more information about power pins and power flags, see the `PWR_FLAG` documentation.

ERC Configuration

The **Violation Severity** panel in Schematic Setup lets you configure what types of ERC messages should be reported as Errors, Warnings, or ignored.



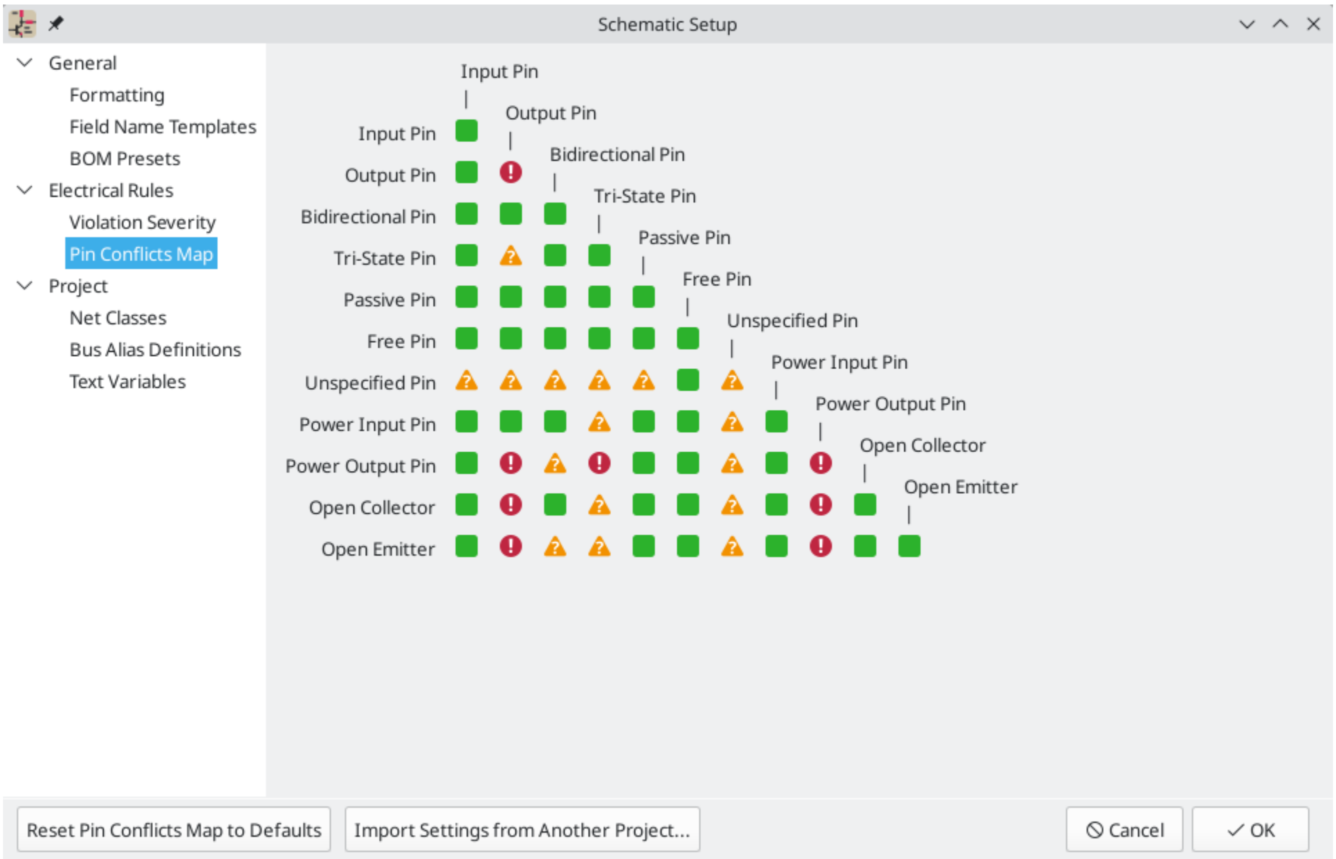
Connections

Pin not connected:	● Error	○ Warning	○ Ignore
Input pin not driven by any Output pins:	● Error	○ Warning	○ Ignore
Input Power pin not driven by any Output Power pins:	● Error	○ Warning	○ Ignore
A pin with a “no connection” flag is connected:	○ Error	● Warning	○ Ignore
Unconnected "no connection" flag:	○ Error	● Warning	○ Ignore
Label not connected to anything:	● Error	○ Warning	○ Ignore
Global label not connected anywhere else in the schematic:	○ Error	● Warning	○ Ignore
Wires not connected to anything:	● Error	○ Warning	○ Ignore
Bus Entry needed:	● Error	○ Warning	○ Ignore
Symbol pin or wire end off connection grid:	○ Error	● Warning	○ Ignore

Conflicts

Duplicate reference designators:	● Error	○ Warning	○ Ignore
Units of same symbol have different values:	● Error	○ Warning	○ Ignore
Different footprint assigned in another unit of the symbol:	● Error	○ Warning	○ Ignore
Different net assigned to a shared pin in another unit of the symbol:	● Error	○ Warning	○ Ignore
Duplicate sheet names within a given sheet:	● Error	○ Warning	○ Ignore
Mismatch between hierarchical labels and sheet pins:	● Error	○ Warning	○ Ignore
More than one name given to this bus or net:	○ Error	○ Warning	● Ignore

The **Pin Conflicts Map** panel in Schematic Setup allows you to configure connectivity rules to define electrical conditions for errors and warnings based on what types of pins are connected to each other. For example, by default an error is produced when an output pin is connected to another output pin.



	Input Pin	Output Pin	Bidirectional Pin	Tri-State Pin	Passive Pin	Free Pin	Unspecified Pin	Power Input Pin	Power Output Pin	Open Collector	Open Emitter
Input Pin	🟩										
Output Pin	🟩	❗									
Bidirectional Pin	🟩	🟩	🟩								
Tri-State Pin	🟩	⚠	🟩	🟩							
Passive Pin	🟩	🟩	🟩	🟩	🟩						
Free Pin	🟩	🟩	🟩	🟩	🟩	🟩					
Unspecified Pin	⚠	⚠	⚠	⚠	⚠	🟩	⚠				
Power Input Pin	🟩	🟩	🟩	⚠	🟩	🟩	⚠	🟩			
Power Output Pin	🟩	❗	⚠	❗	🟩	🟩	⚠	🟩	❗		
Open Collector	🟩	❗	🟩	⚠	🟩	🟩	⚠	🟩	❗	🟩	
Open Emitter	🟩	❗	⚠	⚠	🟩	🟩	⚠	🟩	❗	🟩	🟩

Rules can be changed by clicking on the desired square of the matrix, causing it to cycle through the choices: allowed, warning, error.

List of ERC checks

The table below lists the electrical rules that KiCad checks and the default violation severity for each check. All severities are configurable.

Connections ERC checks

These ERC checks look for issues with wire and label connections in the schematic.

Violation	Description	Default Severity
Pin not connected	This violation occurs when a symbol pin is not connected to a net, unless the pin has a no-connect flag or has electrical type Unconnected.	Error
Input pin not driven by any Output pins	This violation occurs when a symbol pin with electrical type Input is not connected to a driving pin. Driving pins are pins with the type output, bidirectional, tristate, power output, or passive pins.	Error
Input Power pin not driven by any Output Power pins	This violation occurs when a symbol pin with electrical type Input Power is not connected to an Output Power pin. A common cause of this violation is described above.	Error
A pin with a "no connection" flag is connected	The violation occurs when a symbol pin with a no connection flag is connected to a net.	Warning
Unconnected "no connection" flag	This violation occurs when a No connection flag is not connected to a pin or label.	Warning
Label not connected	This violation occurs when a global, hierarchical, local, or directive label is not connected to a pin or another label.	Error
Label connected to only one pin	This violation occurs when a label is connected to a net with fewer than two symbol pins on it. This indicates the label isn't being used to connect anything, as it it is only connected to a single symbol pin.	Warning
Global label only appears once in the schematic	This violation occurs when a global label only appears once in the schematic, meaning that the label is not forming any global connections. This violation is ignored by default to allow users to use global labels to label nets, even if the net does not connect anywhere else.	Ignore

Violation	Description	Default Severity
Local and global labels have the same name	This violation occurs when a local label has the same name as a global label. If these labels are on separate sheets, they will not connect, although they may have been intended to connect. Note that while a local label and a global label with the same name won't connect if they are on different sheets, they will connect if they are on the same sheet.	Warning
Wires not connected to anything	This violation occurs when a wire is not connected to any pin or label.	Error
Bus Entry needed	This violation only applies to projects imported from EAGLE projects. It indicates places where the importer was unable to automatically add bus entries to the imported schematic, so you must add them by hand.	Error
Symbol pin or wire end off connection grid	This violation occurs when a symbol pin or wire end is not aligned to the connection grid. Symbol pins and wire ends need to be aligned to the grid in order to connect to each other. The grid used for this check is defined by the **connection grid** setting in **Schematic Setup → Formatting → Connection grid**.	Warning
Four connection points are joined together	This violation occurs when wires join in a four-way (cross) junction. Such junctions are sometimes considered harmful because it can be unclear if all four wires are intended to be joined or if two wires were intended to cross without a junction.	Ignore
Duplicate pins with different nets	This violation occurs when two pins in the same symbol have the same pin number but are connected to different nets. Symbol pins do not need to be uniquely numbered within a symbol, but all pins with the same pin number need to be connected to the same net. This violation is always an error.	Error (not configurable)
Label connects more than one wire	This violation occurs when a label anchor connects to two wires (where two wires cross without connecting). In this situation is not possible to determine which net the label should connect to.	Warning
Unconnected wire endpoint	This violation occurs when a wire endpoint is not connected to anything.	Warning

Conflicts ERC checks

These ERC checks look for conflicting information in symbols, sheets, and buses.

Violation	Description	Default Severity
Different footprint assigned in another unit of the symbol	This violation occurs when units of a single symbol have different assigned footprints.	Error
Different net assigned to a shared pin in another unit of the symbol	This violation occurs when a pin that is shared between multiple units of a symbol is not connected to the same net in each unit.	Error
Duplicate sheet names within a given sheet	This violation occurs when two hierarchical sheets in the same parent sheet have the same name.	Error
Mismatch between hierarchical labels and sheet pins	This violation occurs when a hierarchical label does not have a corresponding hierarchical sheet pin in the parent sheet, or a hierarchical sheet pin does not have a corresponding hierarchical label in the child sheet.	Error
More than one name given given to this bus or net	This violation occurs when a net has multiple labels attached. Nets can only have a single name, so if multiple labels are attached to a net, one name will be selected and used as the canonical name.	Warning
Conflict between bus alias definitions across schematic sheets	This violation occurs when a bus alias has different members in different sheets. If the same bus alias name is used in multiple sheets, the members of the alias must be the same for each sheet.	Error
Buses are graphically connected but share no bus members	This violation occurs when buses that are graphically connected do not have bus members in common.	Error

Violation	Description	Default Severity
Invalid connection between bus and net items	This violation occurs when a bus is connected to a net item, such as a wire, a label referring to a single net, or a sheet pin referring to a single net. Labels and sheet pins can only be connected to buses if they refer to buses rather than individual signals.	Error
Net is graphically connected to a bus but not a bus member	This violation occurs when a net is connected to a bus with a bus entry but the net is not a member of that bus.	Warning
Ground pin not connected to ground net	This violation occurs when a power input or power output pin with a name containing "GND" is connected to a net with a name that doesn't contain "GND", but another power pin in the same symbol is connected to a "GND" net. This indicates that the GND pin is likely misconnected. The comparison is case-insensitive, so "GND", "gnd", and any other combination of uppercase and lowercase letters are all considered GND pins/nets.	Warning

Miscellaneous ERC checks

These ERC checks look for other miscellaneous issues in the schematic.

Violation	Description	Default Severity
Pin name resembles stacked pin	This violation occurs when a symbol's pin number resembles a stacked pin range but is malformed or invalid, and therefore the pin is not treated as a stacked pin.	Warning
Field name has leading or trailing whitespace	This violation occurs when a symbol or sheet field starts or ends wth whitespace. Leading or trailing whitespace isn't necessarily a problem, but it can be undesired and hard to notice.	Warning
Symbol is not annotated	This violation occurs when a symbol is not annotated with a unique reference designator.	Error
Unresolved text variable	This violation occurs when a text variable (`${variable_name}`) is used without being defined in Schematic Setup.	Error
Undefined netclass	This violation occurs when a netclass is assigned graphically (i.e. in the `Netclass` field of a label or directive label) but isn't defined in Schematic Setup.	Error
SPICE model issue	This violation occurs when a SPICE model has a syntax error or other problem.	Ignore

Violation	Description	Default Severity
Labels are similar (lower/upper case difference only)	This violation occurs when two labels are similar and differ only by the case of some letters. This may be a typo causing two labels to be disconnected when they are intended to be connected.	Warning
Power pins are similar (lower/upper case difference only)	This violation occurs when the net names driven by two global power pins are similar and differ only by the case of some letters. This may be a typo causing two global power pins to be disconnected when they are intended to be connected.	Warning
Power pin and label are similar (lower/upper case difference only)	This violation occurs when a label and the net name driven by a global power pin are similar and differ only by the case of some letters. This may be a typo causing two global power pins to be disconnected when they are intended to be connected.	Warning
Library symbol issue	This violation occurs when one of several symbol library issues is detected: • The symbol library for a symbol is not included and enabled in the library table • A symbol in the schematic does not exist in its symbol library	Warning
Symbol doesn't match copy in library	This violation occurs when a symbol in the schematic is different than the library version of the symbol. You can compare between the schematic and library versions of the symbol using the Compare Symbol with Library tool, which is available by right clicking the violation in the ERC window. If desired, you can update the schematic symbol to match the library symbol.	Warning

Violation	Description	Default Severity
Symbol has more units than are defined	This violation occurs when a symbol has more units placed in the schematic than are defined in the symbol. Units in the schematic must correspond exactly to the symbol definition.	Error
Symbol has units that are not placed	This violation occurs when a unit from a multi-unit symbol is not placed in the schematic. Unplaced units will not be connected to anything.	Warning
Symbol has input pins that are not placed	This violation occurs when a multi-unit symbol has units with input pins that are not placed, so those input pins will not be connected to anything.	Warning
Symbol has bidirectional pins that are not placed	This violation occurs when a multi-unit symbol has units with bidirectional pins that are not placed, so those input pins will not be connected to anything.	Warning
Symbol has power input pins that are not placed	This violation occurs when a multi-unit symbol has units with power input pins that are not placed, so those input pins will not be connected to anything.	Error
Conflict problem between pins	This violation occurs when a connection between pins is not allowed per the allowed connections in the Pin Conflicts Map.	From Pin Conflicts Map

User-definable ERC violations

You can manually trigger schematic ERC warnings or errors using special text variables. These items will appear as errors or warnings when ERC runs. This can be useful to flag items for later followup or review.

To cause an ERC violation, use the text variable `${ERC_ERROR <violation name>}` or `${ERC_WARNING <violation name>}` depending on whether an error or warning is desired. You can place this in a text item, text box, or field, including symbol fields, sheet fields, and label fields. When ERC runs, this will generate a ERC violation with the given violation name. These text variables resolve to an empty string in the schematic, and any text after the braces is included in the ERC 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 `${ERC_ERROR TODO}Calculate resistor value` will appear in the board as just the text "Calculate resistor value", and will generate an ERC error named "TODO" with "Calculate resistor value" in the description.

Berkas Laporan ERC

An ERC report file can be generated and saved by clicking the **Save...** button in the ERC dialog. The file extension for ERC report files is `.rpt`.

NOTE ERC reports can also be generated by the `kicad-cli` tool in either text (`.rpt`) format or JSON.

Assigning Footprints

Before routing a PCB, footprints need to be selected for every component that will be assembled on the board. Footprints define the copper connections between physical components and the routed traces on a circuit board.

Some symbols come with footprints pre-assigned, but for many symbols there are multiple possible footprints, so the user needs to select the appropriate one.

KiCad offers several ways to assign footprints:

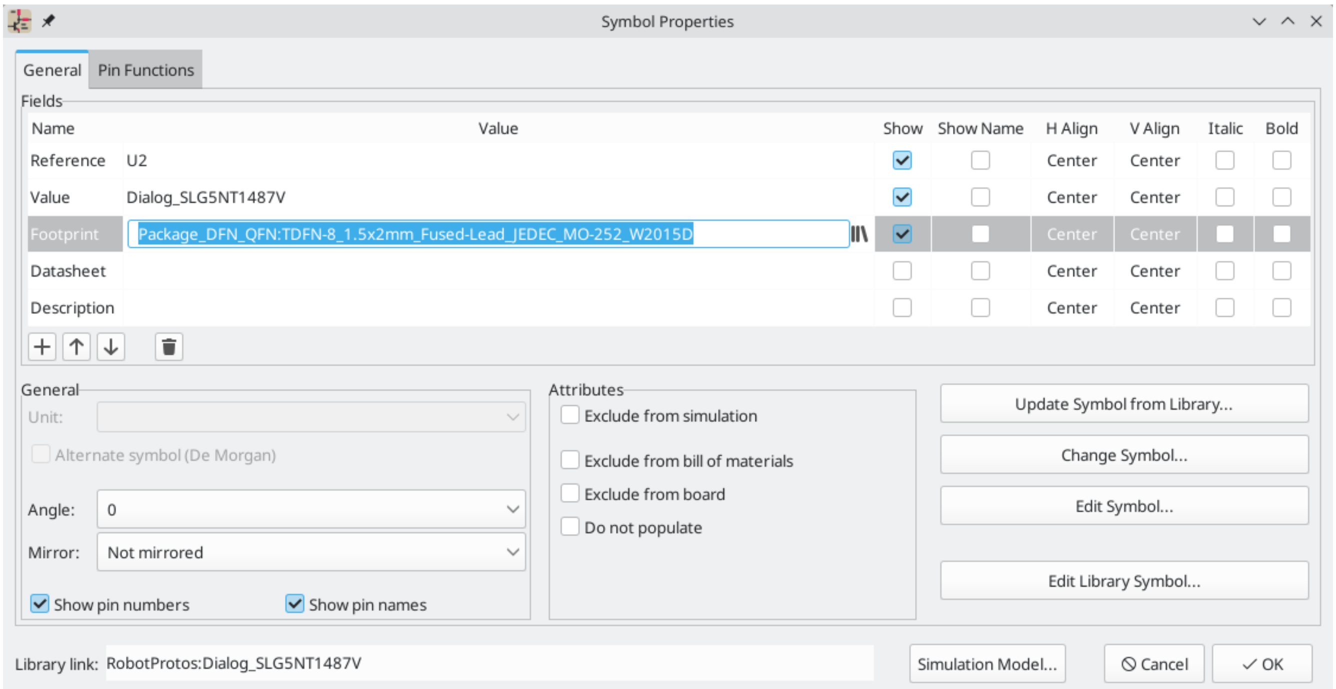
- Properti Simbol
  - Symbol Properties Dialog
  - Symbol Fields Table
- While placing symbols
- Footprint Assignment Tool

Each method will be explained below. Which to use is a matter of preference; one method may be more convenient depending on the situation. All of these methods are equivalent in that they store the name of the selected footprint in the symbol's `Footprint` field.

NOTE The Footprint Library Table needs to be configured before footprints can be assigned. For information on configuring the Footprint Library Table, please see the PCB Editor manual.

Assigning Footprints in Symbol Properties

A symbol's `Footprint` field can be edited directly in the symbol's Properties window.



Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Reference	U2	☑	□	Center	Center	□	□
Value	Dialog_SLG5NT1487V	☑	□	Center	Center	□	□
Footprint	Package_DFN_QFN:TDFN-8_1.5x2mm_Fused-Lead_JEDEC_MO-252_W2015D	☑	□	Center	Center	□	□
Datasheet		□	□	Center	Center	□	□
Description		□	□	Center	Center	□	□

General

Unit:

□ Alternate symbol (De Morgan)

Angle: 0

Mirror: Not mirrored

☑ Show pin numbers ☑ Show pin names

Attributes

□ Exclude from simulation

□ Exclude from bill of materials

□ Exclude from board

□ Do not populate

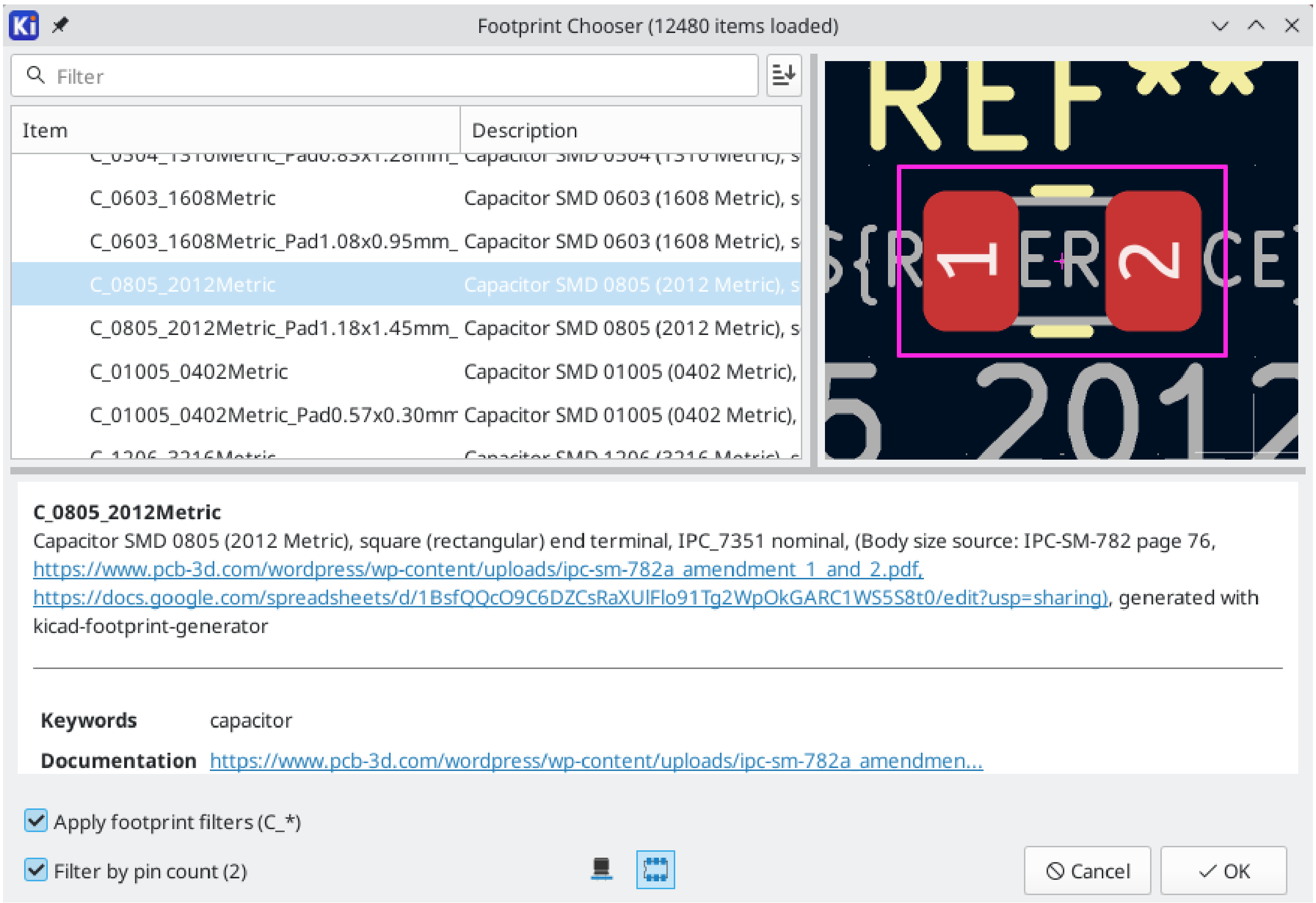
Library link: RobotProtos:Dialog_SLG5NT1487V

Clicking the |||\ button in the `Footprint` field opens the Footprint Chooser, which shows the available footprints sorted by footprint libraries.

The Footprint Chooser filters footprints by name, description, and keywords, as well as any fields that are shown as columns, according to what you type into the search field. `*` and `?` wildcards are available. The footprint search behaves the same as in the symbol chooser dialog.

If the symbol defines any footprint filters, the **apply footprint filters** option can be used to hide footprints that don't match those filters. If the **filter by pin count** option is selected, only footprints that match the symbol's pincount will be listed. You can choose to sort search results alphabetically or by best match by clicking on the ⇣ button.

Single clicking a footprint name selects the footprint and displays it in the preview pane on the right. You can switch between a 2D and 3D preview of the footprint by clicking the ▦ and ▙ buttons. Double clicking on a footprint closes the chooser and sets the symbol's `Footprint` field to the selected footprint.



Item	Description
C_0504_1310Metric_Pad0.83x1.28mm_	Capacitor SMD 0504 (1310 Metric), s
C_0603_1608Metric	Capacitor SMD 0603 (1608 Metric), s
C_0603_1608Metric_Pad1.08x0.95mm_	Capacitor SMD 0603 (1608 Metric), s
C_0805_2012Metric	Capacitor SMD 0805 (2012 Metric), s
C_0805_2012Metric_Pad1.18x1.45mm_	Capacitor SMD 0805 (2012 Metric), s
C_01005_0402Metric	Capacitor SMD 01005 (0402 Metric),
C_01005_0402Metric_Pad0.57x0.30mr	Capacitor SMD 01005 (0402 Metric),
C_1206_3216Metric	Capacitor SMD 1206 (3216 Metric), s

C_0805_2012Metric
Capacitor SMD 0805 (2012 Metric), square (rectangular) end terminal, IPC_7351 nominal, (Body size source: IPC-SM-782 page 76, https://www.pcb-3d.com/wordpress/wp-content/uploads/ipc-sm-782a_amendment_1_and_2.pdf, https://docs.google.com/spreadsheets/d/1BsfQQcO9C6DZCsRaXUlFlo91Tg2WpOkGARC1WS5S8t0/edit?usp=sharing), generated with kicad-footprint-generator

Keywords	capacitor
Documentation	https://www.pcb-3d.com/wordpress/wp-content/uploads/ipc-sm-782a_amendmen...

☑ Apply footprint filters (C_*)

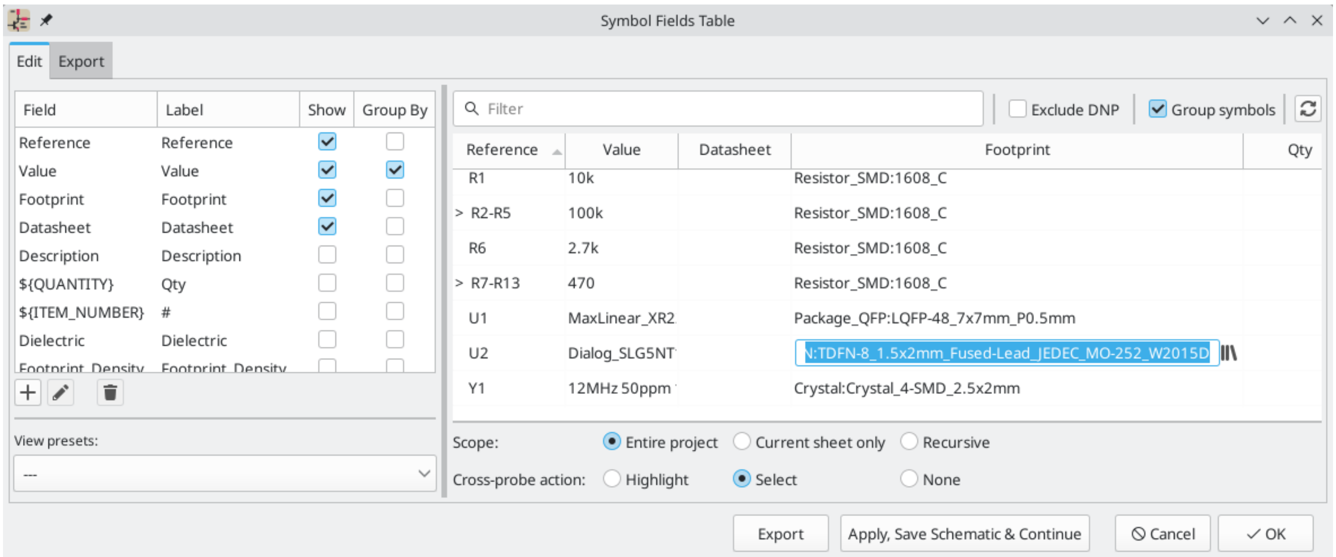
☑ Filter by pin count (2)

Assigning Footprints with the Symbol Fields Table

Rather than editing the properties of each symbol individually, the Symbol Fields Table can be used to view and edit the properties of all symbols in the design in one place. This includes assigning footprints by editing the `Footprint` field of each symbol.

The Symbol Fields Table is accessed with **Tools** → **Edit Symbol Fields...**, or with the ▦ button on the top toolbar.

The `Footprint` field behaves the same here as in the Symbol Properties window: it can be edited directly, or footprints can be selected visually with the Footprint Library Browser.



Field	Label	Show	Group By
Reference	Reference	☑	□
Value	Value	☑	☑
Footprint	Footprint	☑	□
Datasheet	Datasheet	☑	□
Description	Description	□	□
${QUANTITY}	Qty	□	□
${ITEM_NUMBER}	#	□	□
Dielectric	Dielectric	□	□
Footprint_Density	Footprint_Density	□	□

View presets: ---

Filter □ Exclude DNP ☑ Group symbols

Reference	Value	Datasheet	Footprint	Qty
R1	10k		Resistor_SMD:1608_C	
> R2-R5	100k		Resistor_SMD:1608_C	
R6	2.7k		Resistor_SMD:1608_C	
> R7-R13	470		Resistor_SMD:1608_C	
U1	MaxLinear_XR2		Package_QFP:LQFP-48_7x7mm_P0.5mm	
U2	Dialog_SLG5NT		N:TDFN-8_1.5x2mm_Fused-Lead_JEDEC_MO-252_W2015D	
Y1	12MHz 50ppm		Crystal:Crystal_4-SMD_2.5x2mm	

Scope: ● Entire project ○ Current sheet only ○ Recursive

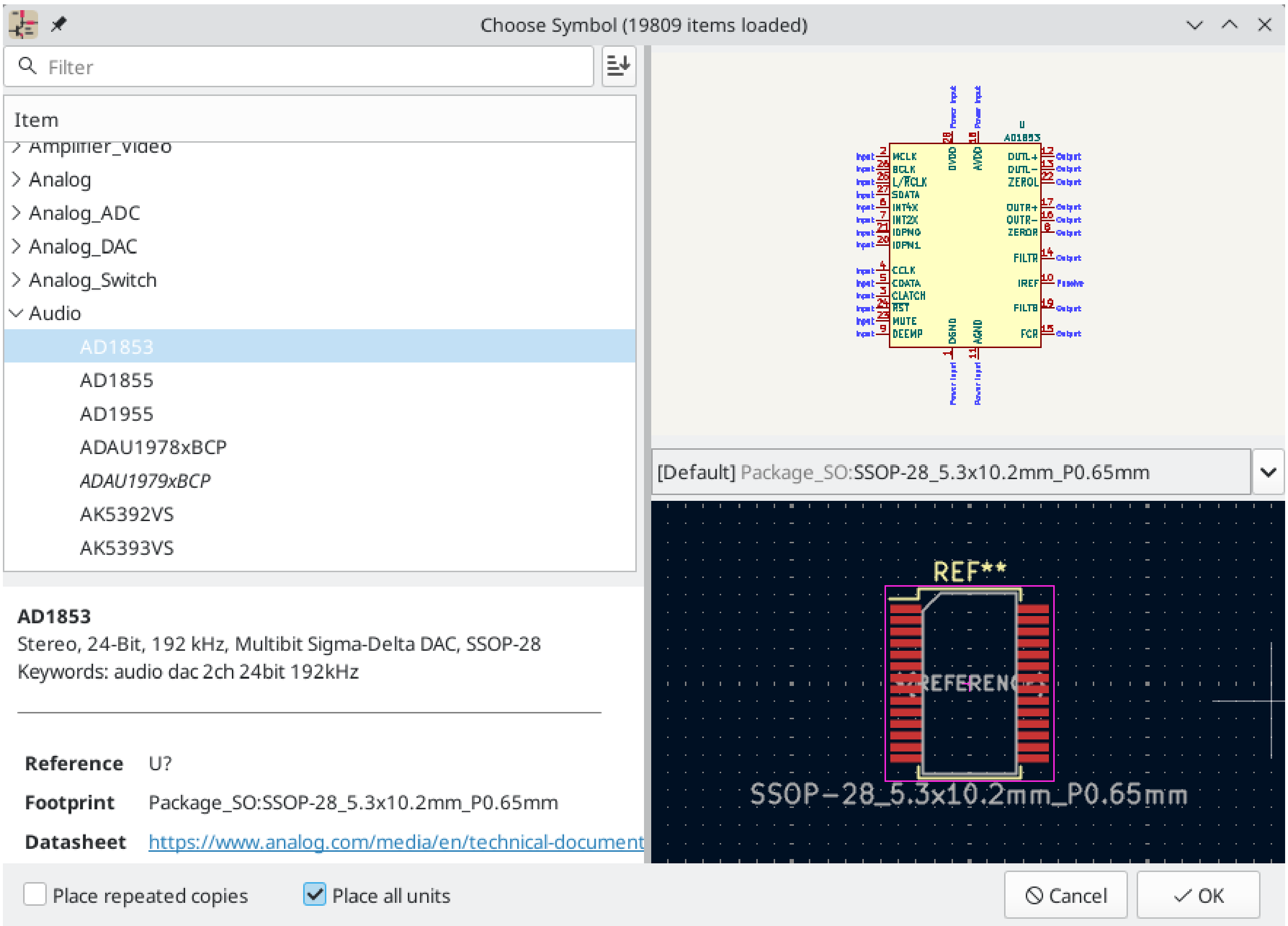
Cross-probe action: ○ Highlight ● Select ○ None

For more information on the Symbol Fields Table, see the section on editing symbol properties.

Assigning Footprints While Placing Symbols

Footprints can be assigned to symbols when the symbol is first added to the schematic.

Some symbols are defined with a default footprint. These symbols will have this footprint preassigned when they are added to the schematic. If a symbol has a default footprint, the footprint will be graphically previewed in the symbol chooser dialog when the symbol is selected. For symbols without a default symbol defined, the footprint dropdown will say "No default footprint", and the footprint preview canvas will say "No footprint specified".



[Default] Package_SO:SSOP-28_5.3x10.2mm_P0.65mm

AD1853
Stereo, 24-Bit, 192 kHz, Multibit Sigma-Delta DAC, SSOP-28
Keywords: audio dac 2ch 24bit 192kHz

Reference	U?
Footprint	Package_SO:SSOP-28_5.3x10.2mm_P0.65mm
Datasheet	https://www.analog.com/media/en/technical-document

□ Place repeated copies ☑ Place all units

Symbols can have footprint filters that specify which footprints are appropriate to use with that symbol. If footprint filters are defined for the selected symbol, all footprints that match the footprint filters will appear as options in the footprint dropdown. The selected footprint will be displayed in the preview canvas and will be assigned to the symbol when the symbol is added to the schematic.

> **NOTE** Footprint options will not appear in the footprint dropdown unless the footprint libraries are loaded. Footprint libraries are loaded the first time the Footprint Editor or Footprint Library Browser are opened in a session.

For more information on footprint filters, see the Symbol Editor Documentation.

Assigning Footprints with the Footprint Assignment Tool

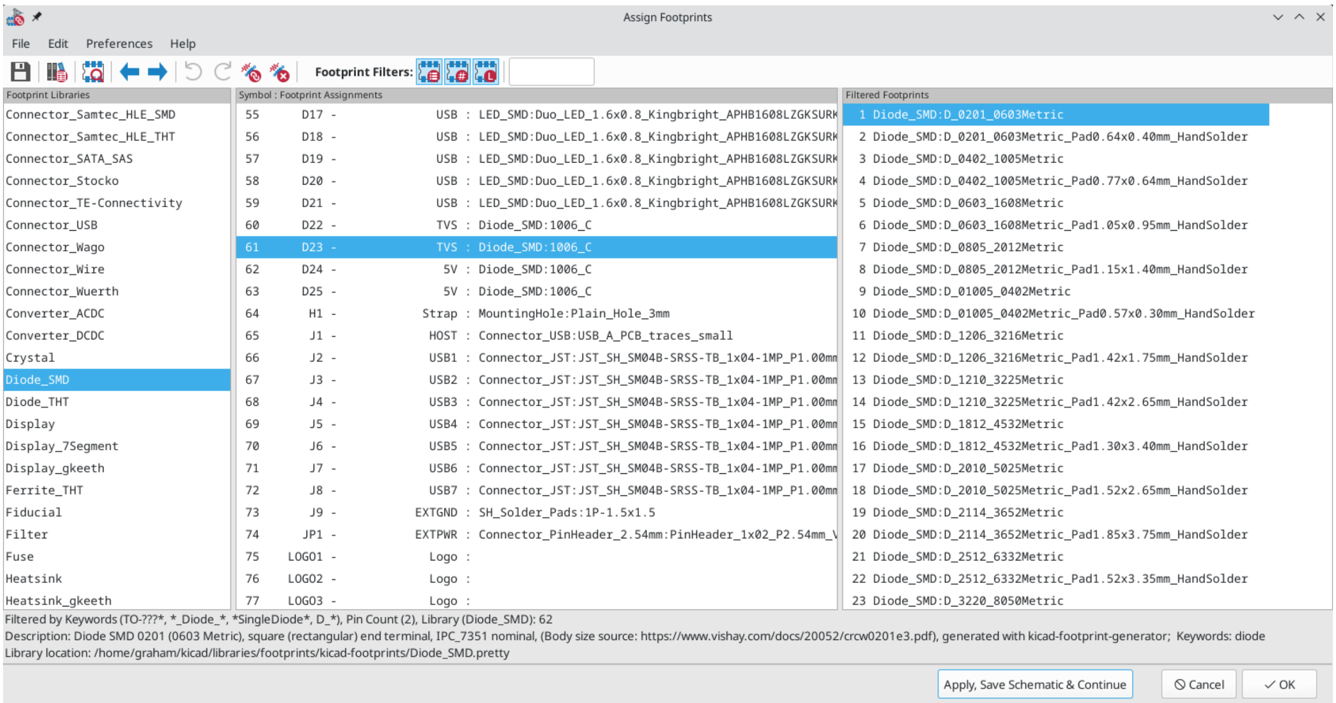
The Footprint Assignment Tool allows you to associate symbols in your schematic to footprints used when laying out the printed circuit board. It provides footprint list filtering, footprint viewing, and 3D component model viewing to help ensure the correct footprint is associated with each component.

Components can be assigned to their corresponding footprints manually or automatically by creating equivalence files (.equ files). Equivalence files are lookup tables associating each component with its footprint.

Run the tool with **Tools** → **Assign Footprints...**, or by clicking the icon in the top toolbar.

Footprint Assignment Tool Overview

The image below shows the main window of the Footprint Assignment Tool.



Footprint Libraries	Symbol : Footprint Assignments	Filtered Footprints
Connector_Samtec_HLE_SMD	55 D17 - USB : LED_SMD:Duo_LED_1.6x0.8_Kingbright_APHB1608LZGKSURK	1 Diode_SMD:D_0201_0603Metric
Connector_Samtec_HLE_THT	56 D18 - USB : LED_SMD:Duo_LED_1.6x0.8_Kingbright_APHB1608LZGKSURK	2 Diode_SMD:D_0201_0603Metric_Pad0.64x0.40mm_HandSolder
Connector_SATA_SAS	57 D19 - USB : LED_SMD:Duo_LED_1.6x0.8_Kingbright_APHB1608LZGKSURK	3 Diode_SMD:D_0402_1005Metric
Connector_Stocko	58 D20 - USB : LED_SMD:Duo_LED_1.6x0.8_Kingbright_APHB1608LZGKSURK	4 Diode_SMD:D_0402_1005Metric_Pad0.77x0.64mm_HandSolder
Connector_TE-Connectivity	59 D21 - USB : LED_SMD:Duo_LED_1.6x0.8_Kingbright_APHB1608LZGKSURK	5 Diode_SMD:D_0603_1608Metric
Connector_USB	60 D22 - TVS : Diode_SMD:1006_C	6 Diode_SMD:D_0603_1608Metric_Pad1.05x0.95mm_HandSolder
Connector_Wago	61 D23 - TVS : Diode_SMD:1006_C	7 Diode_SMD:D_0805_2012Metric
Connector_Wire	62 D24 - 5V : Diode_SMD:1006_C	8 Diode_SMD:D_0805_2012Metric_Pad1.15x1.40mm_HandSolder
Connector_Wuerth	63 D25 - 5V : Diode_SMD:1006_C	9 Diode_SMD:D_01005_0402Metric
Converter_ACDC	64 H1 - Strap : MountingHole:Plain_Hole_3mm	10 Diode_SMD:D_01005_0402Metric_Pad0.57x0.30mm_HandSolder
Converter_DCDC	65 J1 - HOST : Connector_USB:USB_A_PCB_traces_small	11 Diode_SMD:D_1206_3216Metric
Crystal	66 J2 - USB1 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	12 Diode_SMD:D_1206_3216Metric_Pad1.42x1.75mm_HandSolder
Diode_SMD	67 J3 - USB2 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	13 Diode_SMD:D_1210_3225Metric
Diode_THT	68 J4 - USB3 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	14 Diode_SMD:D_1210_3225Metric_Pad1.42x2.65mm_HandSolder
Display	69 J5 - USB4 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	15 Diode_SMD:D_1812_4532Metric
Display_7Segment	70 J6 - USB5 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	16 Diode_SMD:D_1812_4532Metric_Pad1.30x3.40mm_HandSolder
Display_gkeeth	71 J7 - USB6 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	17 Diode_SMD:D_2010_5025Metric
Ferrite_THT	72 J8 - USB7 : Connector_JST:JST_SH_SM04B-SRSS-TB_1x04-1MP_P1.00mm	18 Diode_SMD:D_2010_5025Metric_Pad1.52x2.65mm_HandSolder
Fiducial	73 J9 - EXTGND : SH_Solder_Pads:1P-1.5x1.5	19 Diode_SMD:D_2114_3652Metric
Filter	74 JP1 - EXTPWR : Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_V	20 Diode_SMD:D_2114_3652Metric_Pad1.85x3.75mm_HandSolder
Fuse	75 LOGO1 - Logo :	21 Diode_SMD:D_2512_6332Metric
Heatsink	76 LOGO2 - Logo :	22 Diode_SMD:D_2512_6332Metric_Pad1.52x3.35mm_HandSolder
Heatsink_gkeeth	77 LOGO3 - Logo :	23 Diode_SMD:D_3220_8050Metric

Filtered by Keywords (TO-???*, *_Diode_*, *SingleDiode*, D_*), Pin Count (2), Library (Diode_SMD): 62
Description: Diode SMD 0201 (0603 Metric), square (rectangular) end terminal, IPC_7351 nominal, (Body size source: https://www.vishay.com/docs/20052/crcw0201e3.pdf), generated with kicad-footprint-generator; Keywords: diode
Library location: /home/graham/kicad/libraries/footprints/kicad-footprints/Diode_SMD.pretty

- The left pane contains the list of available footprint libraries associated with the project.
- The center pane contains the list of symbols in the schematic.
- The right pane contains the list of available footprints loaded from the project footprint libraries.
-

The bottom pane describes the filters that have been applied to the footprint list and prints information about the footprint selected in the rightmost pane.

The top toolbar contains the following commands:

💾	Transfer the current footprint associations to the schematic.
📚	Edit the global and project footprint library tables.
🔍	View the selected footprint in the footprint viewer.
⬅	Select the previous symbol without a footprint association.
➡	Select the next symbol without a footprint association.
↺	Undo last edit.
↻	Redo last edit.
↻	Perform automatic footprint association using an equivalence file.
✕	Delete all footprint assignments.
☰	Filter footprint list by footprint filters defined in the selected symbol.
#	Filter footprint list by pin count of the selected symbol.
L	Filter footprint list by selected library.

The following table lists the keyboard commands for the Footprint Assignment Tool:

Right Arrow / Tab	Activate the pane to the right of the currently activated pane. Wrap around to the first pane if the last pane is currently activated.
Left Arrow	Activate the pane to the left of the currently activated pane. Wrap around to the last pane if the first pane is currently activated.
Up Arrow	Select the previous item of the currently selected list.
Down Arrow	Select the next item of the currently selected list.
Page Up	Select the item one full page upwards of the currently selected item.
Page Down	Select the item one full page downwards of the currently selected item.
Home	Select the first item of the currently selected list.
End	Select the last item of the currently selected list.

Manually Assigning Footprints with the Footprint Assignment Tool

To manually associate a footprint with a component, first select a component in the component (middle) pane. Then select a footprint in the footprint (right) pane by double-clicking on the name of the desired footprint. The footprint will be assigned to the selected component, and the next component without an assigned footprint is automatically selected.

> **NOTE** If no footprints appear in the footprint pane, check that the footprint filter options are correctly applied.

When all components have footprints assigned to them, click the **OK** button to save the assignments and exit the tool. Alternatively, click **Cancel** to discard the updated assignments, or **Apply, Save Schematic & Continue** to save the new assignments without exiting the tool.

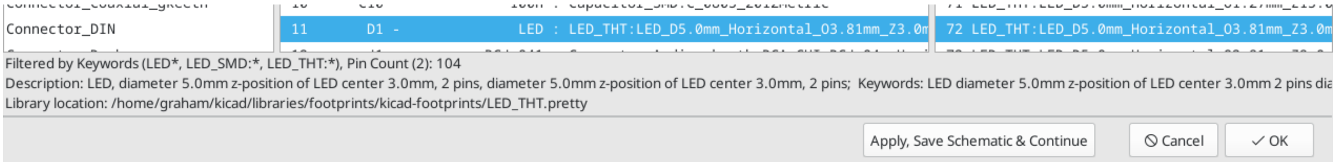
Filtering the Footprint List

There are four filtering options which restrict which footprints are displayed in the footprint pane. The filtering options are enabled and disabled with three buttons and a textbox in the top toolbar.

- : Activate filters that can be defined in each symbol. For example, an opamp symbol might define filters that show only SOIC and DIP footprints.
- : Only show footprints that match the selected symbol's pin count.
- : Only show footprints from the library selected in the left pane.
- Entering text in the textbox hides footprints that do not match the text. This filter is disabled when the box is empty.

When all filters are disabled, the full footprint list is shown.

The applied filters are described in the bottom pane of the window, along with the number of footprints that meet the selected filters. For example, when the symbol's footprint filters and pin count filters are enabled, the bottom pane prints the footprint filters and pin count:



```
Connector_DIN      11    D1 -    LED : LED_THT:LED_D5.0mm_Horizontal_O3.81mm_Z3.0m      72 LED_THT:LED_D5.0mm_Horizontal_O3.81mm_Z3.0m
```

Filtered by Keywords (LED*, LED_SMD:*, LED_THT:*), Pin Count (2): 104
Description: LED, diameter 5.0mm z-position of LED center 3.0mm, 2 pins, diameter 5.0mm z-position of LED center 3.0mm, 2 pins; Keywords: LED diameter 5.0mm z-position of LED center 3.0mm 2 pins dia
Library location: /home/graham/kicad/libraries/footprints/kicad-footprints/LED_THT.pretty

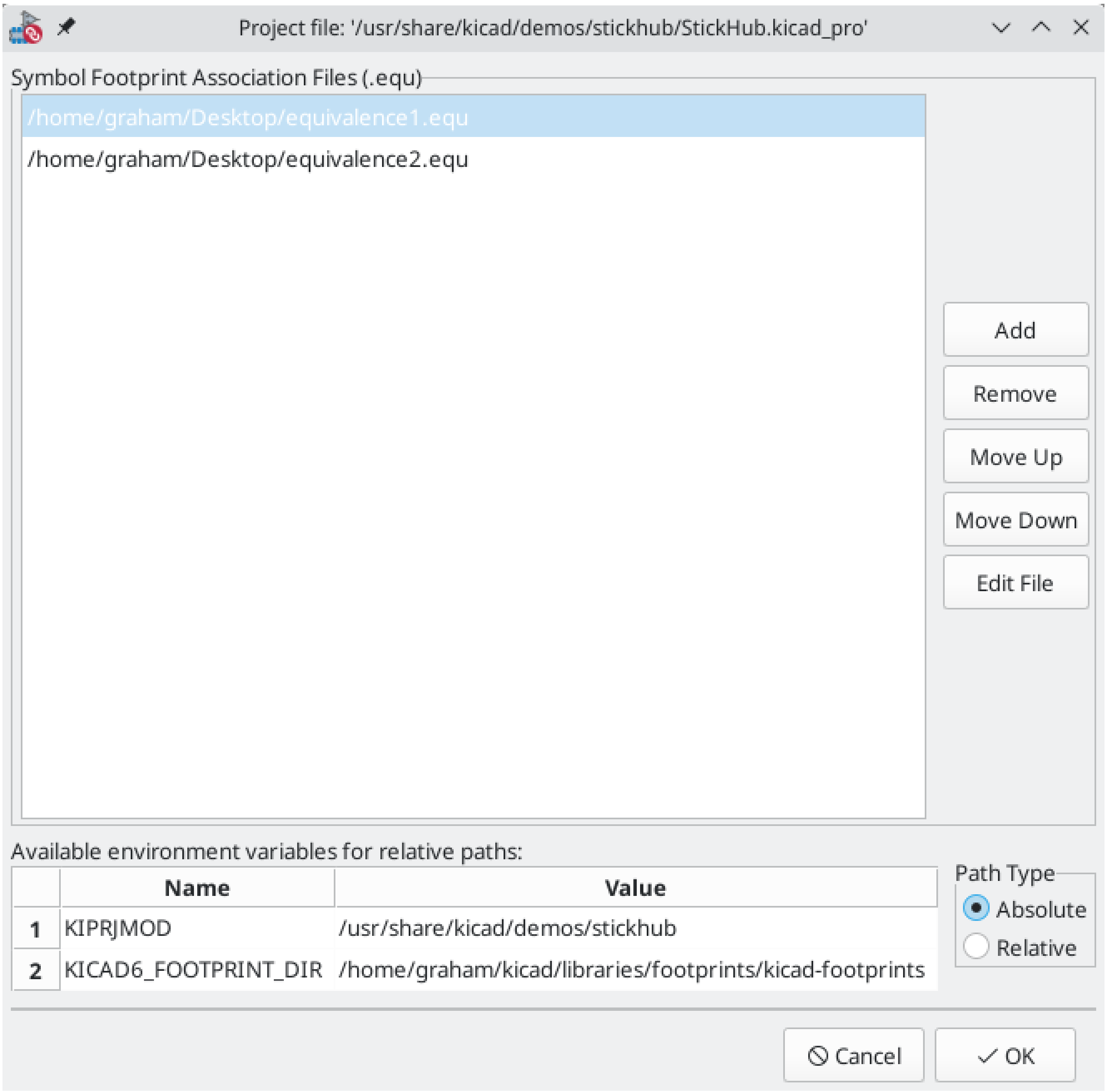
Multiple filters can be used at once to help narrow down the list of possibly appropriate footprints in the footprint pane. The symbols in KiCad's standard library define footprint filters that are designed to be used in combination with the pin count filter.

Automatically Assigning Footprints with the Footprint Assignment Tool

The Footprint Assignment Tool allows you to store footprint assignments in an external file and load the assignments later, even in a different project. This allows you to automatically associate symbols with the appropriate footprints.

The external file is referred to as an equivalence file, and it stores a mapping of a symbol value to a corresponding footprint. Equivalence files typically use the `.equ` file extension. Equivalence files are plain text files with a simple syntax, and must be created by the user using a text editor. The syntax is described below.

You can select which equivalence files to use by clicking **Preferences** → **Manage Footprint Association Files** in the Footprint Assignment Tool.



Project file: '/usr/share/kicad/demos/stickhub/StickHub.kicad_pro'

Symbol Footprint Association Files (.equ)

/home/graham/Desktop/equivalence1.equ

/home/graham/Desktop/equivalence2.equ

Available environment variables for relative paths:

	Name	Value
1	KIPRJMOD	/usr/share/kicad/demos/stickhub
2	KICAD6_FOOTPRINT_DIR	/home/graham/kicad/libraries/footprints/kicad-footprints

Path Type

● Absolute

○ Relative

- Add new equivalence files by clicking the **Add** button.
- Remove the selected equivalence file by clicking the **Remove** button.
- Change the priority of equivalence files by clicking the **Move Up** and **Move Down** buttons. If a symbol's value is found in multiple equivalence files, the footprint from the last matching equivalence file will override earlier equivalence files.
- Open the selected equivalence file by clicking the **Edit File** button.

Relevant environment variables are shown at the bottom of the window. When the **Relative** path option is checked, these environment variables will automatically be used to make paths to selected equivalence files relative to the project or footprint libraries.

Once the desired equivalence files have been loaded in the correct order, automatic footprint association can be performed by clicking the button in the top toolbar of the Footprint Assignment Tool.

All symbols with a value found in a loaded equivalence file will have their footprints automatically assigned. However, symbols that already have footprints assigned will not be updated.

Equivalence File Format

Equivalence files consist of one line for each symbol value. Each line has the following structure:

`'<symbol value>' '<footprint library>:<footprint name>'`

Each name/value must be surrounded by single quotes (`'`) and separated by one or more spaces. Lines starting with `#` are comments.

For example, if you want all symbols with the value `LM4562` to be assigned the footprint `Package_SO:SOIC-8_3.9x4.9_P1.27mm`, the line in the equivalence file should be:

`'LM4562' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'`

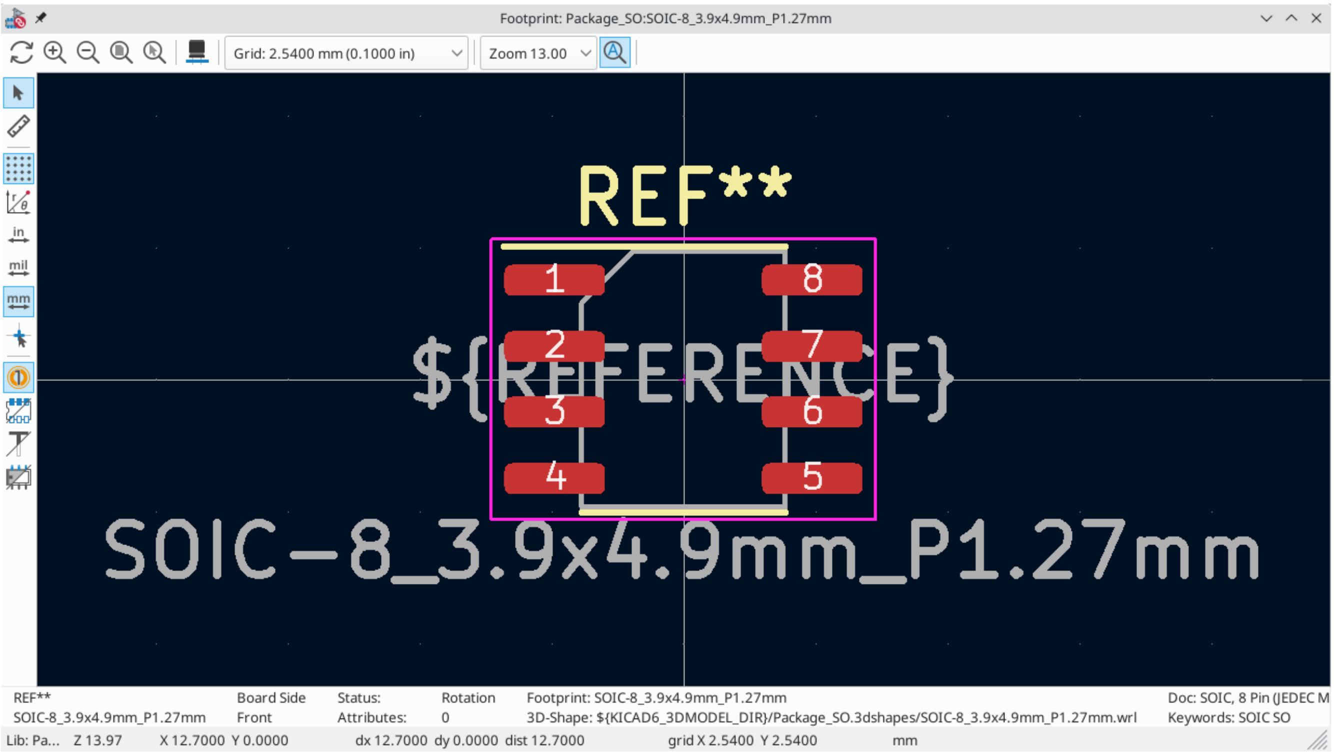
Here is an example equivalence file:

```
#integrated circuits (smd):
'74LV14' 'Package_SO:SOIC-14_3.9x8.7mm_P1.27mm'
'EL7242C' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'DS1302N' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LM324N' 'Package_SO:SOIC-14_3.9x8.7mm_P1.27mm'
'LM358' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LTC1878' 'Package_SO:MSOP-8_3x3mm_P0.65mm'
'24LC512I/SM' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LM2903M' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LT1129_SO8' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LT1129CS8-3.3' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LT1129CS8' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'LM358M' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'TL7702BID' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'TL7702BCD' 'Package_SO:SOIC-8_3.9x4.9_P1.27mm'
'U2270B' 'Package_SO:SOIC-16_3.9x9.9_P1.27mm'

#regulators
'LP2985LV' 'Package_TO_SOT_SMD:SOT-23-5_HandSoldering'
```

Viewing the Current Footprint

The Footprint Assignment Tool contains a footprint viewer. Clicking the button in the top toolbar launches the footprint viewer and shows the selected footprint.



Footprint: Package_SO:SOIC-8_3.9x4.9mm_P1.27mm
Grid: 2.5400 mm (0.1000 in)
Zoom 13.00
in
mil
mm
REF**
${REFERENCE}
1
2
3
4
5
6
7
8
SOIC-8_3.9x4.9mm_P1.27mm

REF**	Board Side	Status:	Rotation	Footprint: SOIC-8_3.9x4.9mm_P1.27mm	Doc: SOIC, 8 Pin (JEDEC M
SOIC-8_3.9x4.9mm_P1.27mm	Front	Attributes:	0	3D-Shape: ${KICAD6_3DMODEL_DIR}/Package_SO.3dshapes/SOIC-8_3.9x4.9mm_P1.27mm.wrl	Keywords: SOIC SO

Lib: Pa... Z 13.97 X 12.7000 Y 0.0000 dx 12.7000 dy 0.0000 dist 12.7000 grid X 2.5400 Y 2.5400 mm

The top toolbar contains the following commands:

⟳	Refresh view
⊕	Zoom in
⊖	Zoom out
🔍	Zoom to fit drawing in display area
▬	Show 3D viewer

The left toolbar contains the following commands:

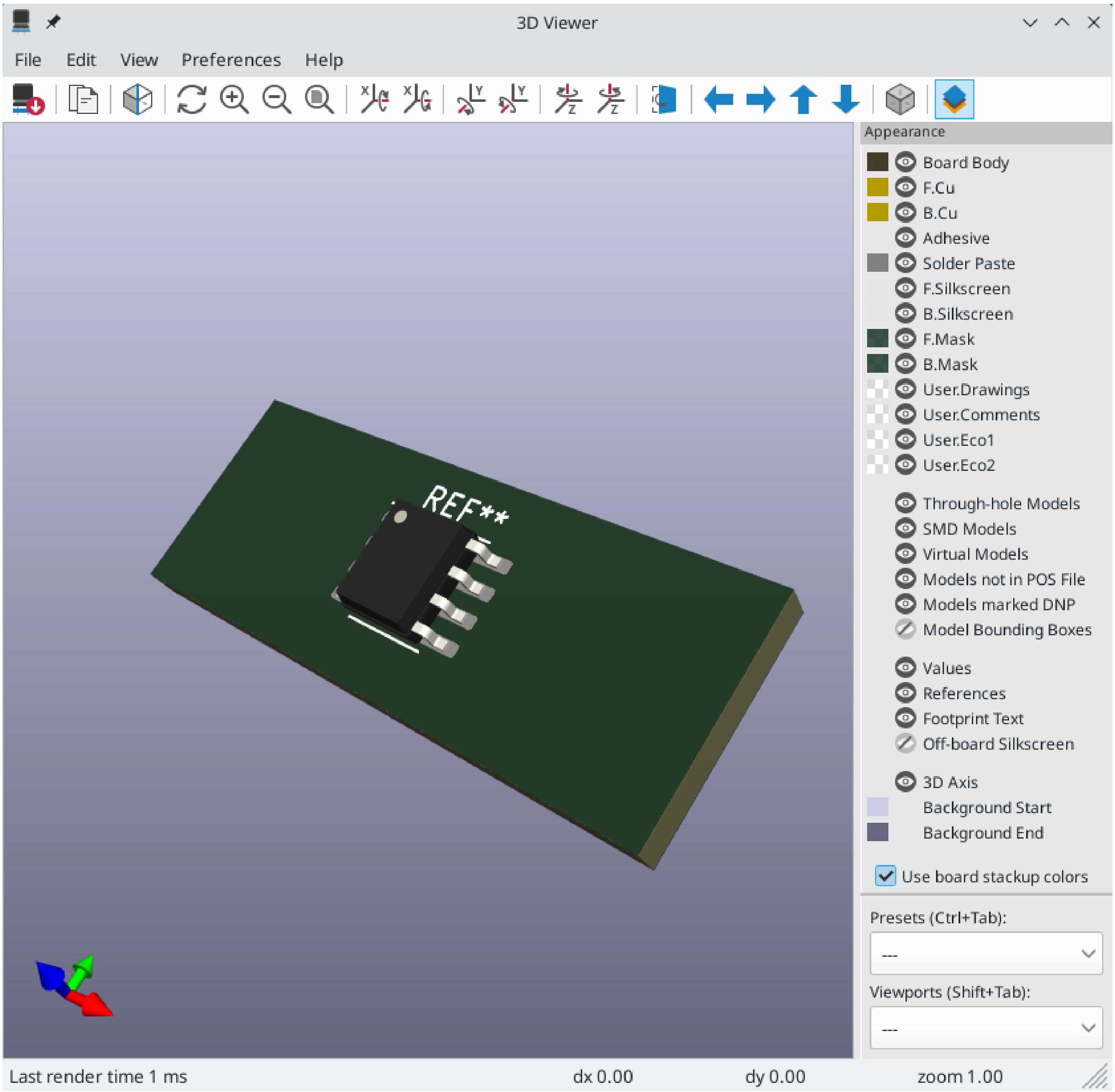
↖	Use the select tool
📏	Interactively measure between two points
⠿	Display grid dots or lines
r θ	Switch between polar and cartesian coordinate systems
in	Use inches
mil	Display coordinates in mils (1/1000 of an inch)
mm	Display coordinates in millimeters
✛	Toggle display of full-window crosshairs
①	Toggle between drawing pads in sketch or normal mode
	Toggle between drawing pads in normal mode or outline mode
T	Toggle between drawing text in normal mode or outline mode
	Toggle between drawing graphic lines in normal mode or outline mode

Viewing the Current 3D Model

Clicking the button opens the footprint in the 3D model viewer.

NOTE

If a 3D model does not exist for the current footprint, only the footprint itself will be shown in the 3D Viewer.



☑ Use board stackup colors

Presets (Ctrl+Tab): ---

Viewports (Shift+Tab): ---

Last render time 1 ms dx 0.00 dy 0.00 zoom 1.00

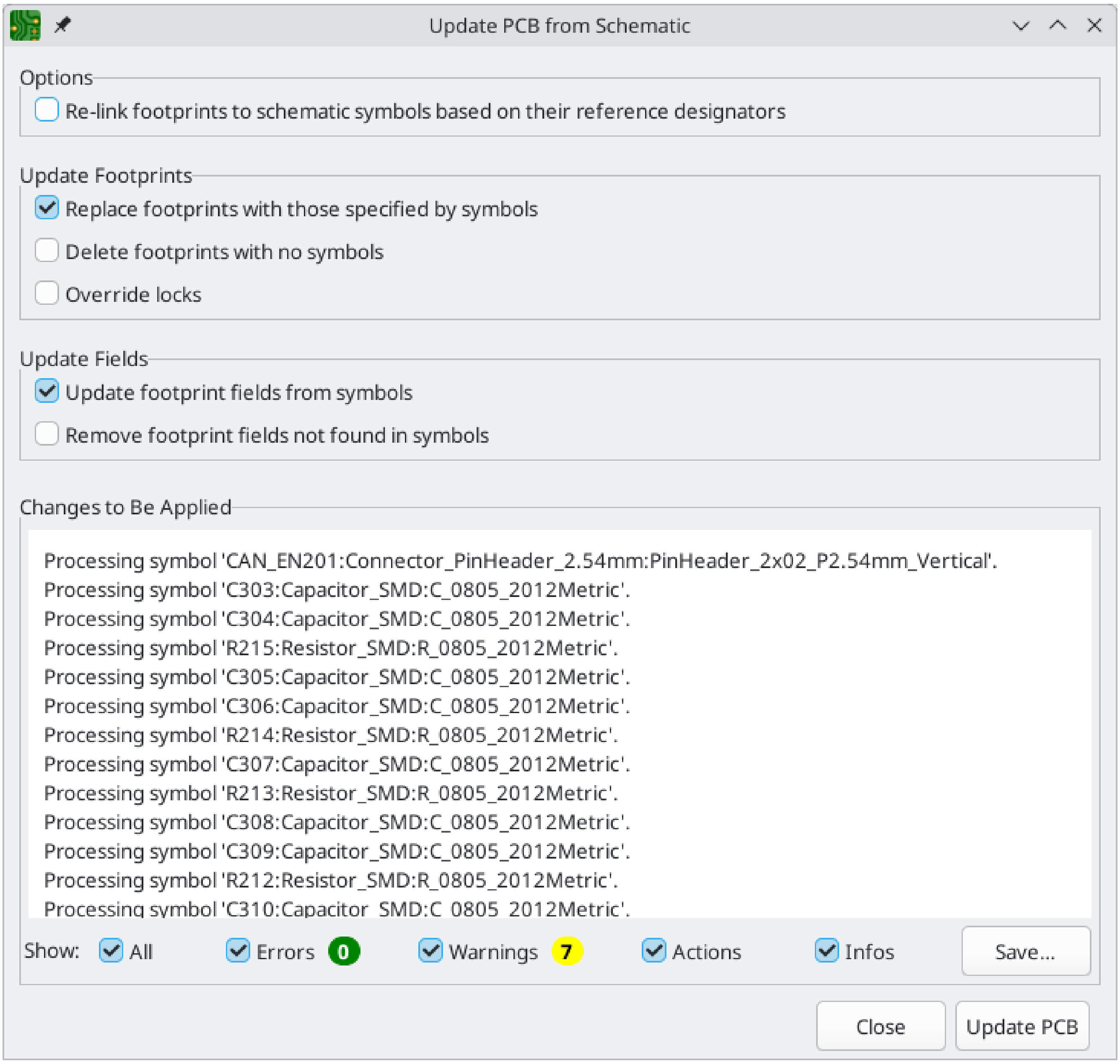
The 3D Viewer is described in the PCB Editor manual.

Forward and back annotation

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.



Update PCB from Schematic

Options

☐ Re-link footprints to schematic symbols based on their reference designators

Update Footprints

☑ Replace footprints with those specified by symbols

☐ Delete footprints with no symbols

☐ Override locks

Update Fields

☑ Update footprint fields from symbols

☐ Remove footprint fields not found in symbols

Changes to Be Applied

```
Processing symbol 'CAN_EN201:Connector_PinHeader_2.54mm:PinHeader_2x02_P2.54mm_Vertical'.
Processing symbol 'C303:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'C304:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'R215:Resistor_SMD:R_0805_2012Metric'.
Processing symbol 'C305:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'C306:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'R214:Resistor_SMD:R_0805_2012Metric'.
Processing symbol 'C307:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'R213:Resistor_SMD:R_0805_2012Metric'.
Processing symbol 'C308:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'C309:Capacitor_SMD:C_0805_2012Metric'.
Processing symbol 'R212:Resistor_SMD:R_0805_2012Metric'.
Processing symbol 'C310:Capacitor_SMD:C_0805_2012Metric'.
```

Show: ☑ All ☑ Errors 0 ☑ Warnings 7 ☑ Actions ☑ Infos

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. Symbols with the

Exclude from board attribute are not transferred to the PCB.

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
Re-link footprints to schematic symbols based on their reference designators	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. 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. This option should generally be left unchecked. See below for more details on when to use this option.
Group footprints based on symbol group	If checked, footprints will be added to groups in the PCB if their linked symbols are grouped.
Replace footprints with those specified by symbols	If checked, footprints in the PCB will be replaced with the footprint that is specified in the corresponding schematic symbol. 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.
Delete footprints with no symbols	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. If unchecked, footprints without a corresponding symbol will not be deleted.
Override locks	If checked, locking a footprint will not affect whether a footprint is deleted or replaced based on changes in the schematic. If unchecked, locked footprints will never be deleted or replaced even if they otherwise would be.
Update footprint fields from symbols	If checked, new and updated fields in symbols will be transferred to the corresponding footprints, keeping symbol and footprint fields in sync. If unchecked, footprint fields will not be updated when fields change in the corresponding symbols.
Remove footprint fields not found in symbols	If checked, footprint fields will be removed if they do not exist in the corresponding symbol. 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.

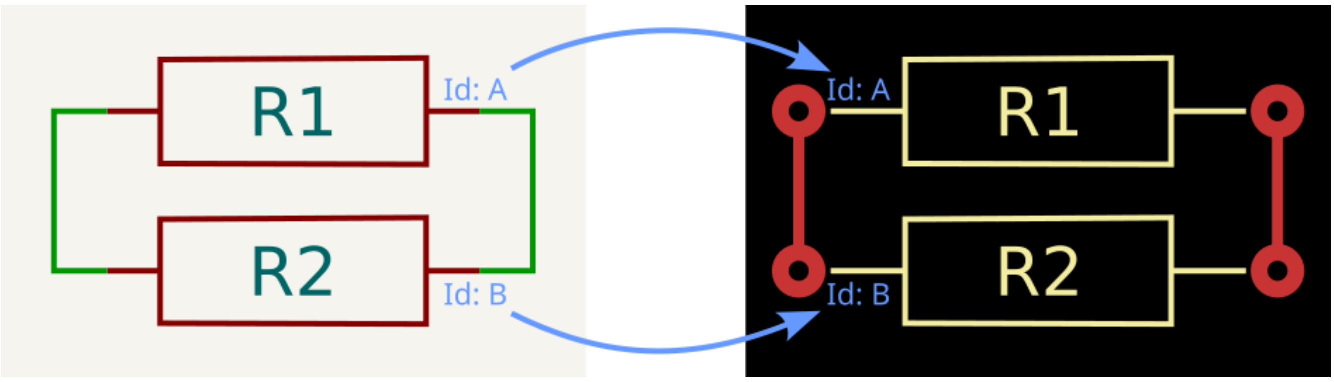
Re-linking symbols and footprints

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.

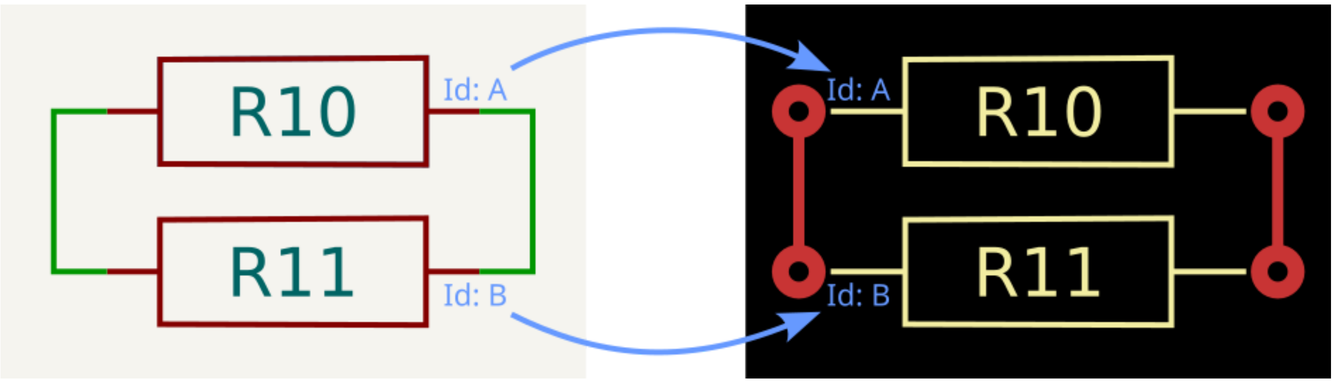
Re-linking by unique identifier (default)

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:



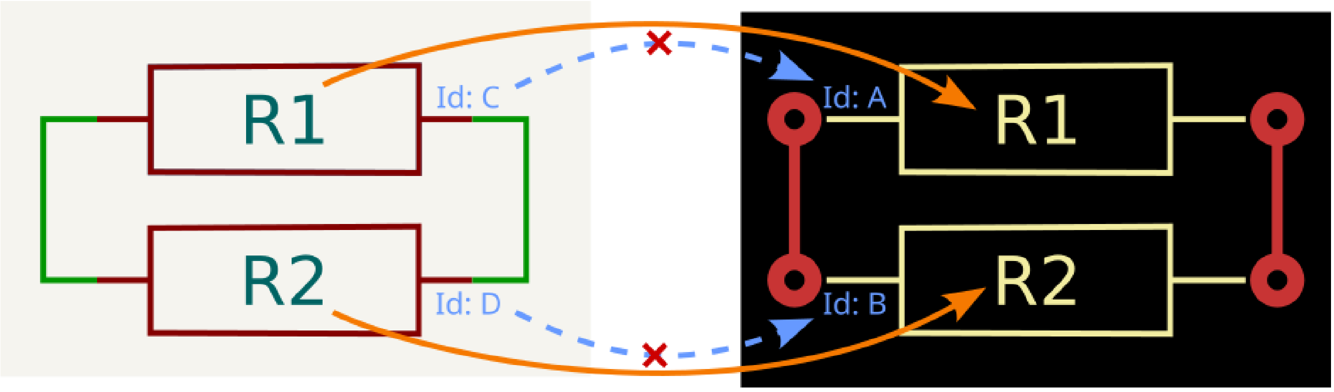
If symbol reference designators are changed in the schematic (e.g. by re-annotation), running the **Update PCB from Schematic** process will update the reference designators on the PCB.



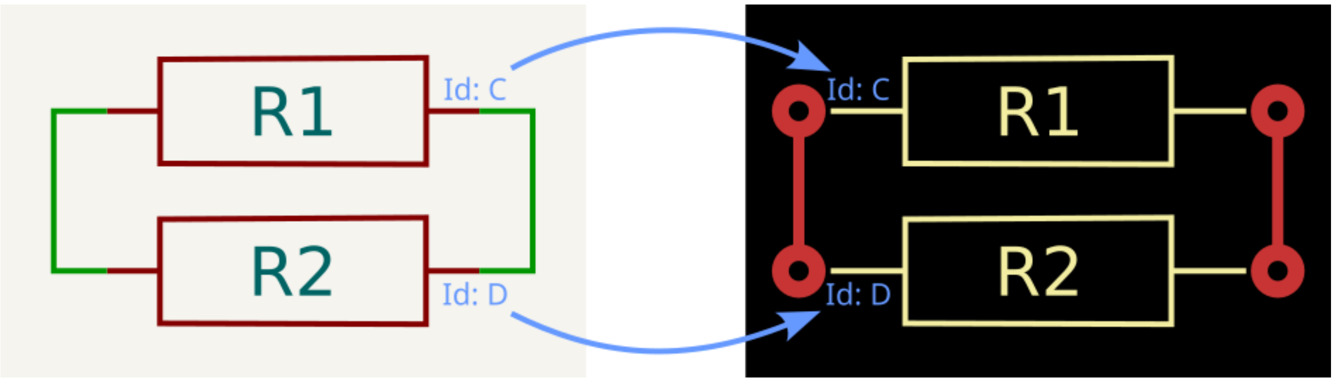
Re-linking by reference designator

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.



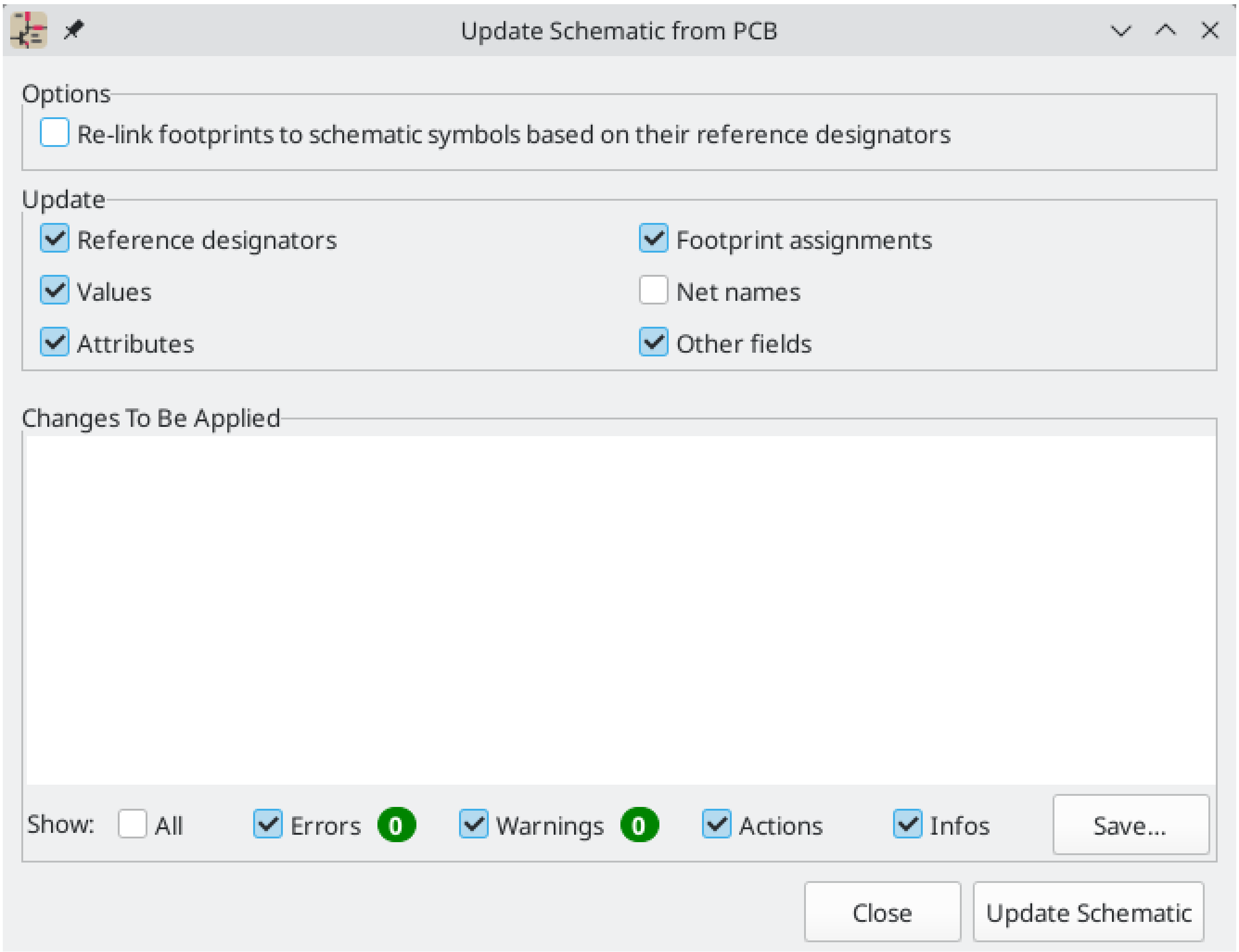
Re-linking the footprints by reference designator causes KiCad to re-create the links, using the matching reference designators as a guide.



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 also known as backannotation.



Options

☐ Re-link footprints to schematic symbols based on their reference designators

Update

☑ Reference designators
☑ Values
☑ Attributes
☑ Footprint assignments
☐ Net names
☑ Other fields

Changes To Be Applied

Show: ☐ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

The tool syncs changes in reference designators, values, attributes (like DNP or Exclude From BOM), footprint assignments, other fields, 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	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. If unchecked, footprints and symbols will be linked by unique identifier as usual, rather than by reference designator.
Reference designators	If checked, symbol reference designators will be updated to match the reference designators of the linked footprints. If unchecked, symbol reference designators will not be updated.
Values	If checked, symbol values will be updated to match the values of the linked footprints. If unchecked, symbol values will not be updated.
Attributes	If checked, symbol attributes (like exclude from BOM and DNP) will be updated to match the corresponding attributes of the linked footprints. If unchecked, symbol attributes will not be updated.
Footprint assignments	If checked, footprint assignments will be updated for symbols which have had their footprints changed or replaced in the board. If unchecked, symbol footprint assignments will not be updated.
Net names	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. If unchecked, net names will not be updated in the schematic.
Other fields	If checked, other symbol fields will be updated to match the corresponding fields of the linked footprints. Reference designator, value, and footprint are each controlled by their own separate option. If unchecked, other fields will not be updated in the schematic.

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

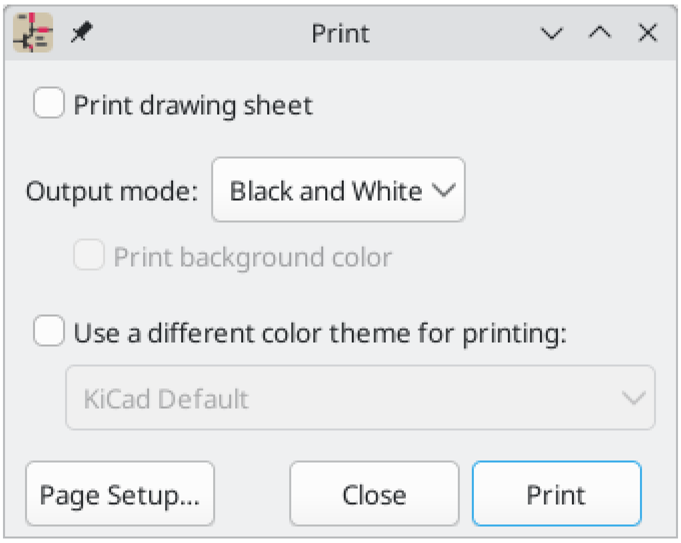
Select changes can also be synced from the PCB back to the schematic by exporting a CMP file from the PCB editor (**File** → **Export** → **Footprint Association (.cmp) File...**) and importing it in the Schematic Editor (**File** → **Import** → **Footprint Assignments...**).

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.

Generating Outputs

Printing

KiCad can print the schematic to a standard printer using **File→Print…**.



□ Print drawing sheet

Output mode: Black and White

□ Print background color

□ Use a different color theme for printing:

KiCad Default

Printing options

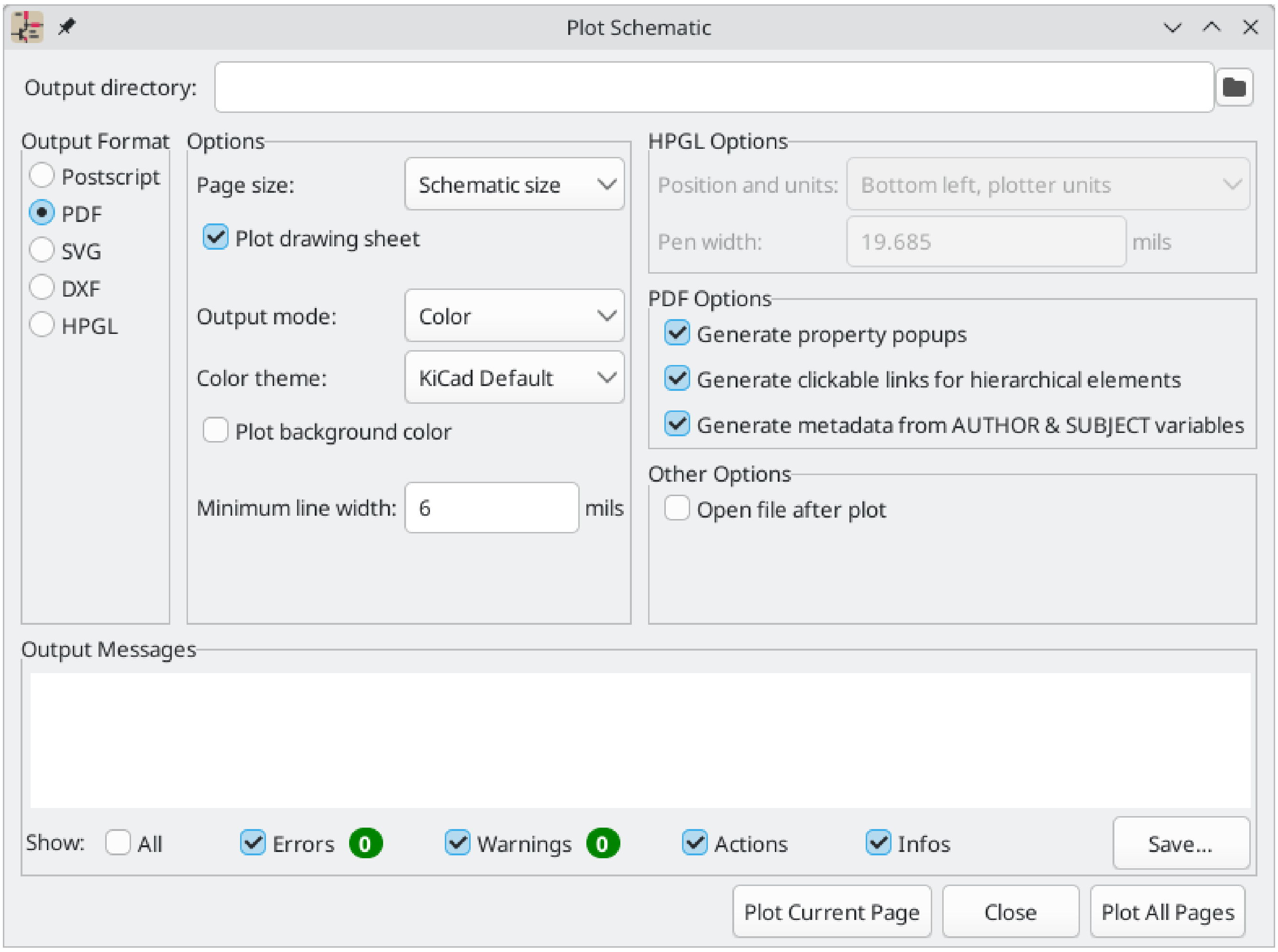
- **Print drawing sheet:** Include the drawing sheet border and title block in the printed schematic.
- **Output mode:** Print the schematic in color or black and white.
- **Print background color:** Include the background color in the printed schematic. This option is only enabled when printing in color.
- **Use a different color theme for printing:** Select a different color scheme for printing than the one selected for display in the Schematic Editor.
- **Page Setup…:** Opens a page setup dialog for setting paper size and orientation.
- **Close:** Closes the dialog without printing.
- **Print:** Opens the system print dialog.

NOTE Printing uses platform- and printer-specific drivers and may have unexpected results. When printing to a file, **Plotting** is recommended instead of **Printing**.

Plotting

KiCad can plot schematics to a file using **File → Plot…**.

The supported output formats are Postscript, PDF, SVG, and DXF.



Output directory:

Output Format
○ Postscript
● PDF
○ SVG
○ DXF
○ HPGL

Options
Page size: Schematic size
☑ Plot drawing sheet
Output mode: Color
Color theme: KiCad Default
□ Plot background color
Minimum line width: 6 mils

HPGL Options
Position and units: Bottom left, plotter units
Pen width: 19.685 mils

PDF Options
☑ Generate property popups
☑ Generate clickable links for hierarchical elements
☑ Generate metadata from AUTHOR & SUBJECT variables

Other Options
□ Open file after plot

Output Messages
Show: □ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

The **Output Messages** pane displays messages about the generated files. Different kinds of messages can be shown or hidden using the checkboxes, and the messages can be saved to a file using the **Save...** button.

The **Plot Current Page** button plots the current page of the schematic. The **Plot All Pages** button plots all pages of the schematic. One file is generated for each page, except for PDF output, which plots each schematic page as a separate page in a single PDF file.

Plotting options

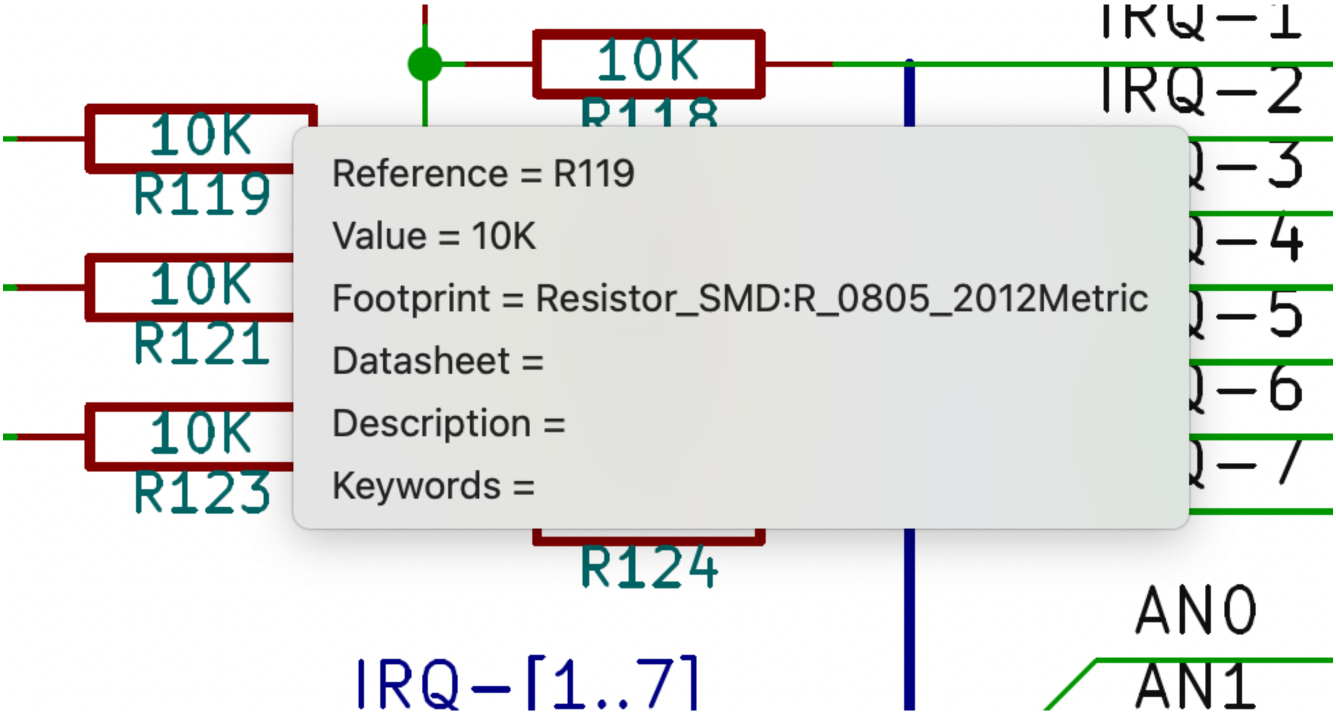
- **Output directory:** Specify the location to save plotted files. If this is a relative path, it is created relative to the project directory. This path can use text variables, including both project text variables and built-in text variables.
- **Output Format:** Select the format to plot in. Some formats have different options than others.
- **Page size:** Sets the page size to use for the plotted output. This can be set to match the schematic size or to another sheet size.
- **Plot drawing sheet:** Include the drawing sheet border and title block in the printed schematic.
- **Output mode:** Sets the output to color or black and white. Not all output formats support color.
- **Color theme:** Selects the color theme to use for the plotted output.
- **Plot background color:** Includes the schematic background color in the plotted output. The background color will not be plotted if the output format does not support color or the output mode is black and white.

Minimum line width: Selects the minimum width for lines. Any lines narrower than this width will be plotted with this minimum width.

- **Generate property popups:** Enables the interactive PDF features described below. This option only applies for PDF output.
- **Generate clickable links for hierarchical elements:** Enables clickable hierarchical sheets, hierarchical sheet pins, and hierarchical labels. When enabled, clicking a hierarchical sheet or sheet pin in the PDF will open the PDF page for that subsheet. Clicking a hierarchical label will open the page for the parent sheet. If **Generate property popups** is also enabled, links will be generated instead of property popups for hierarchical sheets, pins, and labels (i.e. this option takes priority). This option only applies for PDF output.
- **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. This option only applies for PDF output.
- **Export units:** Sets the units (inches or millimeters) in the exported DXF file.
- **Open file after plot:** automatically opens the plotted output file when plotting is complete.

Interactive PDF features

Plotted PDFs can optionally have several interactive features.



Reference = R119

Value = 10K

Footprint = Resistor_SMD:R_0805_2012Metric

Datasheet =

Description =

Keywords =

- Hyperlinks can be clicked.
- The table of contents is populated with schematic sheets as well as the symbols and hierarchical labels in each sheet.
- Clicking on many schematic elements displays a popup menu containing relevant information.
  - Symbols display their symbol fields.
  - Hierarchical subsheets display their sheetname and filename, as well as an option to enter the sheet itself. This is replaced by a direct link to the subsheet if the **Generate clickable links for**

hierarchical elements option is enabled.

- Labels display the resolved net and netclass.
- Buses display their members.

> **NOTE** Some of these features are not supported in all PDF readers. The clickable links generated by the **Generate clickable links for hierarchical elements** option are more widely supported than other interactive features.

Generating a bill of materials

KiCad can generate a bill of materials that lists all of the components in the design. BOMs are configurable: you can select which components are included, how components are ordered, which symbol fields are included and in what order, and what the output format is.

BOMs are exported using the Symbol Fields Table. As a shortcut to open the **Export** tab of this dialog, you can select **Tools → Generate Bill of Materials...** or use the .bom button on the top toolbar.

The contents of the BOM are configured in the **Edit** tab. The format of the exported BOM file is configured in the **Export** tab. The BOM is written when you press the **Export** button at the bottom of the dialog.

BOM contents

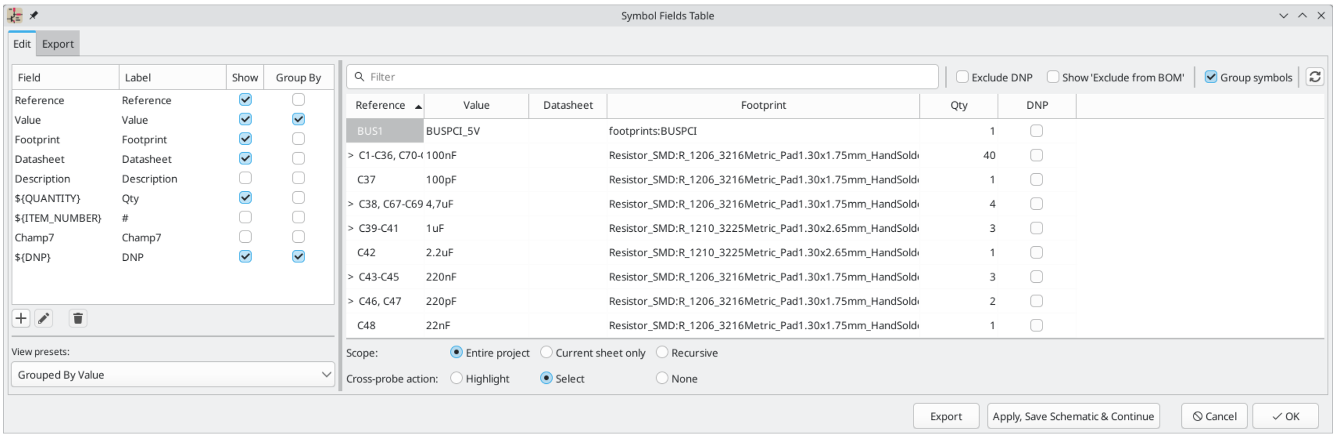
The exported BOM will contain the components (rows) and fields (columns) shown in the **Edit** tab, with the same grouping and sorting.

The **Exclude from BOM** and **Do not Populate** attributes affect the **Edit** tab and the exported BOM as follows:

- Components with the **Exclude from BOM** attribute set are always excluded from BOM exports. They are shown in the **Edit** tab unless the **Include 'Exclude from BOM' Symbols** option is unchecked in the ⚙ menu.
- Components with the **Do not Populate** (DNP) attribute set are included in BOM exports and shown in the **Edit** tab unless the **Include 'DNP' Symbols** option is unchecked in the ⚙ menu.

You can filter the components that are exported by typing in the **Filter** textbox. This matches against symbols' references. Wildcards are supported: `*` matches any number of any characters, including none, and `?` matches any single character.

You can also limit the displayed components to those in the current sheet, the current sheet and all of its subsheets, or the entire schematic using the dropdown menu next to the filter box.



Field	Label	Show	Group By
Reference	Reference	☑	□
Value	Value	☑	☑
Footprint	Footprint	☑	□
Datasheet	Datasheet	☑	□
Description	Description	□	□
${QUANTITY}	Qty	☑	□
${ITEM_NUMBER}	#	□	□
Champ7	Champ7	□	□
${DNP}	DNP	☑	☑

View presets: Grouped By Value

□ Exclude DNP □ Show 'Exclude from BOM' ☑ Group symbols

Reference	Value	Datasheet	Footprint	Qty	DNP
BUS1	BUSPCI_5V		footprints:BUSPCI	1	□
> C1-C36, C70-(	100nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	40	□
C37	100pF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	1	□
> C38, C67-C69	4,7uF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	4	□
> C39-C41	1uF		Resistor_SMD:R_1210_3225Metric_Pad1.30x2.65mm_HandSold	3	□
C42	2.2uF		Resistor_SMD:R_1210_3225Metric_Pad1.30x2.65mm_HandSold	1	□
> C43-C45	220nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	3	□
> C46, C47	220pF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	2	□
C48	22nF		Resistor_SMD:R_1206_3216Metric_Pad1.30x1.75mm_HandSold	1	□

Scope: ● Entire project ○ Current sheet only ○ Recursive

Cross-probe action: ○ Highlight ● Select ○ None

The panel on the left side of the dialog contains a list of columns that are available to show in the **Edit** tab and include in the BOM. You can collapse this panel by clicking the ⬅ button.

Fields with the **Include** box checked will be shown in the **Edit** tab and included as columns in the BOM. You can also show or hide columns by right clicking the column header in the right part of the dialog and selecting the desired columns. To reorder the columns, drag a column header to the left or right. You can set an arbitrary name for each column by clicking in the **BOM Name** column and entering a new name.

Fields with the **Group By** box checked are used to group components together. Components are grouped into the same line if all of their **Group By** fields are identical and the **Group symbols** box at the top of the dialog is checked.

Presets are available to configure the list of fields. Presets store which fields are displayed, which fields are used for grouping, and the column order. You can create and save your own presets or use one of several default presets. Custom presets can be deleted in this dialog or in the Schematic Setup dialog.

The built-in presets "Grouped By Value" and "Grouped By Value and Footprint" replicate legacy BOM scripts, while "Attributes" shows only the reference and value fields and the DNP, exclude from board, exclude from simulation, and exclude from BOM attributes.

Some virtual fields are available that may be useful in BOM exports. Adding a field in the Symbol Fields Table beginning with a text variable will not create a new field in the symbols, but will create a special column in the table and BOM with auto-generated values for each component. The following variables may be especially useful for creating virtual fields in custom BOM formats:

- `${QUANTITY}` creates a field that contains the number of grouped instances of that component.
- `${ITEM_NUMBER}` creates a field that contains the row number of the component in the BOM.
- `${SYMBOL_NAME}` creates a field that contains the name of the schematic symbol.
- `${SYMBOL_LIBRARY}` creates a field that contains the name of the schematic symbol library.
- `${DNP}` creates a field with a checkbox that controls the component's DNP attribute. In the BOM, this field resolves to the string "DNP" if the component's DNP attribute is set, or an empty string otherwise.
- `${EXCLUDE_FROM_BOARD}` creates a field with a checkbox that controls the component's exclude from board attribute. In the BOM, this field resolves to the string "Excluded from board" if the component's exclude from board attribute is set, or an empty string otherwise.
-

`${EXCLUDE_FROM_SIM}` creates a field with a checkbox that controls the component's exclude from simulation attribute. In the BOM, this field resolves to the string "Excluded from simulation" if the component's exclude from simulation attribute is set, or an empty string otherwise.

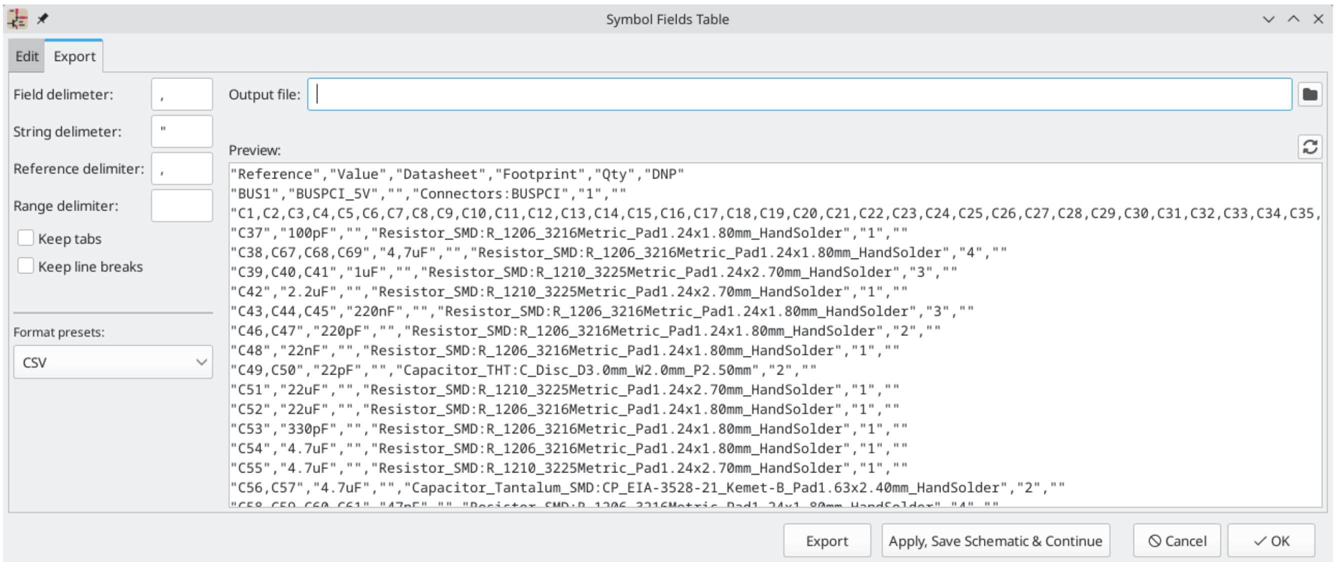
- `${EXCLUDE_FROM_BOM}` creates a field with a checkbox that controls the component's exclude from BOM attribute. Components with the exclude from BOM attribute set are not included in the BOM.

Other text variables are also available.

The full functionality of the **Edit** tab, including virtual field behavior, is explained in more detail in the Symbol Fields Table documentation.

BOM format

The **Export** tab contains settings concerning the output file format for the BOM and displays a preview of the raw BOM file output.



Field delimiter: ,
String delimiter: "
Reference delimiter: ,
Range delimiter:
□ Keep tabs
□ Keep line breaks
Format presets: CSV

Output file:

Preview:

```
"Reference","Value","Datasheet","Footprint","Qty","DNP"
"BUS1","BUSPCI_5V","","Connectors:BUSPCI","1",""
"C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,C33,C34,C35,
"C37","100pF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","1",""
"C38,C67,C68,C69","4,7uF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","4",""
"C39,C40,C41","1uF","","Resistor_SMD:R_1210_3225Metric_Pad1.24x2.70mm_HandSolder","3",""
"C42","2.2uF","","Resistor_SMD:R_1210_3225Metric_Pad1.24x2.70mm_HandSolder","1",""
"C43,C44,C45","220nF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","3",""
"C46,C47","220pF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","2",""
"C48","22nF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","1",""
"C49,C50","22pF","","Capacitor_THT:C_Disc_D3.0mm_W2.0mm_P2.50mm","2",""
"C51","22uF","","Resistor_SMD:R_1210_3225Metric_Pad1.24x2.70mm_HandSolder","1",""
"C52","22uF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","1",""
"C53","330pF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","1",""
"C54","4.7uF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","1",""
"C55","4.7uF","","Resistor_SMD:R_1210_3225Metric_Pad1.24x2.70mm_HandSolder","1",""
"C56,C57","4.7uF","","Capacitor_Tantalum_SMD:CP_EIA-3528-21_Kemet-B_Pad1.63x2.40mm_HandSolder","2",""
"C58,C59,C60,C61","47nF","","Resistor_SMD:R_1206_3216Metric_Pad1.24x1.80mm_HandSolder","4",""
```

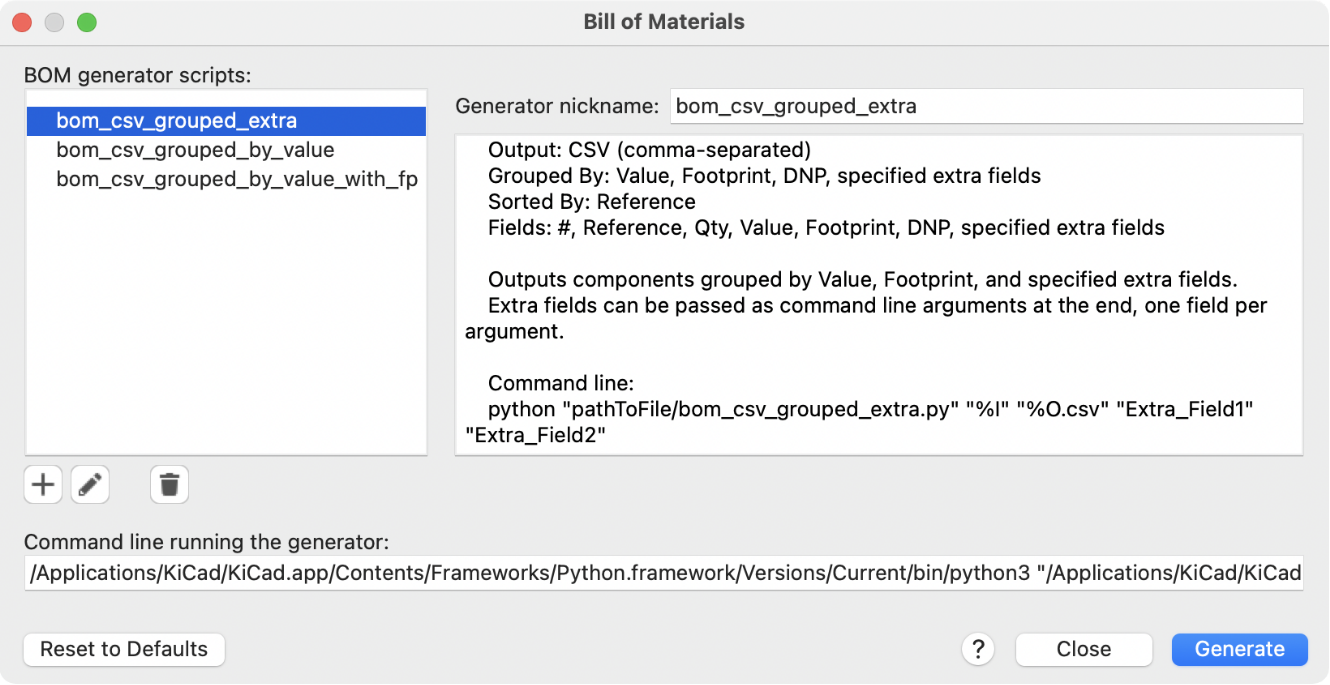
At the top you can specify the output file. Pressing the **Export** button will write the BOM to this file path. This path can contain text variables.

The settings on the left control how the BOM information is formatted in the file. You can change the delimiter between fields, the delimiter that surrounds each field, the delimiter that separates a sequence of references (e.g. the comma in `R1,R3`), and the delimiter for a range of references (e.g. the dash in `R1-R3`). If no range delimiter is given, ranges will not be used: `R1-R3` will be written out as `R1,R2,R3`, for example, assuming `,` as a reference delimiter. Tabs and newlines in fields can be preserved or stripped, depending on the **Keep tabs** and **Keep line breaks** settings.

Several default format presets are available. You can select a comma-separated value (CSV) format, a tab-separated value (TSV) format, or a semicolon-separated format. You can also create and save your own presets. Custom presets can be deleted in this dialog or in the Schematic Setup dialog.

Legacy BOM generation

Previous versions of KiCad used external scripts to process the design information into the desired output format. This BOM generation tool is still available by selecting **Tools** → **Generate Legacy Bill of Materials...** .



```
   Output: CSV (comma-separated)
   Grouped By: Value, Footprint, DNP, specified extra fields
   Sorted By: Reference
   Fields: #, Reference, Qty, Value, Footprint, DNP, specified extra fields

   Outputs components grouped by Value, Footprint, and specified extra fields.
   Extra fields can be passed as command line arguments at the end, one field per
argument.

   Command line:
   python "pathToFile/bom_csv_grouped_extra.py" "%I" "%O.csv" "Extra_Field1"
"Extra_Field2"
```

```
/Applications/KiCad/KiCad.app/Contents/Frameworks/Python.framework/Versions/Current/bin/python3 "/Applications/KiCad/KiCad
```

Several BOM generator scripts are included with KiCad, and users can also create their own. BOM generator scripts generally use Python or XSLT, but other tools can be used as long as you can specify a command line for KiCad to execute when running the generator.

You can select which BOM generator to use in the **BOM generator scripts** list. The rest of the dialog displays information about the selected generator. You can change the displayed name of the generator with the **Generator nickname** textbox.

The pane at right displays information about the selected script. When the generator is executed, the right pane instead displays output from the script.

The text box at the bottom contains the command that KiCad will use to execute the generator. It is automatically populated when a script is selected, but the command may need to be hand-edited for some generators. KiCad saves the command line for each generator when the BOM tool is closed, so command line customizations are preserved. For more details about the command line, see the advanced documentation.

On Windows, the BOM Generator dialog has an additional option **Show console window**. When this option is unchecked, BOM generators run in a hidden console window and any output is redirected and printed in the dialog. When this option is checked, BOM generators run in a visible console window, which may be necessary if the generator plugin provides a graphical user interface.

BOM generator scripts

By default, the legacy BOM tool presents three output script options.

- `bom_csv_grouped_extra` outputs a CSV with a single section containing every component in the design. Components are grouped by value, footprint, DNP (do not populate), and any additional fields that are specified on the command line. To specify extra fields, add the desired field names as quoted strings at the end of the command line. For example, to include the `MPN` field, the end of the command line would be: `<path to script>/bom_csv_grouped_extra.py "%I" "%O.csv" "MPN"`. The columns in the BOM are:
    - Line item number
    - Reference designator(s)

  - Quantity
  - Value
  - Footprint
  - DNP
  - Specified extra fields
- `bom_csv_grouped_by_value` outputs a CSV with two sections. The first section contains every component in the design, with a single component on each line. The second section also contains every component, but components are grouped by symbol name, value, footprint, and DNP (do not populate). The columns in the BOM are:
  - Line item number
  - Quantity
  - Reference designator(s)
  - Value
  - Symbol library and symbol name
  - Footprint
  - Datasheet
  - DNP
  - Any other symbol fields
- `bom_csv_grouped_by_value_with_fp` outputs a CSV with a single section containing every component in the design. Components are grouped by value, footprint, and DNP (do not populate). The columns in the BOM are:
  - Reference designator(s)
  - Quantity
  - Value
  - Symbol name
  - Footprint
  - Symbol description
  - Vendor
  - DNP

Additional generator scripts are installed with KiCad but are not populated in the generator script list by default. The location of these scripts depends on the operating system and may vary based on installation location.

Operating System	Location
Windows	`C:\Program Files\KiCad\10.0\bin\scripting\plugins\`
Linux	`/usr/share/kicad/plugins/`
macOS	`/Applications/KiCad/KiCad.app/Contents/SharedSupport/plugins/`

Additional scripts can be added to the list of BOM generator scripts by clicking the + button. Scripts can be removed by clicking the 🗑 button. The ✏ button opens the selected script in a text editor.

For more information on creating and using custom BOM generators, see the advanced documentation.

BOM export from PCB editor

The PCB Editor can export a BOM through **File** → **Fabrication Outputs** → **BOM...**. This method provides no control over the output format and does not include all symbol information, but is useful for PCB-only workflows that do not involve a schematic. In general, it is recommended to use the schematic editor's BOM export tool instead.

Generating a Netlist

A netlist is a file which describes electrical connections between symbol pins. These connections are referred to as nets. Netlist files contain:

- A list of symbols and their pins.
- A list of connections (nets) between symbol pins.

Many different netlist formats exist. Sometimes the symbols list and the list of nets are two separate files. This netlist is fundamental in the use of schematic capture software, because the netlist is the link with other electronic CAD software, such as PCB layout software, simulators, and programmable logic compilers.

KiCad supports several netlist formats:

- KiCad format, which can be imported by the KiCad PCB Editor. However, the "Update PCB from Schematic" tool should be used instead of importing a KiCad netlist into the PCB editor.
- OrCAD PCB2 format, for designing PCBs with OrCAD.
- Allegro format, for designing PCBs with Allegro.
- PADS format, for designing PCBs with PADS.
- CADSTAR format, for designing PCBs with CADSTAR.
- Spice format, for use with various external circuit simulators.

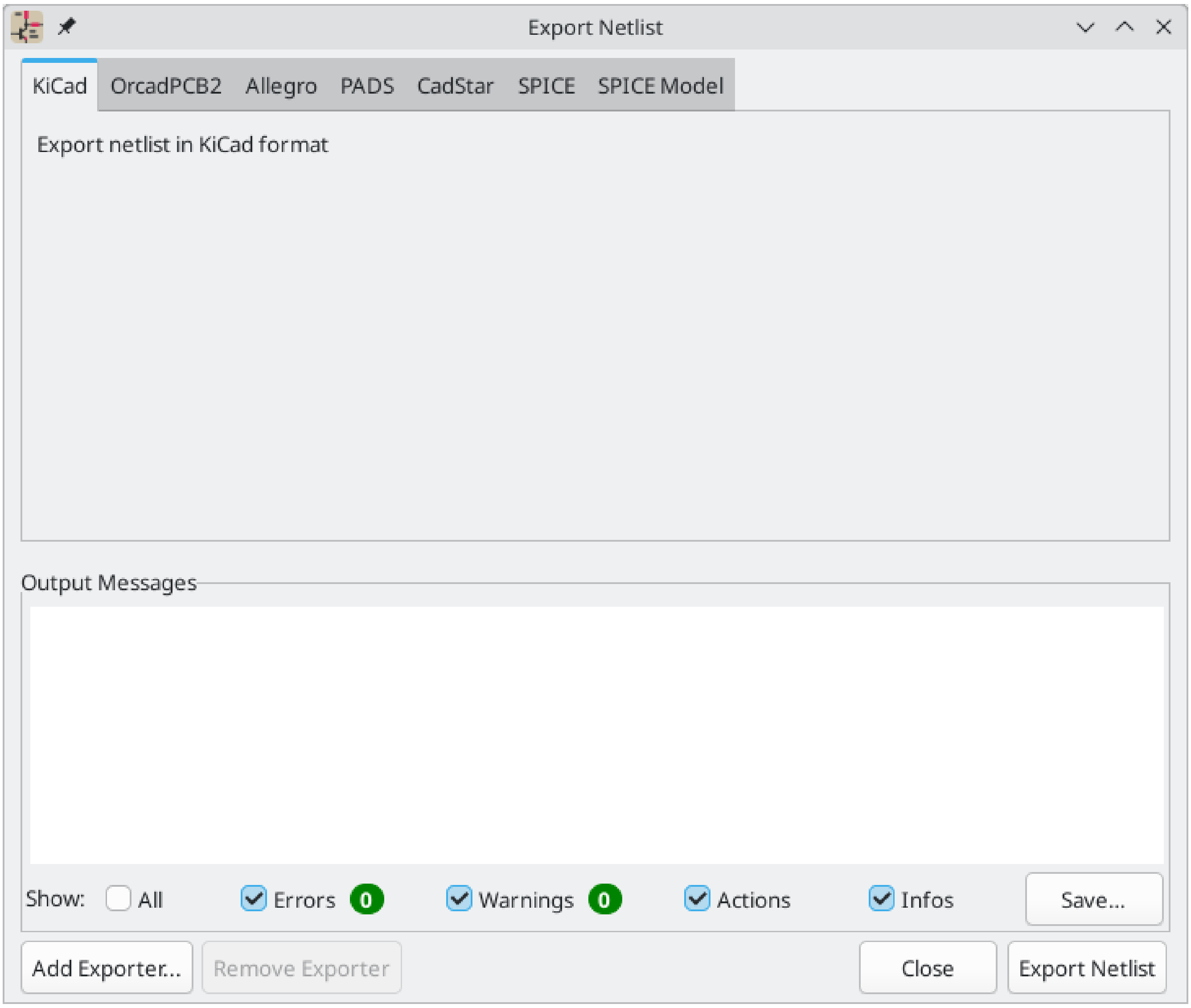
NOTE

In KiCad version 5.0 and later, it is not necessary to create a netlist for transferring a design from the schematic editor to the PCB editor. Instead, use the "Update PCB from Schematic" tool.

NOTE Other software tools that use netlists may have restrictions on spaces and special characters in component names, pins, nets, and other fields. For compatibility, be aware of such restrictions in other tools you plan to use, and name components, nets, etc. accordingly.

Format-format Netlist

Netlists are exported with the Export Netlist dialog (**File** → **Export** → **Netlist...**).



Export netlist in KiCad format

Show: ☐ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

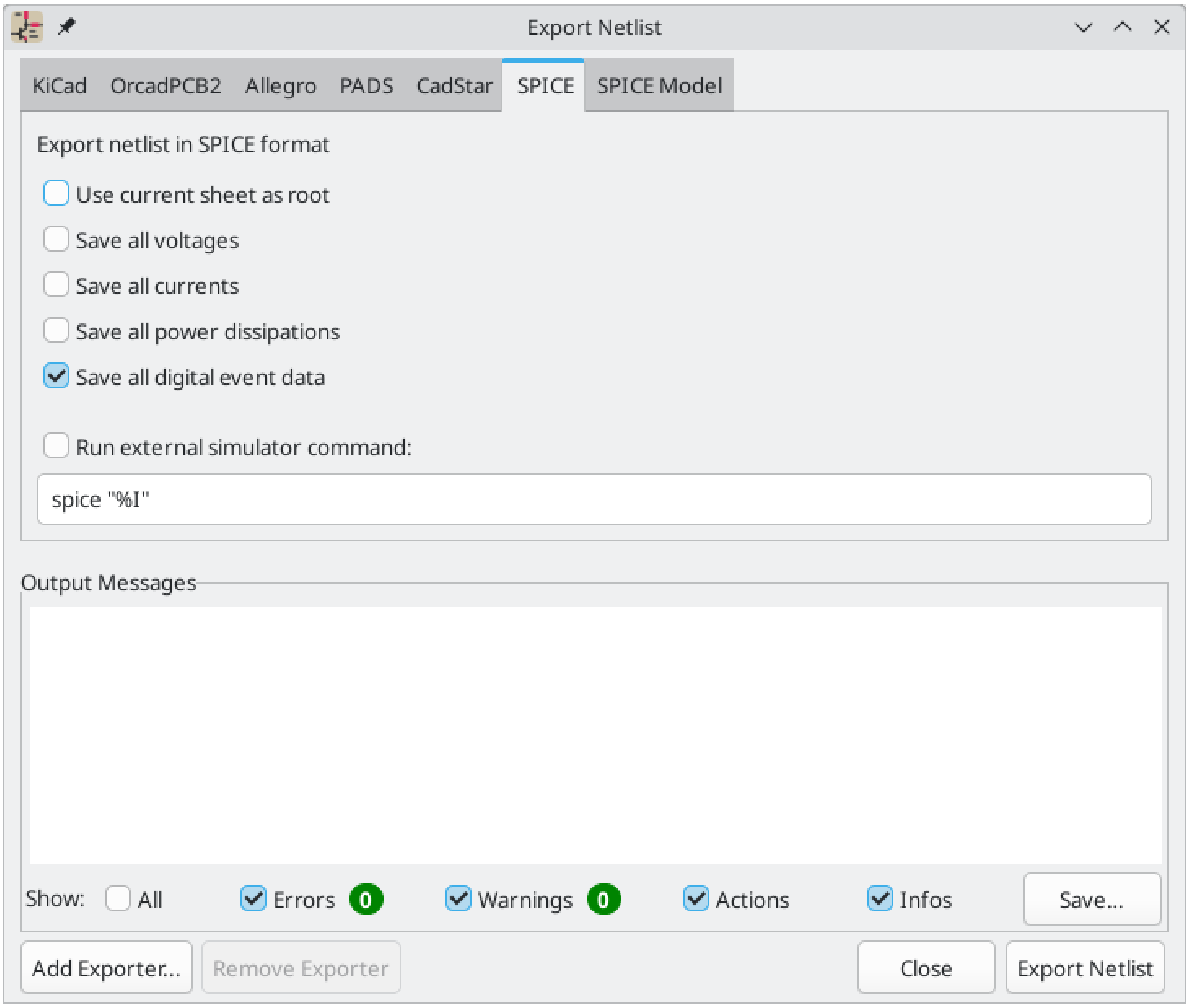
KiCad supports exporting netlists in several formats: KiCad, OrcadPCB2, Allegro, PADS, CADSTAR, Spice, and Spice Model. Each format can be selected by selecting the corresponding tab at the top of the window. Some netlist formats have additional options.

Clicking the **Export Netlist** button prompts for a netlist filename and saves the netlist.

NOTE Netlist generation can take up to several minutes for large schematics.

Custom generators for other netlist formats can be added by clicking the **Add Generator...** button. Custom generators are external tools that are called by KiCad, for example Python scripts or XSLT stylesheets. For more information on custom netlist generators, see the section on adding custom netlist generators.

Spice Netlist Format



Export Netlist

Export netlist in SPICE format

☐ Use current sheet as root

☐ Save all voltages

☐ Save all currents

☐ Save all power dissipations

☑ Save all digital event data

☐ Run external simulator command:

spice "%I"

Output Messages

Show: ☐ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

The Spice netlist format offers several options.

- When the **use current sheet as root** is selected, only the current sheet is exported to a subcircuit model. Otherwise, the entire schematic sheet is exported.
- The **Save all voltages** option adds a `.save all` command to the netlist, which causes the simulator to save all node voltages.
- The **Save all currents** option adds a `.probe alli` command to the netlist, which causes the simulator save all node currents.
- The **Save all power dissipations** adds `.probe` commands to save the power dissipation in each component.
- The **Save all digital event data** removes the `esave none` command from the netlist, which causes digital event data to be saved. Digital event data may consume a lot of memory.

NOTE Exact behavior may vary between simulation tools.

Passive symbol values are automatically adjusted to be compatible with various Spice simulators. Specifically:

- `µ` and `M` as unit prefixes are replaced with `u` and `Meg`, respectively
- Units are removed (e.g. `4.7kΩ` is changed to `4.7k`)
- Values in RKM format are rewritten to be Spice-compatible (e.g. `4u7` is changed to `4.7u`)

The Spice netlist exporter also provides an easy way to simulate the generated netlist with an external simulator. This can be useful for running a simulation without using KiCad's internal ngspice simulator, or for running an ngspice simulation with options that are not supported by KiCad's simulator tool.

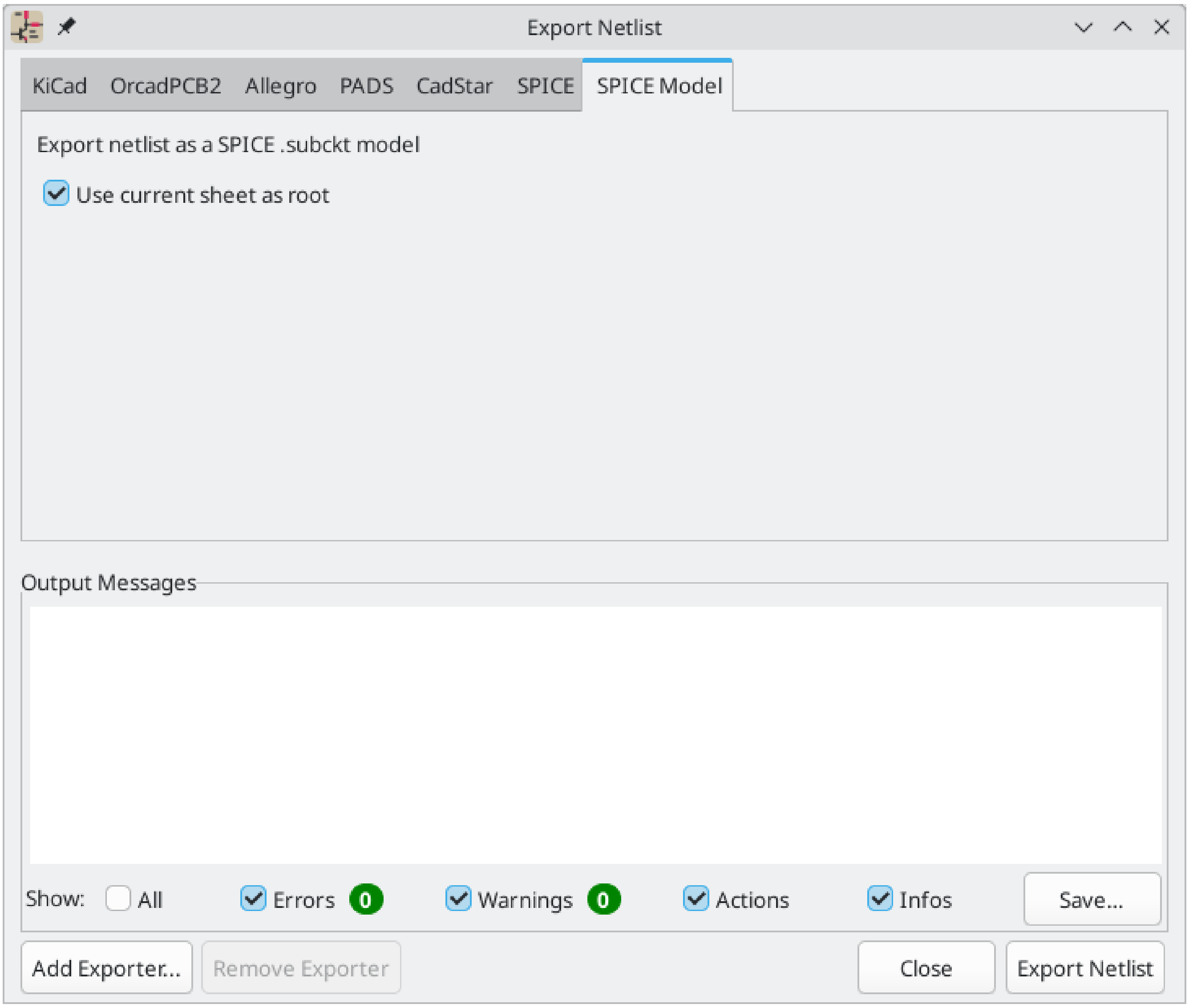
Enter the path to the external simulator in the text box, with `%I` representing the generated netlist. Check the **run external simulator command** box to generate the netlist and automatically run the simulator.

NOTE The default simulator command (`spice "%I"`) must be adjusted to point to a simulator installed on your system.

Spice simulators expect simulation commands (`.PROBE`, `.AC`, `.TRAN`, etc.) to be included in the netlist. Any text line included in the schematic diagram starting with a period (`.`) will be included in the netlist. If a text object contains multiple lines, only the lines beginning with a period will be included.

`.include` directives for including model library files are automatically added to the netlist based on the Spice model settings for the symbols in the schematic.

Spice Model Netlist Format



Export netlist as a SPICE .subckt model

☑ Use current sheet as root

Show: ☐ All ☑ Errors 0 ☑ Warnings 0 ☑ Actions ☑ Infos

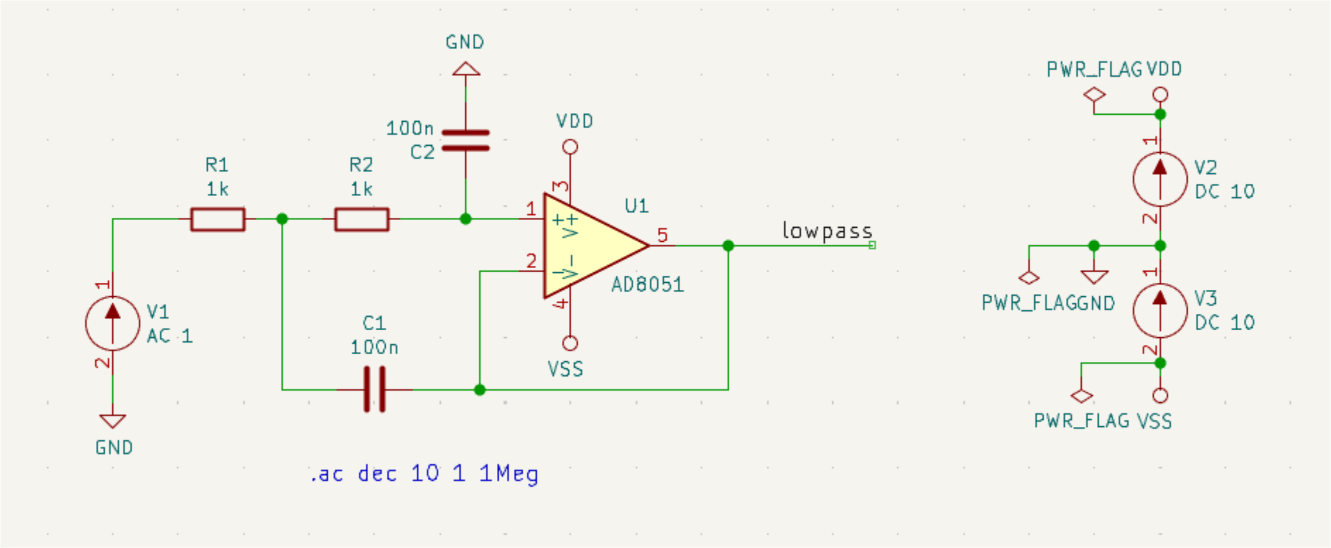
KiCad can also export a netlist of the schematic as a Spice subcircuit model, which can be included in a separate Spice simulation. Any hierarchical labels in the schematic are used as pins for the subcircuit model. Each pin in the model is annotated with a comment describing the pin's electrical direction:

- `Input` hierarchical labels are mapped to an `input` annotation
- `Output` hierarchical labels are mapped to an `output` annotation
- `Bidirectional` hierarchical labels are mapped to an `inout` annotation
- `Tri-state` hierarchical labels are mapped to a `tristate` annotation
- `Passive` hierarchical labels are mapped to a `passive` annotation

When the **use current sheet as root** is selected, only the current sheet is exported to a subcircuit model. Otherwise, the entire schematic sheet is exported.

Contoh Netlist

Below is the schematic from the `sallen_key` project included in KiCad's simulation demos.



The KiCad format netlist for this schematic is as follows:

```
(export (version "E")
  (design
    (source "/usr/share/kicad/demos/simulation/sallen_key/sallen_key.kicad_sch")
    (date "Sun 01 May 2022 03:14:05 PM EDT")
    (tool "Eeschema (6.0.4)")
    (sheet (number "1") (name "/") (tstamps "/")
      (title_block
        (title)
        (company)
        (rev)
        (date)
        (source "sallen_key.kicad_sch")
        (comment (number "1") (value ""))
        (comment (number "2") (value ""))
        (comment (number "3") (value ""))
        (comment (number "4") (value ""))
        (comment (number "5") (value ""))
        (comment (number "6") (value ""))
        (comment (number "7") (value ""))
        (comment (number "8") (value ""))
        (comment (number "9") (value "")))))
  (components
    (comp (ref "C1")
      (value "100n")
      (libsource (lib "sallen_key_schlib") (part "C") (description ""))
      (property (name "Sheetname") (value ""))
      (property (name "Sheetfile") (value "sallen_key.kicad_sch"))
      (sheetpath (names "/") (tstamps "/"))
      (tstamps "00000000-0000-0000-0000-00005789077d"))
    (comp (ref "C2")
      (value "100n")
      (fields
        (field (name "Fieldname") "Value")
        (field (name "SpiceMapping") "1 2")
        (field (name "Spice_Primitive") "C"))
      (libsource (lib "sallen_key_schlib") (part "C") (description ""))
      (property (name "Fieldname") (value "Value"))
      (property (name "Spice_Primitive") (value "C"))
      (property (name "SpiceMapping") (value "1 2"))
      (property (name "Sheetname") (value ""))
      (property (name "Sheetfile") (value "sallen_key.kicad_sch"))
      (sheetpath (names "/") (tstamps "/"))
      (tstamps "00000000-0000-0000-0000-00005789085b"))
    (comp (ref "R1")
      (value "1k")
      (fields
        (field (name "Fieldname") "Value")
        (field (name "SpiceMapping") "1 2")
        (field (name "Spice_Primitive") "R"))
      (libsource (lib "sallen_key_schlib") (part "R") (description ""))
      (property (name "Fieldname") (value "Value"))
      (property (name "SpiceMapping") (value "1 2"))
      (property (name "Spice_Primitive") (value "R"))
      (property (name "Sheetname") (value ""))
      (property (name "Sheetfile") (value "sallen_key.kicad_sch"))
      (sheetpath (names "/") (tstamps "/"))
      (tstamps "00000000-0000-0000-0000-0000578906ff"))
    (comp (ref "R2")
      (value "1k")
      (fields
```

In Spice format, the netlist is as follows:

```
.title KiCad schematic
.include "ad8051.lib"
XU1 Net-_C2-Pad1_ /lowpass VDD VSS /lowpass AD8051
C2 Net-_C2-Pad1_ GND 100n
C1 /lowpass Net-_C1-Pad2_ 100n
R2 Net-_C2-Pad1_ Net-_C1-Pad2_ 1k
R1 Net-_C1-Pad2_ Net-_R1-Pad2_ 1k
V1 Net-_R1-Pad2_ GND AC 1
V2 VDD GND DC 10
V3 GND VSS DC 10
.ac dec 10 1 1Meg
.end
```

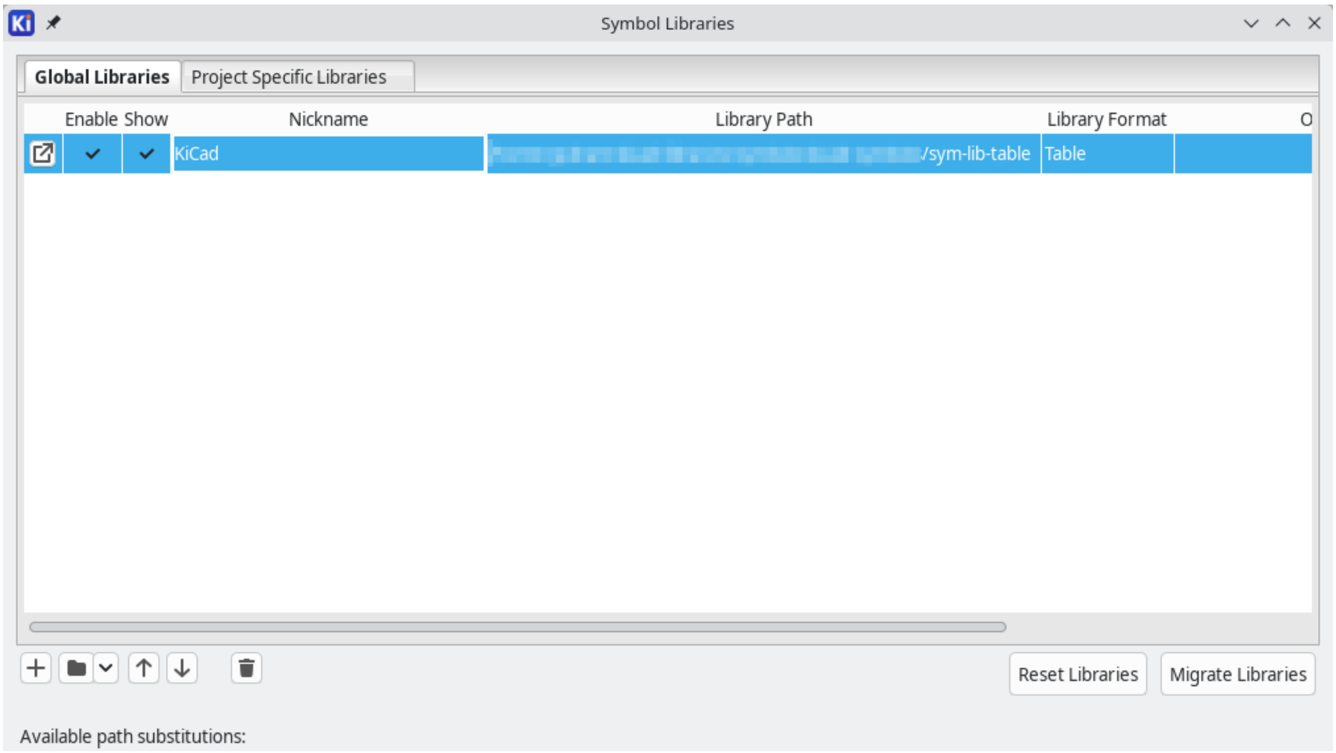
Symbols and Symbol Libraries

KiCad organizes symbols into symbol libraries, which hold collections of symbols. Each symbol in a schematic is uniquely identified by a full name that is composed of a library nickname and a symbol name. For example, the identifier `Audio:AD1853` refers to the `AD1853` symbol in the `Audio` library.

Managing symbol libraries

KiCad uses a table of symbol libraries to map a symbol library nickname to an underlying symbol library on disk. Kicad uses a global symbol 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 symbol library table, use **Preferences** → **Manage Symbol Libraries...**.

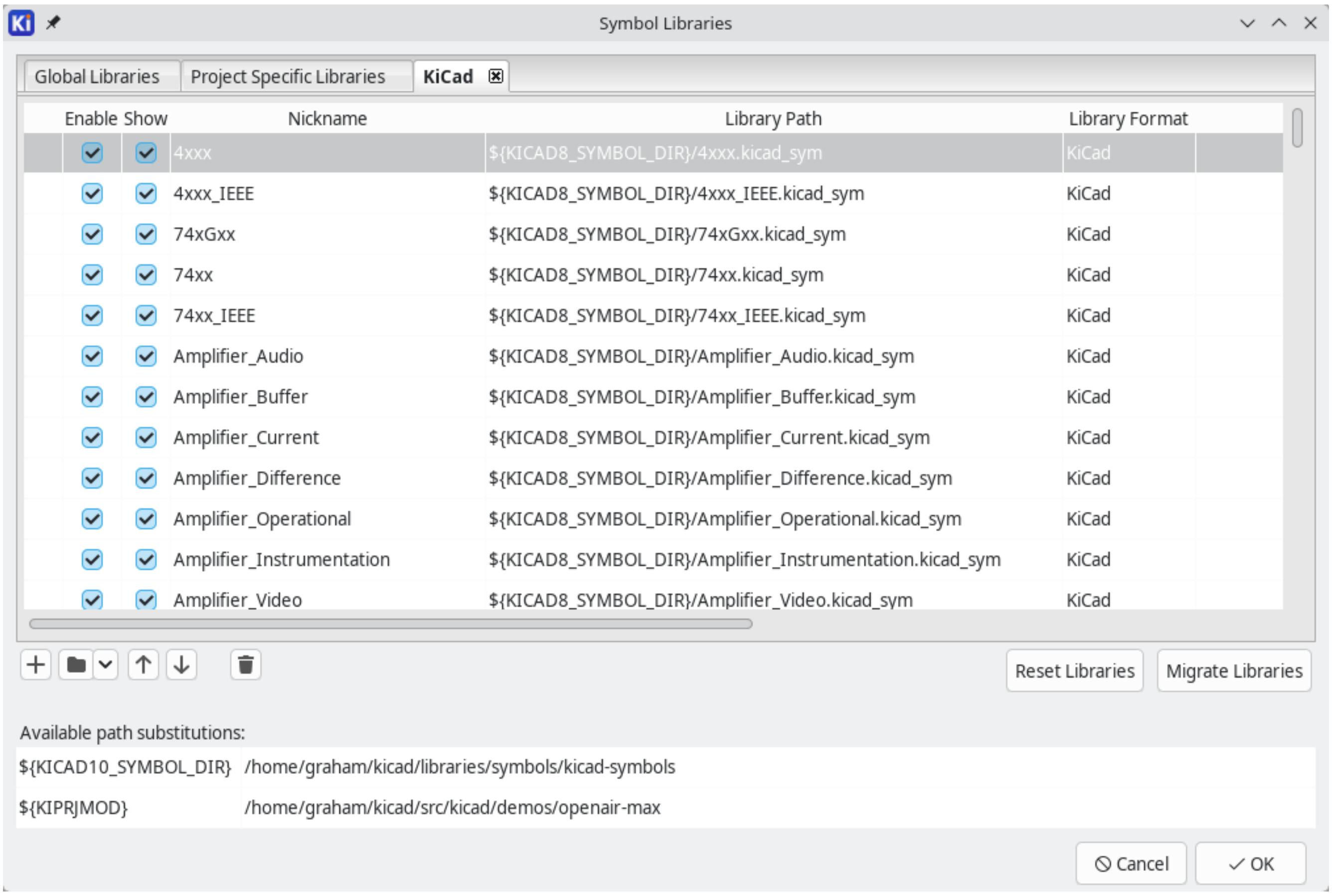


	Enable	Show	Nickname	Library Path	Library Format	O
↗	✓	✓	KiCad	[redacted]/sym-lib-table	Table	

The global symbol library table contains the list of libraries that are always available regardless of the currently loaded project. The table is saved in the file `sym-lib-table` in the KiCad configuration folder. The location of this folder depends on the operating system being used.

The project specific symbol library table contains the list of libraries that are available specifically for the currently loaded project. If there are any project-specific symbol libraries, the table is saved in the file `sym-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.



Enable	Show	Nickname	Library Path	Library Format
☑	☑	4xxx	${KICAD8_SYMBOL_DIR}/4xxx.kicad_sym	KiCad
☑	☑	4xxx_IEEE	${KICAD8_SYMBOL_DIR}/4xxx_IEEE.kicad_sym	KiCad
☑	☑	74xGxx	${KICAD8_SYMBOL_DIR}/74xGxx.kicad_sym	KiCad
☑	☑	74xx	${KICAD8_SYMBOL_DIR}/74xx.kicad_sym	KiCad
☑	☑	74xx_IEEE	${KICAD8_SYMBOL_DIR}/74xx_IEEE.kicad_sym	KiCad
☑	☑	Amplifier_Audio	${KICAD8_SYMBOL_DIR}/Amplifier_Audio.kicad_sym	KiCad
☑	☑	Amplifier_Buffer	${KICAD8_SYMBOL_DIR}/Amplifier_Buffer.kicad_sym	KiCad
☑	☑	Amplifier_Current	${KICAD8_SYMBOL_DIR}/Amplifier_Current.kicad_sym	KiCad
☑	☑	Amplifier_Difference	${KICAD8_SYMBOL_DIR}/Amplifier_Difference.kicad_sym	KiCad
☑	☑	Amplifier_Operational	${KICAD8_SYMBOL_DIR}/Amplifier_Operational.kicad_sym	KiCad
☑	☑	Amplifier_Instrumentation	${KICAD8_SYMBOL_DIR}/Amplifier_Instrumentation.kicad_sym	KiCad
☑	☑	Amplifier_Video	${KICAD8_SYMBOL_DIR}/Amplifier_Video.kicad_sym	KiCad

Available path substitutions:

${KICAD10_SYMBOL_DIR}	/home/graham/kicad/libraries/symbols/kicad-symbols
${KIPRJMOD}	/home/graham/kicad/src/kicad/demos/openair-max

KiCad's symbol library management system allows directly using many types of symbol libraries, including formats that are native to other non-KiCad EDA tools:

- KiCad symbol libraries (`.kicad_sym` files)
- KiCad Legacy symbol libraries (`.lib` files)
- Altium Designer libraries (`.SchLib` or `.IntLib` files)
- CADSTAR Schematic Archive libraries (`.lib` files)
- KiCad database library configuration files (`.kicad_dbl` files)
- Eagle libraries (`.xml` files)
- EasyEDA (JLCEDA) Standard Edition libraries (`.json` files)
- EasyEDA (JLCEDA) Professional Edition libraries (`.elibz` , `.epro` , or `.zip` files)
- KiCad HTTP library configuration files (`.kicad_httplib` files)

Non-KiCad symbol libraries, including KiCad Legacy symbol libraries, can be migrated to KiCad `.kicad_sym` format using the **Migrate Libraries** button (see the migrating libraries section).

NOTE

KiCad only supports writing to KiCad's native `.kicad_sym` format symbol libraries. All other symbol library formats are read-only. To modify a non-KiCad format symbol 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 symbol library table contains a single entry, which points to another symbol library table containing all of KiCad's default symbol 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 symbol 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

Symbol libraries can only be used if they have been added to either the global or project-specific symbol 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 Symbol Editor or Symbol Chooser.

Libraries can be disabled by unchecking the **Enable** checkbox in the first column. Inactive 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. Alternatively, you can hide a library from the library browsers while still loading it from disk by unchecking the **Show** checkbox.

NOTE Hiding a library is primarily useful in conjunction with database libraries. Database libraries define symbols by adding metadata to symbol definitions from another symbol library. This base symbol library needs to be loaded, so it cannot be disabled, but for convenience the library can be hidden if it is not needed beyond the database library definition.

A range of libraries can be selected by clicking the first library in the range and then Shift-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 symbol names because it is used as a separator between nicknames and symbols.

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 (`.kicad_sym`) can be saved. Other symbol 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 symbol 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 symbol 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 symbol 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 symbol library tables allows libraries to be relocated without breaking the symbol 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, `${KICAD9_SYMBOL_DIR}` will automatically resolve to the value of `${KICAD10_SYMBOL_DIR}` if there is no `KICAD9_SYMBOL_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.

Usage patterns

Symbol libraries can be defined either globally or specifically to the currently loaded project. Symbol libraries defined in the user's global table are always available and are stored in the `sym-lib-table` file in the user's KiCad configuration folder. The project-specific symbol library table is active only for the currently open project file.

Ada keuntungan dan kerugian untuk masing-masing metode tersebut. Mendefinisikan keseluruhan pustaka di dalam tabel global berarti pustaka-pustaka tersebut akan selalu tersedia saat dibutuhkan. Namun kerugiannya adalah waktu yang dibutuhkan untuk memuat pustaka-pustaka tersebut akan bertambah.

Mendefinisikan keseluruhan pustaka simbol di dalam tabel proyek berarti bahwa Anda hanya memiliki pustaka yang dibutuhkan oleh proyek saja, sehingga akan mengurangi waktu yang dibutuhkan untuk memuat pustaka-pustaka tersebut. Namun kekurangannya adalah Anda harus selalu mengingat untuk menambahkan setiap pustaka simbol yang dibutuhkan untuk setiap proyek.

Maka pola pemakaian yang baik adalah mendefinisikan secara global pustaka-pustaka yang umum digunakan. Sedangkan untuk pustaka-pustaka yang spesifik yang hanya dibutuhkan oleh proyek, kita definisikan di dalam tabel pustaka proyek. Tidak ada batasan bagi suatu pustaka untuk didefinisikan ke tabel manapun.

Migrating symbol libraries to KiCad format

Non-KiCad format libraries, including legacy libraries (`.lib` files), are read-only. They need to be converted to KiCad format (`.kicad_sym` files) 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 symbol 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 Symbol Editor and saving them as a new library.

Legacy project remapping

When loading a schematic created prior to the symbol library table implementation, KiCad will attempt to remap the symbol library links in the schematic to the appropriate library table symbols. The success of this process is dependent on several factors:

- pustaka asli yang digunakan di skematik masih tersedia dan belum berubah dari saat simbol ditambahkan ke dalam skematik.
- semua operasi penyelamatan telah dilakukan ketika terdeteksi untuk membuat sebuah pustaka penyelamat atau menjaga agar pustaka penyelamat yang ada tetap dalam kondisi terkini.
- integritas pustaka tembolok simbol proyek tidak dalam kondisi rusak.

WARNING

Proses pemetaan ulang akan membuat cadangan untuk keseluruhan berkas yang berubah selama proses pemetaan, di dalam folder 'rescue-backup' di folder proyek. Selalu buat cadangan secara manual untuk proyek Anda sebelum melakukan pemetaan ulang, untuk berjaga-jaga jika pemetaan tidak berjalan sebagaimana mestinya.

WARNING

Operasi penyelamatan dilakukan walaupun telah dinonaktifkan, untuk memastikan bahwa simbol-simbol yang tepat telah tersedia untuk dipetakan ulang. Jangan membatalkan operasi ini atau pemetaan ulang akan gagal untuk memetakan simbol-simbol skematik secara benar. Setiap tautan simbol yang rusak harus diperbaiki secara manual.

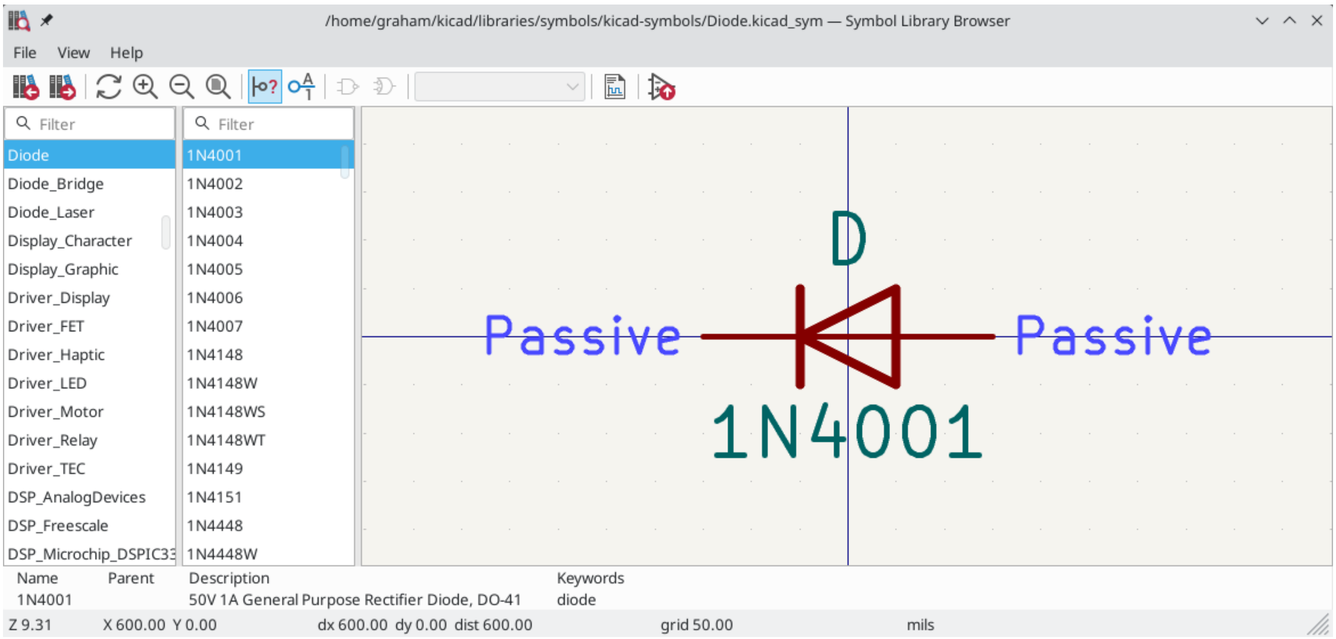
NOTE

If the original libraries have been removed and the rescue was not performed, the cache library can be used as a recovery library as a last resort. Copy the cache library to a new file name and add the new library file to the top of the library list using a version of KiCad prior to the symbol library table implementation.

Browsing symbol libraries

The Symbol Library Browser allows you to quickly examine the contents of symbol libraries. The Symbol Library Viewer can be accessed by clicking icon on the main Symbol Editor toolbar or with **View → Symbol Library Browser**.

To examine the contents of a library, select a library from the list in the left hand panel. All symbols in the selected library will appear in the second panel. Select a symbol name to view the symbol.


/home/graham/kicad/libraries/symbols/kicad-symbols/Diode.kicad_sym — Symbol Library Browser
File
View
Help
Filter
Filter
D
Passive
Passive
1N4001

Diode
Diode_Bridge
Diode_Laser
Display_Character
Display_Graphic
Driver_Display
Driver_FET
Driver_Haptic
Driver_LED
Driver_Motor
Driver_Relay
Driver_TEC
DSP_AnalogDevices
DSP_Freescale
DSP_Microchip_DSPIC33

1N4001
1N4002
1N4003
1N4004
1N4005
1N4006
1N4007
1N4148
1N4148W
1N4148WS
1N4148WT
1N4149
1N4151
1N4448
1N4448W

Name	Parent	Description	Keywords
1N4001		50V 1A General Purpose Rectifier Diode, DO-41	diode

Z 9.31 X 600.00 Y 0.00 dx 600.00 dy 0.00 dist 600.00 grid 50.00 mils

Double clicking the name of a symbol or using the button adds the symbol to the schematic.

The top toolbar contains the following commands:

	Select previous symbol in library.
	Select next symbol in library.
	Zoom tools.
?	Toggle display of pin electrical types.
A 1	Toggle display of pin numbers.
Standard	Select the body style of a symbol with multiple body styles.
Unit A	Select the unit of a multi-unit symbol.
	Open the symbol's datasheet, if it is defined.
	Insert current symbol into the schematic.

Creating and editing symbols

A symbol is a schematic representation of a component. A symbol is composed of:

- Graphical items (lines, circles, arcs, text, etc.) that determine how symbol looks in a schematic.
- Pins, which connect to wires in the schematic and correspond to footprint pads in the PCB. Pins have both graphic properties (line, clock, inverted, low level active, etc.) and electrical properties (input, output, bidirectional, etc.) used by the Electrical Rules Check (ERC) tool.
- Fields, such as references, values, corresponding footprint names for PCB design, etc.

A symbol library is composed of one or more symbols. Generally the symbols are logically grouped by function, type, and/or manufacturer. Each symbol library is a single file with the `.kicad_sym` extension.

Symbols can be derived from another symbol in the same library. Derived symbols share the base symbol's graphical shape and pin definitions, but can override the base symbol's property fields (value, footprint, footprint filters, datasheet, description, etc.). Derived symbols can be used to define symbols that are similar to a base part. For example, 74LS00, 74HC00, and 7437 symbols could all be derived from a 7400 symbol. In previous versions of KiCad, derived symbols were referred to as aliases.

Symbol Editor overview

KiCad provides a symbol editing tool that allows you to create libraries; add, edit, delete, or transfer symbols between libraries; export symbols to files; and import symbols from files. The Symbol Editor can be launched from the KiCad Project Manager or from the Schematic Editor (**Tools** → **Symbol Editor**). You can also open the Symbol Editor from the a symbol in the schematic; in this way you can edit either the library copy or the schematic copy of that symbol in the editor.

NOTE Editing the library version of a symbol will not affect any copies of that symbol that have been added to a schematic until the schematic copy is updated from the library. Conversely, editing the schematic version of a symbol will not affect the library version of a symbol or any other copies of that symbol in a schematic.

In general, the flow for designing a symbol involves:

- Menentukan apakah simbol perlu dibuat dalam bentuk lebih dari satu bagian.
- Defining if the symbol has an alternate body style (also known as a De Morgan representation).
- Membuat desain representasi simbolis menggunakan garis, persegi, lingkaran, poligon, dan teks.
- Adding pins by carefully defining each pin's graphical elements, name, number, and electrical property (input, output, tri-state, power output, etc.).
- Determining if the symbol should be derived from another symbol with the same graphical design and pin definition.
- Menambahkan atribut opsional seperti nama *footprint* yang digunakan oleh perangkat lunak desain PCB, dan/atau menentukan bagian-bagian yang perlu ditampilkan.
- Mendokumentasikan simbol dengan menambahkan deskripsi dan tautan ke lembar data, dsb.

Menyimpan ke dalam pustaka yang diinginkan.

The Symbol Editor main window is shown below. It has three toolbars for quick access to common features and a symbol 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 Symbol Editor Preferences.

In addition to the toolbars, there are collapsible panels for the symbol tree, Properties Manager, and selection filter on the left. The bottom of the window contains a message panel that shows details about the selected object.

Symbol Editor main window

Top toolbar

The main toolbar is at the top of the main window. It has buttons for the undo/redo commands, zoom commands, symbol properties dialogs, and unit/representation management controls.

	Create a new symbol in the selected library.
	Save the currently selected library. All modified symbols in the library will be saved.
↶	Undo last edit.
↷	Redo last undo.
⟳	Refresh display.
	Zoom in.
	Zoom out.
	Zoom to fit symbol in display.
	Zoom to fit selection.
	Rotate counter-clockwise.
	Rotate clockwise.
	Mirror horizontally.
	Mirror vertically.
	Edit the current symbol's properties.
	Edit the symbol's pins in a tabular interface.
	Open the symbol's datasheet, if it is defined.
	Run the symbol checker to test the current symbol for design errors.
Standard ⌄	Select the body style to display. The drop down control will be disabled if the current symbol does not have multiple body styles.
Unit A ⌄	Select the unit of a multi-unit symbol to display. The drop down control will be disabled if the current symbol does not have multiple units.
	Enable synchronized pin edit mode. When this mode is enabled, any pin modifications are propagated to all other symbol units. Pin number changes are not propagated. This mode is automatically enabled for symbols with multiple interchangeable units and cannot be enabled for symbols with only one unit.
	Insert current symbol into the schematic.

Left toolbar display controls

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

⠿	Toggle grid visibility on and off.
⠿	Toggle item-specific grid overrides on and off.
in mil mm	Set units to inches, mils (0.001 inch), 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).
?	Toggle display of pin electrical types.
▷	Toggle display of hidden (invisible) pins.
T	Toggle display of hidden (invisible) fields.
☰	Toggle display of library and symbol tree.
✕	Toggle display of Properties Manager panel.

Right toolbar tools

Placement and drawing tools are located in the right toolbar.

↖	Select tool. Right-clicking with the select tool opens the context menu for the object under the cursor. Left-clicking with the select tool displays the attributes of the object under the cursor in the message panel at the bottom of the main window. Double-left-clicking with the select tool will open the properties dialog for the object under the cursor.
A	Pin tool. Left-click to add a new pin.
T	Graphical text tool. Left-click to add a new graphical text item.
▤	Graphical textbox tool. Left-click to add a new graphical textbox item.
□	Rectangle tool. Left-click to begin drawing the first corner of a graphical rectangle. Left-click again to place the opposite corner of the rectangle.
○	Circle tool. Left-click to begin drawing a new graphical circle from the center. Left-click again to define the radius of the circle.
◜	Arc tool. Left-click to begin drawing a new graphical arc item from the first arc end point. Left-click again to define the second arc end point. Adjust the radius by dragging the arc center point.
	Bezier curve tool. Left-click to begin drawing a new graphical bezier curve item. First click for the start point, then for the control points and the end point. Adjust the curve by dragging the points.
╱	Connected line tool. Left-click to begin drawing a new graphical line item in the current symbol. Left-click for each additional connected line. Double-left-click to complete the line.
	Connected line tool. Left-click to begin drawing a new graphical line item in the current symbol. Left-click for each additional connected line. Double-left-click to complete the line.
(0,0)	Anchor tool. Left-click to set the anchor position of the symbol.
⊗	Delete tool. Left-click to delete an object from the current symbol.

Browsing, modifying, and saving symbols

The button displays or hides the list of available libraries, which allows you to select an active library. When a new symbol is created, it will be placed in the active library.

Clicking on a symbol name opens that symbol in the editor, and hovering the cursor over the name of a symbol displays a preview of the symbol.

NOTE

Some symbols are derived from other symbols. Derived symbol names are displayed in *italics* in the treeview. If a derived symbol is opened, its symbol graphics will not be editable. Its symbol fields will be editable as normal. To edit the graphics of a base symbol and all of its derived symbols, open the base symbol.

After modification, a symbol can be saved in the current library or a different library. To save the modified symbol in the current library, click the 💾 icon.

> **NOTE** Saving a modified symbol also saves all other modified symbols in the same library.

To save the symbol changes to a new symbol, click **File** → **Save As…**. The symbol can be saved in the current library or a different library (including a new library), and a new name can be set for the symbol. Alternatively, you can use **File** → **Save Copy As…**, which behaves the same as **Save As** except that the original symbol remains open rather than switching to the new symbol.

To create a new file containing only the current symbol, click **File** → **Export** → **Symbol…**. This file will be a standard symbol library file which will contain only one symbol. The library will not be added to your library table.

The editor can also open symbols from the schematic. To edit a symbol from the schematic, right click a symbol in the Schematic Editor and select **Open in Schematic Editor** (`Ctrl` + `E`).

Editing and saving the schematic copy of a symbol will only update that symbol in the schematic; it will not update other copies of that symbol in the schematic, and it will not change the original library copy of the symbol. When you open the schematic copy of a symbol, the Schematic 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 symbol instead, or press `Ctrl` + `Shift` + `E`.

Creating a new symbol library

You can create a new symbol library by clicking **File** → **New Library…**. This opens a file browser for you to choose a location and name for the new library. Symbol libraries are saved as a file; the file name is the library name plus the suffix `.kicad_sym`.

The file browser also lets you choose whether the new library should be added to the global symbol library table or the project symbol 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 symbol library tables are managed using **Preferences** → **Manage Symbol Libraries…**. This includes deleting and renaming symbol libraries. For more information about managing library tables, see the symbol library table documentation.

After choosing a name, location, and library table, a new, empty library is created. When you create new symbols, you can save them in this library.

Creating a new symbol

To create a new symbol in the current symbol library, click the button. You will be asked for a number of symbol properties.

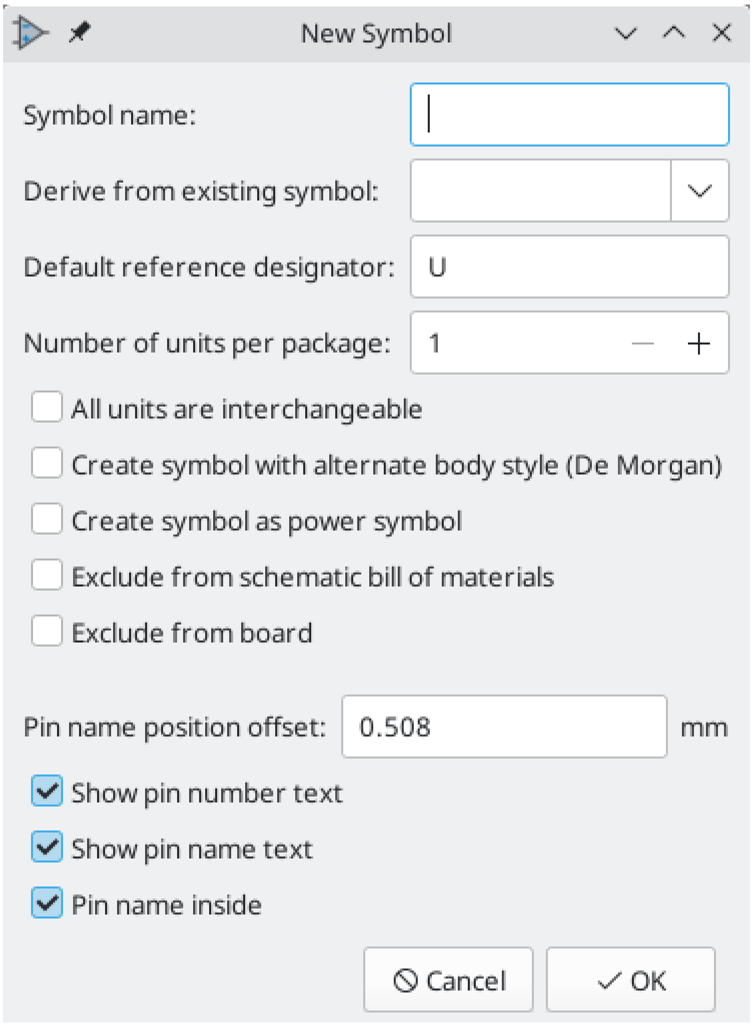
- A symbol name
- An optional base symbol to derive the new symbol from. The new symbol will use the base symbol's graphical shape and pin configuration, but other symbol information can be modified in the derived symbol. The base symbol must be in the same library as the new derived symbol.
- The reference designator prefix (`U`, `C`, `R`…).
- The number of units per package, and whether those units are interchangeable (for example a 7400 quad NAND symbol could have 4 units, one for each gate).

- If an alternate body style (sometimes referred to as a "De Morgan equivalent") is desired.
- Whether the symbol is a power symbol. Power symbols appear in the **Add Power Symbol** dialog in the Schematic editor, make global net connections based on their value, cannot be assigned a footprint, and are excluded from the PCB and bill of materials.
- Whether the symbol should be excluded from the bill of materials.
- Whether the symbol should be excluded from the PCB.

There are also several graphical options.

- The offset between the end of each pin and its pin name.
- Whether the pin number and pin name should be displayed.
- Whether the pin names should be displayed alongside the pins or at the ends of the pins inside the symbol body.

These properties can also be changed later in the Symbol Properties window.



Symbol name:

Derive from existing symbol:

Default reference designator: U

Number of units per package: 1

□ All units are interchangeable

□ Create symbol with alternate body style (De Morgan)

□ Create symbol as power symbol

□ Exclude from schematic bill of materials

□ Exclude from board

Pin name position offset: 0.508 mm

☑ Show pin number text

☑ Show pin name text

☑ Pin name inside

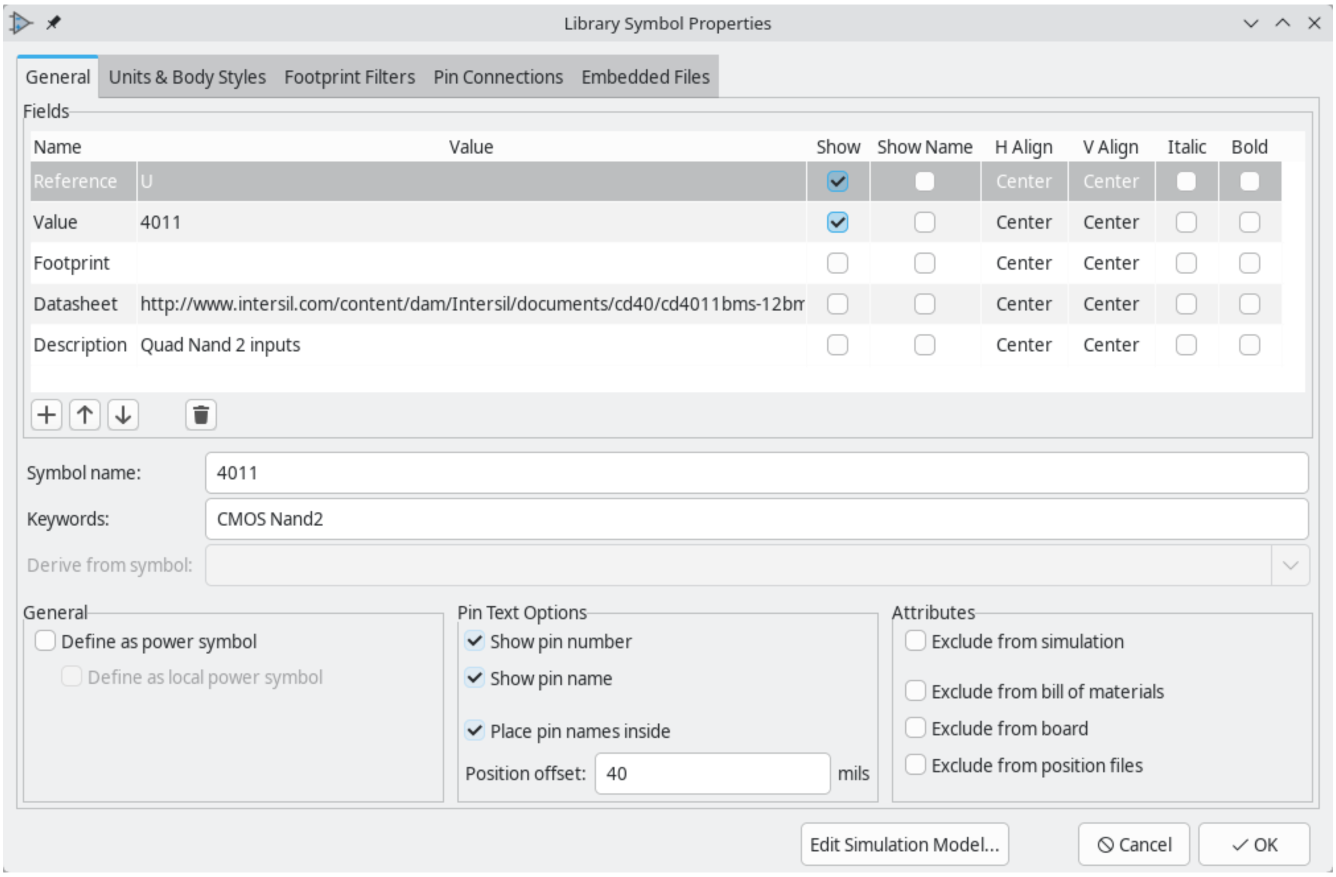
A new symbol will be created in the selected library using the given properties. The new symbol is opened in the editor.

The blue cross in the center is the symbol anchor, which specifies the symbol origin i.e. the coordinates (0, 0). The anchor can be repositioned by selecting the button and clicking on the new desired anchor position.

Editing symbol properties

Symbols 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), pin connection settings, and filters for appropriate footprints. These are initially defined in the library copy of the symbol, but they can be modified on a per-instance basis once a symbol is added to a schematic. In other words, two copies of the same symbol in a single schematic can have their properties edited separately.

To change the symbol properties, click on the button to show the Symbol Properties dialog. You can also double click an empty spot in the editing canvas.



Fields

Name	Value	Show	Show Name	H Align	V Align	Italic	Bold
Reference	U	☑	□	Center	Center	□	□
Value	4011	☑	□	Center	Center	□	□
Footprint		□	□	Center	Center	□	□
Datasheet	http://www.intersil.com/content/dam/Intersil/documents/cd40/cd4011bms-12bm	□	□	Center	Center	□	□
Description	Quad Nand 2 inputs	□	□	Center	Center	□	□

Symbol name: 4011

Keywords: CMOS Nand2

Derive from symbol:

General
- □ Define as power symbol
- □ Define as local power symbol

Pin Text Options
- ☑ Show pin number
- ☑ Show pin name
- ☑ Place pin names inside
- Position offset: 40 mils

Attributes
- □ Exclude from simulation
- □ Exclude from bill of materials
- □ Exclude from board
- □ Exclude from position files

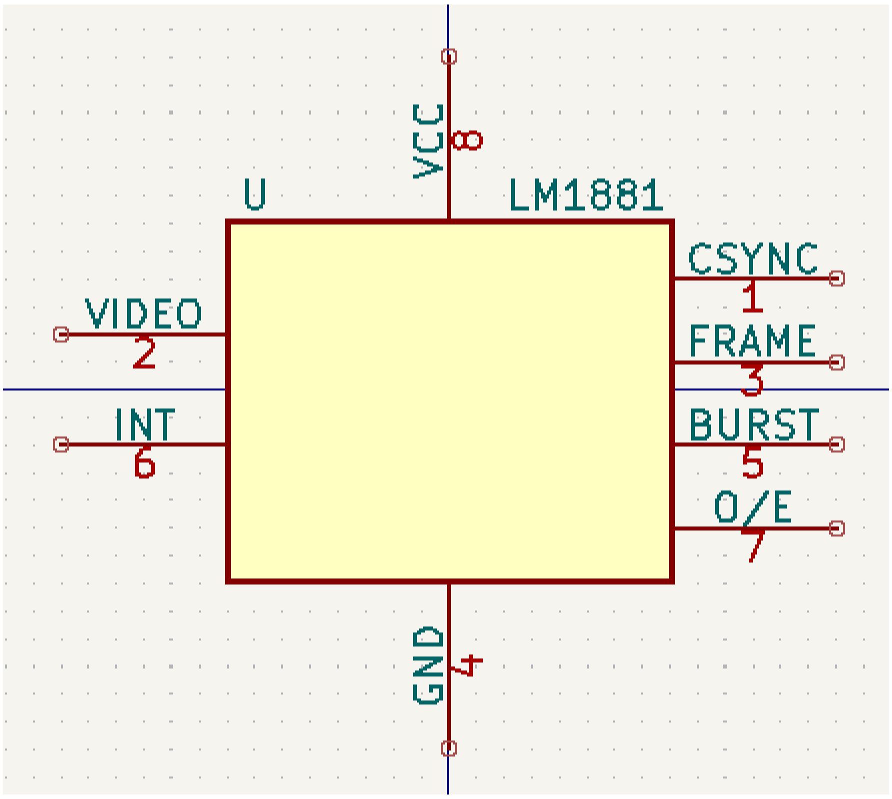
The most important settings are in the **General** tab. Here you can edit symbol fields as well as many basic properties for the symbol. Other tabs contain options for multiple units and body styles, footprint filters, jumper pins, and embedded files.

Symbol fields are explained in a separate section. The other settings in the general tab are:

- **Symbol name** is the symbol's name in the library. Symbols are identified by a combination of the library and symbol name. In older versions of KiCad, the symbol name was linked to the `Value` field. This link is removed in KiCad 7.0 and later.
- The **keywords** should contain additional terms related to the component. Keywords are primarily used, in combination with the symbol name and the `Description` field, for searching for the symbol in the Symbol Chooser and the Symbol Editor. Those three items are also displayed when you select a symbol in the Symbol Chooser.
- **Define as power symbol** and **Define as local power symbol** control whether the symbol acts as a power symbol, making connections between otherwise separate nets.

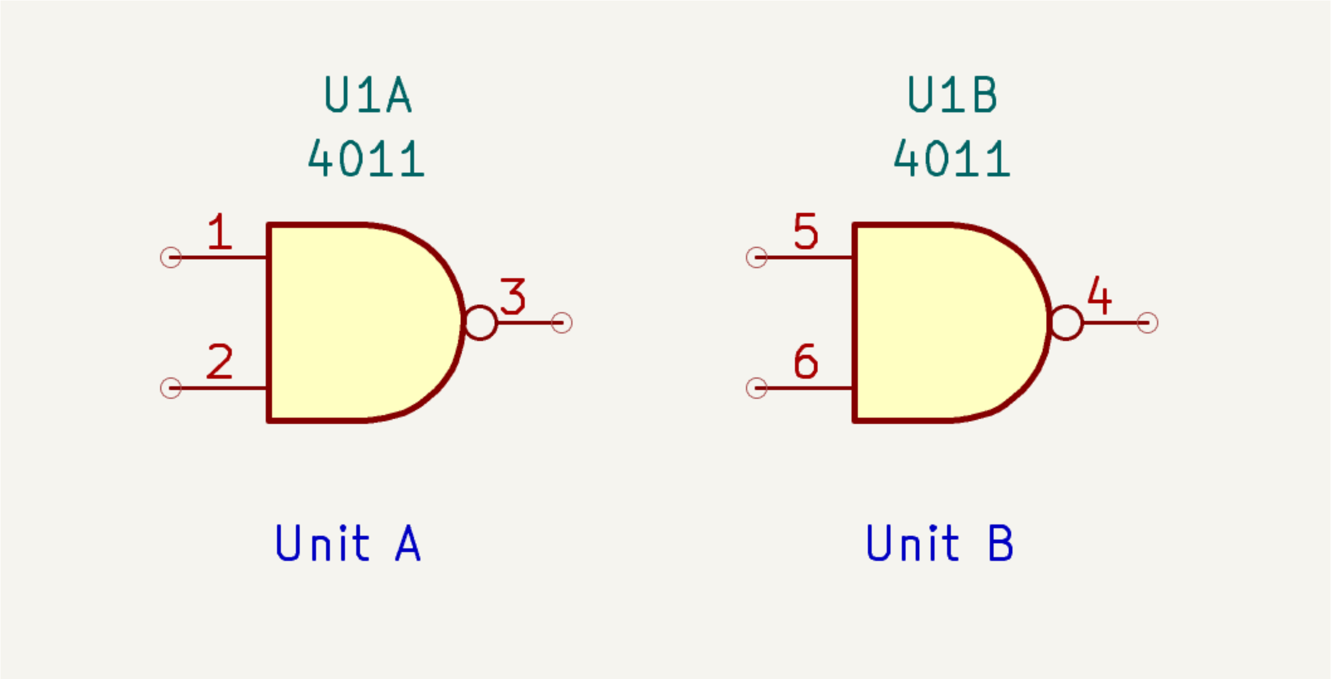
- The graphic options **Show pin number** and **Show pin name** control the visibility of the pin number and pin name text.
- **Place pin names inside** controls the pin name position relative to the pin body. The pin names will be displayed inside the symbol outline if the option is checked. In this case the **Pin Name Position Offset** property defines the shift of the text away from the body end of the pin. A value from `0.02` to `0.05` inches is usually reasonable.
- **Exclude from simulation**, **Exclude from bill of materials**, **Exclude from board**, and **Exclude from position files** set the default state of these symbol attributes when the symbol is added to the schematic.

The example below shows a symbol with the **Place pin name inside** option unchecked. Notice the position of the names and pin numbers.



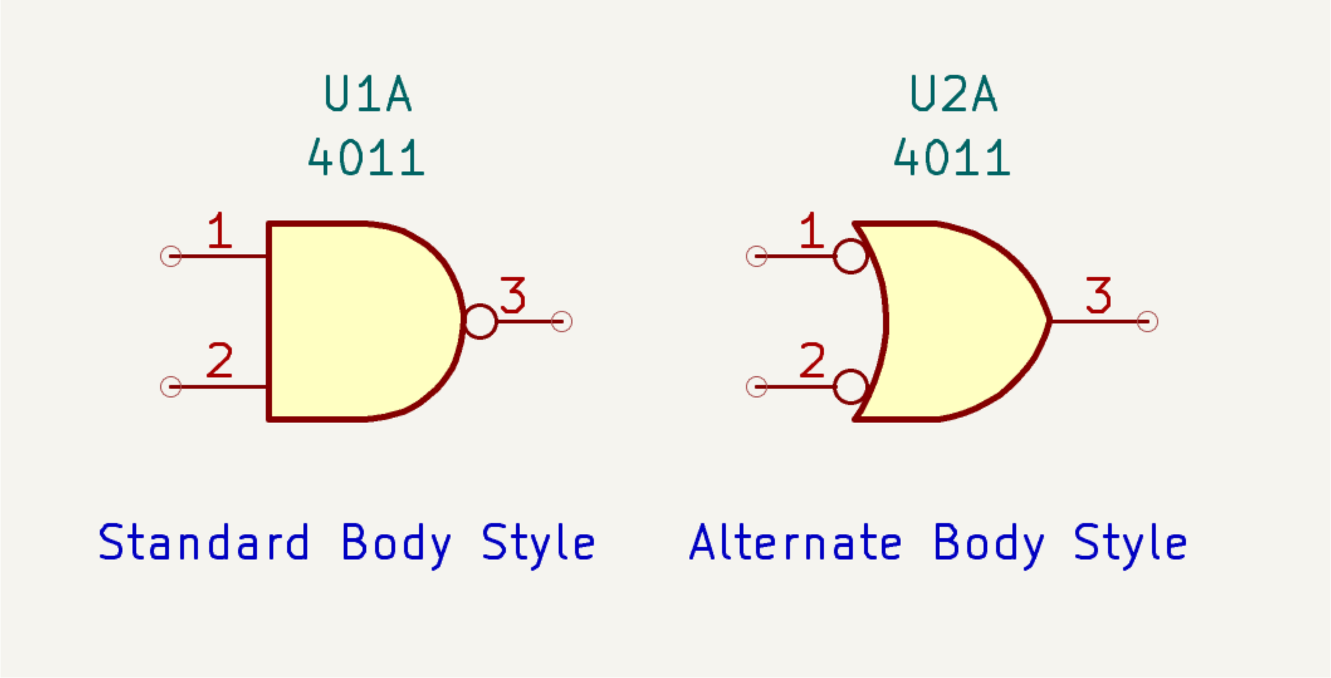
Symbol units and alternate body styles

Symbols can have more than one unit per package, each with different graphics and pin configurations. This is often used for logic gates, opamps, or other components that have multiple subunits within one physical package. Each symbol unit can be placed separately in the schematic, which can make it easier to draw neat schematics.



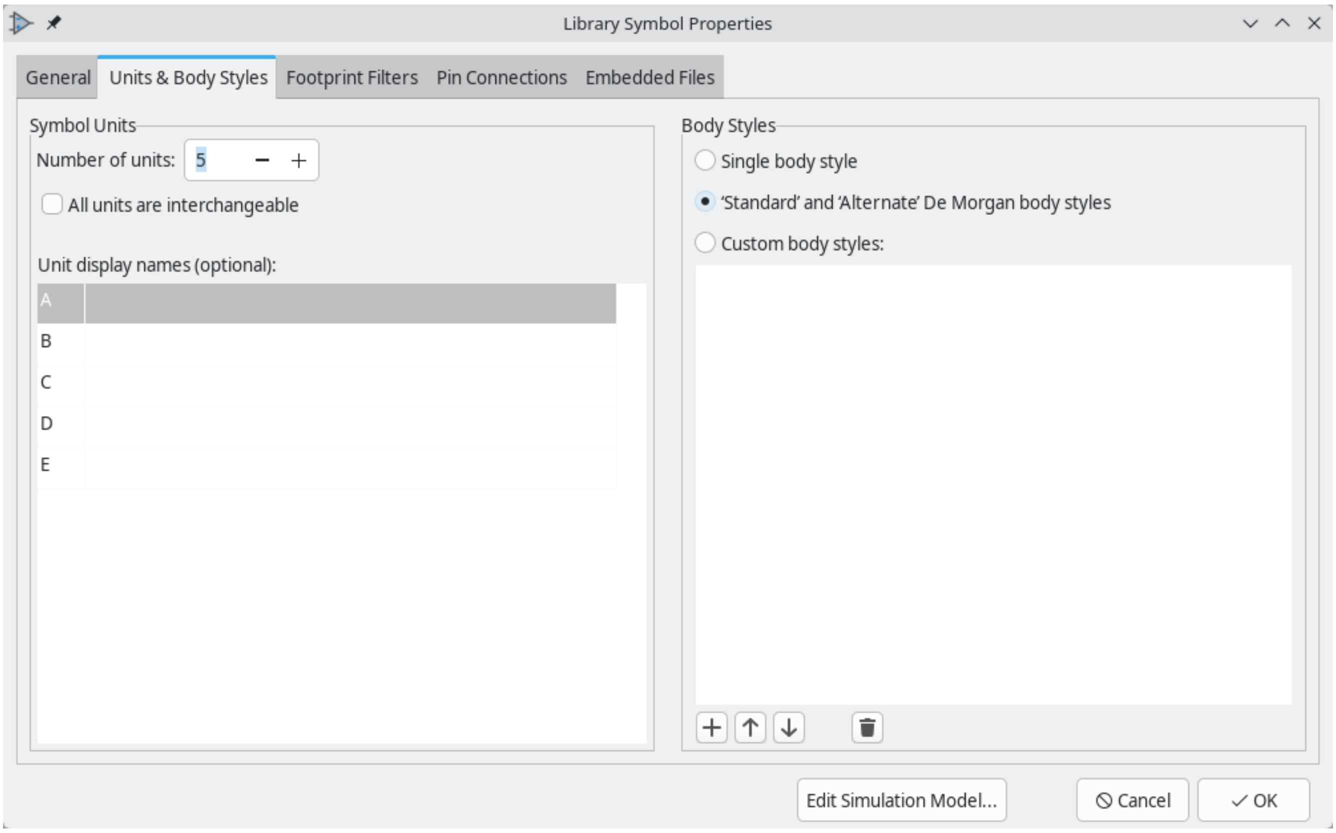
Symbols can also have multiple body styles, which are alternate graphical representations of the same symbol. For example, the symbol for a NAND gate could have one body style showing an ANSI-style AND gate with inverting "bubble" on the output, another body style drawn as an OR gate with inverting "bubbles" on the inputs, and a third body style with an IEC-style rectangular NAND gate symbol.

NOTE Before KiCad 10.0, a maximum of one alternate body style per symbol was possible, and a symbol's alternate body style was sometimes referred to as a "De Morgan equivalent". Beginning in KiCad 10.0, multiple alternate body styles are possible.



By default, pins are part of a single unit and body style in the symbol. Alternatively, pins can be set as common to all units or common to all body styles. When a pin is common to all units or all body styles, it only needs to be created once, no matter how many units or body styles are used. Graphics and text may also be set as common to all units or body styles, but typically graphics are specific to a single body style so that each body style can have a unique appearance.

The settings for adjusting a symbol's units and body styles are in the **Units & Body Styles** tab of the Symbol Properties dialog.



Library Symbol Properties

General | Units & Body Styles | Footprint Filters | Pin Connections | Embedded Files

Symbol Units

Number of units: 5 − +

□ All units are interchangeable

Unit display names (optional):

A	
B	
C	
D	
E	

Body Styles

○ Single body style

● 'Standard' and 'Alternate' De Morgan body styles

○ Custom body styles:

Edit Simulation Model... | Cancel | OK

- **Number of Units** controls the number of symbol units.
- **All units are interchangeable** should be set if all symbol units are equivalent. If all units are interchangeable, you can use the Synchronized Pin Edits mode to quickly make changes to all units.
- The **Unit Display Names** table lets you set an optional name for each unit. If you do not configure display names, symbol units are named `Unit A`, `Unit B`, etc.
- The **Body Styles** options let you select between a single body style, two body styles for traditional "Standard" and "De Morgan" representations of a symbol, and an arbitrary number of custom body styles. If you enable custom body styles, you can set the name for each style.

You can only view and edit one symbol unit and body style at a time in the symbol editor's editing canvas. Use the Unit A ⌄ unit selection dropdown to select the unit you wish to edit, and use the Standard ⌄ body style selection dropdown to select a body style.

NOTE

Synchronized Pins Edit Mode can be enabled by clicking the icon. In this mode, pin modifications are propagated between symbol units; changes made in one unit will be reflected in the other units as well. When this mode is disabled, pin changes made in one unit do not affect other units. This mode is enabled automatically when **All units are interchangeable** is checked, but it can be disabled. The mode cannot be enabled when **All units are interchangeable** is unchecked or when the symbol only has one unit.

Symbol graphics

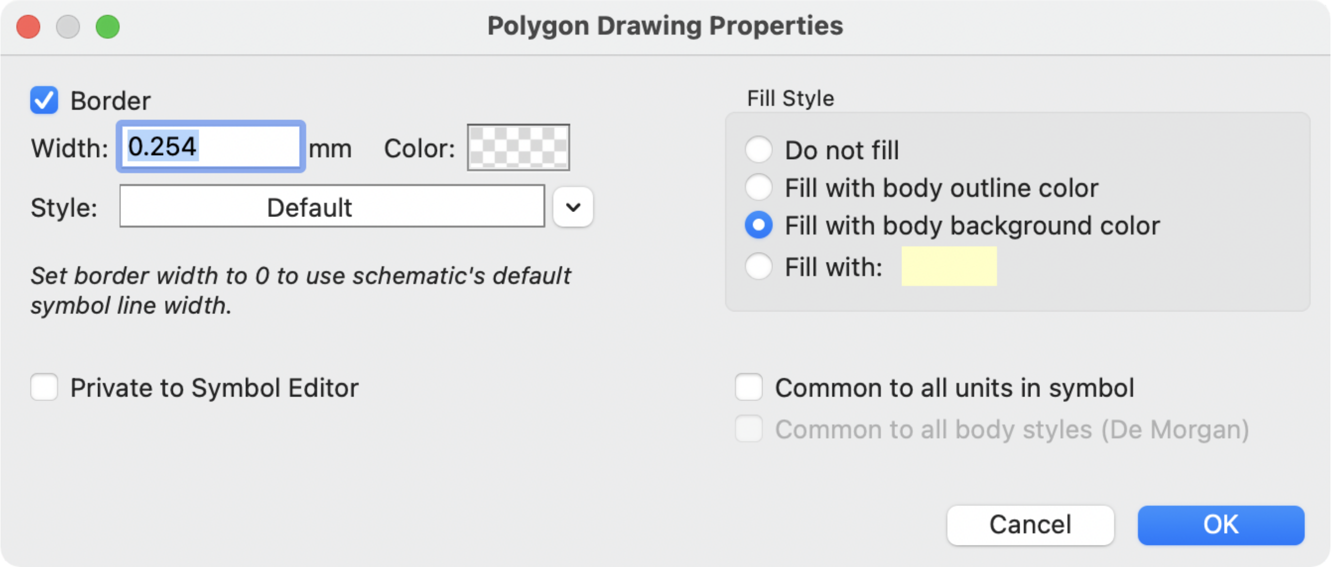
Graphical elements create the visual representation of a symbol and contain no electrical connection information. You can draw new graphic shapes using the buttons on the right toolbar. The following types of objects are available:

- Lines (╱) and polygons (⊳) defined by start and end points.
- Rectangles (□) defined by two diagonal corners.
- Circles (○) defined by the center and radius.
- Arcs (◜) defined by the starting and ending point of the arc and its center.
- Graphical text (T) and textboxes (☰), which is automatically oriented to be readable, even when the symbol is mirrored. Note that graphic text items are not the same as symbol fields.

Each graphic item (line, arc, circle, etc.) can be defined as common to all units and/or body styles or specific to a given unit and/or body style.

Element options can be quickly accessed by right-clicking on the element to display the context menu for the selected element. You can also double-left-click on an element to modify its properties, or edit its properties using the Properties Manager panel.

Below is the properties dialog for a polygon element.



Polygon Drawing Properties

☑ Border

Width: 0.254 mm Color:

Style: Default

Set border width to 0 to use schematic's default symbol line width.

Fill Style

○ Do not fill
○ Fill with body outline color
● Fill with body background color
○ Fill with:

□ Private to Symbol Editor

□ Common to all units in symbol

□ Common to all body styles (De Morgan)

Properti dari suatu elemen grafis antara lain:

- **Border** determines whether the the shape’s outline should be drawn.
- **Width** and **color** define the line width and color of the border. A border width of `0` uses the schematic’s default symbol line width. **Style** determines the line style of the border (solid, dashed, dotted, etc.).
- **Fill Style** determines if the shape defined by the graphical element is to be drawn unfilled or filled. The fill color can be the color theme’s body outline color, body background color, or a custom color.
- **Common to all units in symbol** determines if the graphical element is drawn for each unit in symbol with more than one unit per package or if the graphical element is only drawn for the current unit.

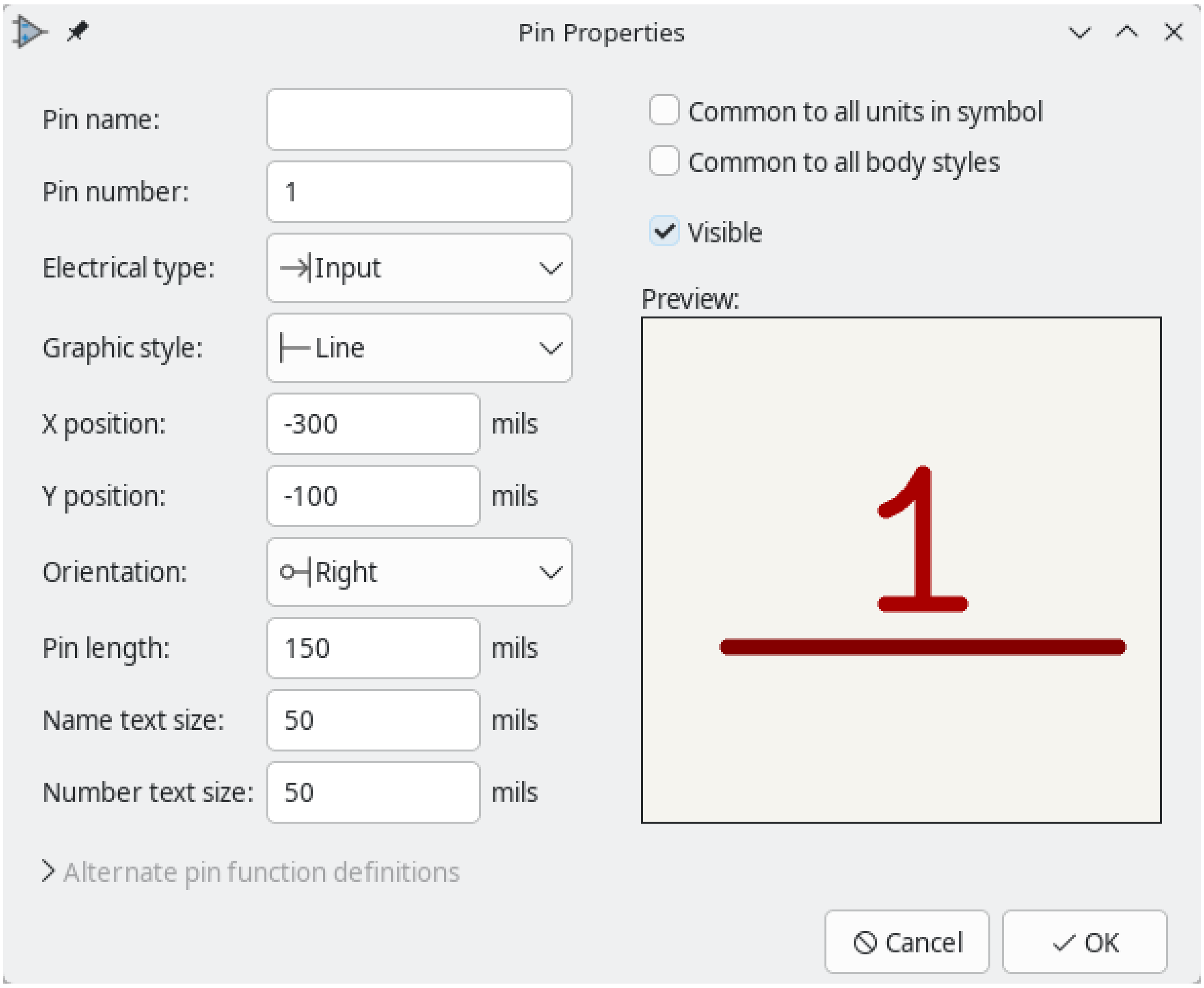
- **Common to all body styles** determines if the graphical element is drawn for each symbolic representation in symbols with an alternate body style or if the graphical element is only drawn for the current body style.
- **Private to Symbol Editor** causes the shape to be visible only when the symbol is edited in the Symbol Editor. The shape will be hidden when the symbol is added to a schematic.

Symbol pins

Symbol pins are the parts of a symbol that connect to wires in the schematic. Each symbol pin corresponds to a footprint pad in the PCB, and pins are matched to pads by number, so each pin's number must match the corresponding pad number in the footprint.

You can create and insert a pin by clicking on the button. This opens the Pin Properties dialog, then allows you to place the pin in the symbol. After placing a pin, you can edit its properties by reopening its Properties dialog (E or double click on the pin) or by using the Properties Manager.

NOTE You can edit all of a symbol's pins at once using the Pin Table. This may be a more convenient way to edit pins, especially for symbols with many pins.


Pin Properties

Pin name:

Pin number: 1

Electrical type: Input

Graphic style: Line

X position: -300 mils

Y position: -100 mils

Orientation: Right

Pin length: 150 mils

Name text size: 50 mils

Number text size: 50 mils

□ Common to all units in symbol
□ Common to all body styles
☑ Visible

Preview:

Pins have the following properties:

- **Pin name** is an optional name for the pin. Pin names can be displayed as part of the symbol, and if present they are used as part of automatically generated net names for unnamed nets attached to the pin.

Pin names cannot contain spaces; any spaces in a pin name are automatically replaced with underscores. To define a pin name with an inverted signal (overbar) use the `~` (tilde) character followed by the text to invert in braces. For example, `~{FO}O` would display $\overline{\text{FO}}$ O.

- **Pin number** is a string that uniquely identifies the pin. Symbol pins are matched to footprint pads by pin/pad number, so pin numbers should correspond to pad numbers. Pin numbers are typically unique within a symbol. In addition to numbers, pin numbers can contain letters or symbols. Pin numbers cannot contain spaces; any spaces in a pin number are automatically replaced with underscores.

- **Electrical type** describes how the pin behaves electrically, such as input, output, or open collector. Electrical types affect ERC and some other behavior.

- **Graphic style** controls the appearance of the pin, such as an inverted pin or a clock input, but does not have any other effects.

- **X position** and **Y position** are the location of the pin relative to the symbol's anchor.

- **Orientation** is the direction the pin extends (left, right, up, or down) relative to its wire attachment point.

- **Pin length** is the length of the pin.

- **Name text size** and **Number text size** are the size that the pin name and number, respectively, are displayed at.

- **Common to all units in symbol** controls whether the pin is part of all symbol units or only the currently active symbol unit.

- **Common to all body styles** controls whether the pin is shown in all body styles or only the currently active body style.

- **Visible** controls whether the pin is shown or hidden. Note that hidden power input pins have special behavior.

- **Alternate pin definitions** are optional additional combinations of pin name, electrical type, and graphic style that can be selected in the schematic instead of the default pin definition. These can be used, for example, to represent selectable pin functions in a microcontroller.

Pin electrical types

Each pin in a symbol has an electrical type, such as input, output, or tri-state.

Choosing the correct electrical type is important for the schematic ERC tool. ERC will check that pins are connected appropriately, for example ensuring that input pins are driven and power inputs receive power from an appropriate source.

You can use the **Pin Conflicts Map** in the schematic editor to configure which pin types are allowed to connect and which will conflict. The default Pin Conflicts settings are briefly explained below. For more information, see the ERC documentation.

Additionally, some pin types have special behavior outside of ERC. In the router, pads corresponding to a **free** pin can be connected to copper of any other net without causing a DRC error, and multiple pads corresponding to a single **unconnected** pin do not need to be connected to each other in the board.

NOTE

The pin type that produces the optimal ERC pin conflict checking behavior is not always the same as the pin's conceptual pin type. When selecting a pin type, you should consider how that type will interact with the pin type of other connected pins and whether that will result in the desired ERC behavior. An example is an analog control pin that generates a current and senses the voltage generated by that current flowing through an external resistor. This pin could be considered an input pin because it senses a voltage provided externally. However, in a schematic this pin will be connected to a resistor pin (passive) and not to an output pin. There shouldn't be an ERC violation if the pin isn't connected to an output pin; in fact, there should be an ERC violation if the pin *does* connect to another output pin, as the pin would be sourcing a current on a net that is already driven. Therefore such a pin should have the Output pin type even though it is sensing a voltage and could be considered an input.

Pin Type	Description
Input	A pin which is exclusively an input. The default Pin Conflicts settings allow input pins to connect to most other types of pin. Also, an ERC violation will be produced if an input pin is not driven, i.e. it is not connected to a pin with type output, bidirectional, tristate, power output, or passive.
Output	A pin which is exclusively an output. The default Pin Conflicts settings allow output pins to connect to most types of pin that aren't also outputs.
Bidirectional	A pin that can be either an input or an output, such as a microcontroller data bus pin. The default Pin Conflicts settings allow bidirectional pins to connect to most other types of pins, though there are a few more restrictions than with input pins.
Tri-state	A three state output pin (high, low, or high impedance). The default Pin Conflicts settings allow tri-state pins to connect to most other types of pins, but warnings are generated when they are connected to most types of output or power pins.
Passive	A pin that is not connected to active electronics, for example pins on a resistor or connector. The default Pin Conflicts settings allow passive pins to connect to most other types of pin.
Free	A pin that does not electrically affect the operation of the device. These pins typically represent package leads that are not internally connected to the chip. The default Pin Conflicts settings allow free pins to connect to most other types of pin. In the PCB editor, pads corresponding to free pins can be connected to copper of any other net without causing a DRC error.
Unspecified	A pin which has an unspecified type. With the default Pin Conflicts settings, ERC generates warnings when unspecified pins are connected to most other types of pins.

Power input	A pin that powers the device. The default Pin Conflicts settings allow power input pins to connect to most other pin types. However, power input pins that are not connected to a power output pin generate an ERC violation. Additionally, power input pins that are marked invisible are automatically connected to the net with the same name as the pin. This behavior is supported primarily for legacy projects and is not recommended for new designs. See the Hidden Power Pin section for more information.
Power output	A pin that provides power to other pins, such as a regulator output. The default Pin Conflicts settings allow power output pins to connect to most types of input pins, but not output pins.
Open collector	An open collector logic output. The default Pin Conflicts settings allow open collector pins to connect to most input pins and other open collector pins, but not to most other types of outputs.
Open emitter	An open emitter logic output. The default Pin Conflicts settings allow open collector pins to connect to most input pins and other open emitter pins, but not to most other types of outputs.
Unconnected	A pin that should not be connected to anything. ERC does not allow pins of type unconnected to connect to any other type of pin, and ERC will not generate an "unconnected pin" violation when pins of this type are left unconnected. Unconnected pins are not configurable in the ERC Pin Conflicts map. If a footprint has multiple pads corresponding to a single unconnected pin, the pads do not need to be connected to each other in the board. When multiple pins of type unconnected are stacked in a symbol, they are connected to separate nets, whereas stacked pins of other types are connected to the same net. Note that this pin type is different than placing a no connect flag on a pin in the schematic. The unconnected pin type indicates that the pin should never be connected in any schematic, while a no connect flag indicates that the pin is intentionally unconnected in the current schematic.

Pin stacks

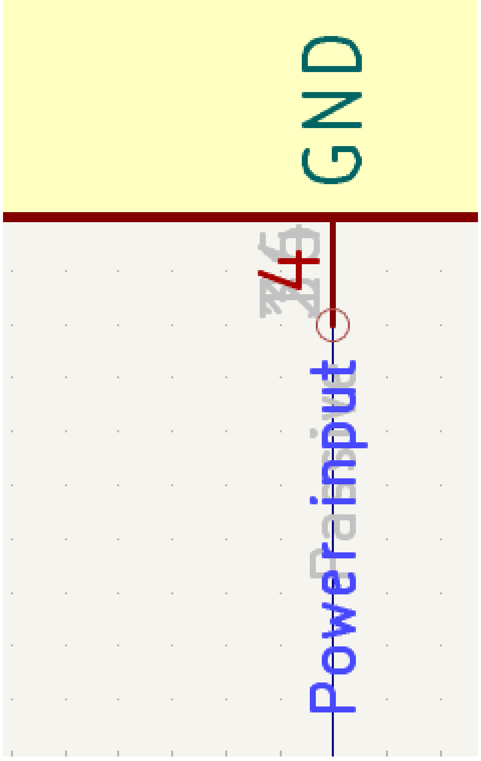
When a symbol has multiple pins that should always be connected to each other, you can choose to draw the symbol so that those pins are *stacked*. Because the pins are in the exact same location, all pins in the stack are always connected to each other and to any attached wires or labels. As an example, ground (or power) pins are often stacked in symbols where the ground pins should always be connected together.

There are two ways to draw stacked pins in the Symbol Editor; both are equivalent.

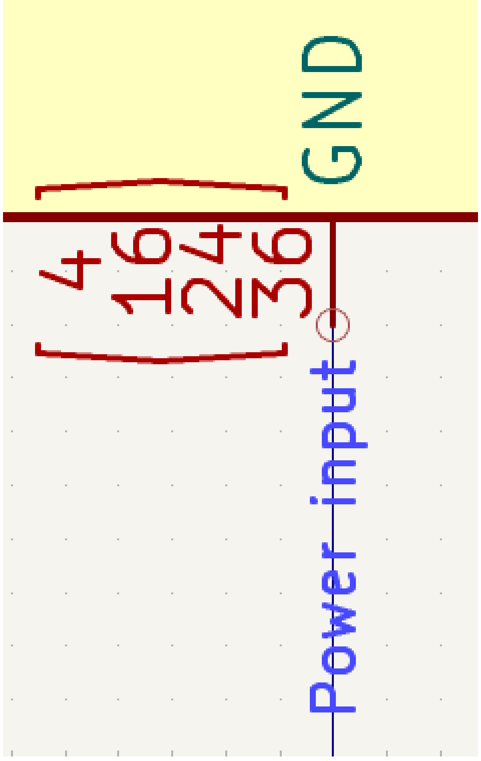
- You can place multiple individual pins in the same location so that they are literally stacked on top of each other. Each pin in the stack corresponds to a physical pin in the package (and to a pad in the footprint). Each pin in the stack has its own pin number. Because only the top pin in the stack is shown, it

You can place a single pin and enter multiple numbers in square brackets (`[]`) as the pin number. For example, a pin with the pin number given as `[1-3]` represents pins 1, 2, and 3. `[1,2,3]` and `[1-2,3]` also represent the same set of pins. Despite being shown as a single pin, all the pin numbers are displayed. This method was introduced in KiCad 10.0.

Both forms of pin stacks are equivalent and result in the same electrical connections on the PCB. The second method has the advantage of visually showing all stack's pin numbers.



Exploded pin stack (separate pins)



Single pin with multiple pin numbers

You can convert an exploded pin stack (a stack of individual pins) to a single pin with multiple pin numbers by right clicking the pin stack and choosing **Convert Stacked Pins**. You can do the reverse operation, converting a single pin with multiple pin numbers to a stack of individual pins, by right clicking the pin and choosing **Explode Stacked Pin**.

When working with exploded pin stacks (a stack of separate pins), there are several considerations to keep in mind.

- Typically, the top pin in the stack is visible, but the other pins in the stack should be marked as invisible. This has the advantage of avoiding the crowded appearance of multiple pins, their numbers, and their names printed on top of each other, but the disadvantage of not clearly showing that the additional pins exist.
- The top pin can be set to the most appropriate electrical type, but the other pins should usually be set to the Passive electrical type. This allows ERC to check connections to the pin stack, while avoiding ERC conflicts between pins in the stack. If the pin type is Power Input, it's especially important to set the hidden pins to Passive because this avoids creating hidden power input pins, which act as global labels.

WARNING Invisible pins with the Power Input type act like global labels.

Defining pins for multiple units and body styles

Symbols with multiple units and/or graphical representations are particularly problematic when creating and editing pins. Most commonly, pins are specific to each symbol unit (because each unit has a different set of pins) and to each body style (because the form and position is different between the normal body style and the alternate form).

The symbol library editor allows the simultaneous creation of pins. By default, changes made to a pin are made for all units of a multiple unit symbol and to both representations for symbols with an alternate symbolic representation. The only exception to this is the pin's graphical type and name, which remain unlinked between symbol units and body styles. This dependency was established to allow for easier pin creation and editing in most cases. This dependency can be disabled by toggling the 🔗 icon on the main tool bar. This will allow you to create pins for each unit and representation completely independently.

Pins can be common or specific to different units. Pins can also be common to both symbolic representations or specific to each symbolic representation. When a pin is common to all units, it only has to drawn once. Pins are set as common or specific in the pin properties dialog.

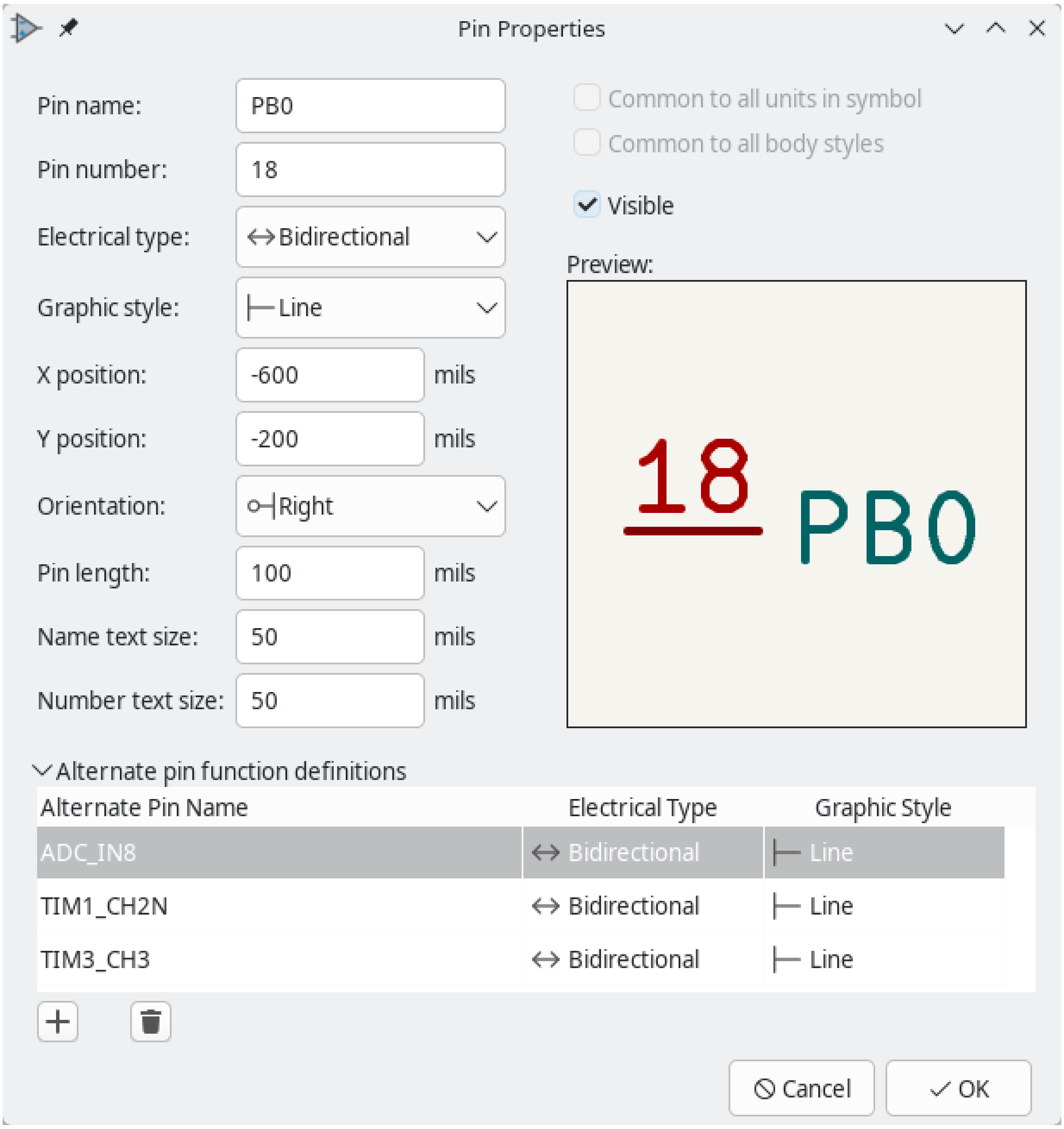
An example is the output pin in the 7400 quad dual input NAND gate. Since there are four units and two symbolic representations, there are eight separate output pins defined in the symbol definition. When creating a new 7400 symbol, unit A of the normal symbolic representation will be shown in the library editor. To edit the pin style in the alternate symbolic representation, it must first be enabled by selecting the alternate representation with the Standard ⌄ dropdown control. To edit the pin number for each unit, select the appropriate unit using the Unit A ⌄ drop down control.

Alternate pin function definitions

Symbol pins can have alternate pin functions defined for them. Alternate pin functions allow you to select a different name, electrical type, and graphical style for a pin when a symbol has been placed in the schematic.

This can be used for pins that have multiple functions, such as microcontroller pins.

Alternate pin functions are added in the Pin Properties dialog as shown below. Each alternate definition contains a pin name, electrical type, and graphic style. This microcontroller pin has all of its peripheral functions defined in the symbol as alternate pin names.



Pin name: PB0

Pin number: 18

Electrical type: ↔ Bidirectional

Graphic style: ⊢ Line

X position: -600 mils

Y position: -200 mils

Orientation: Right

Pin length: 100 mils

Name text size: 50 mils

Number text size: 50 mils

□ Common to all units in symbol

□ Common to all body styles

☑ Visible

Alternate pin function definitions

Alternate Pin Name	Electrical Type	Graphic Style
ADC_IN8	↔ Bidirectional	⊢ Line
TIM1_CH2N	↔ Bidirectional	⊢ Line
TIM3_CH3	↔ Bidirectional	⊢ Line

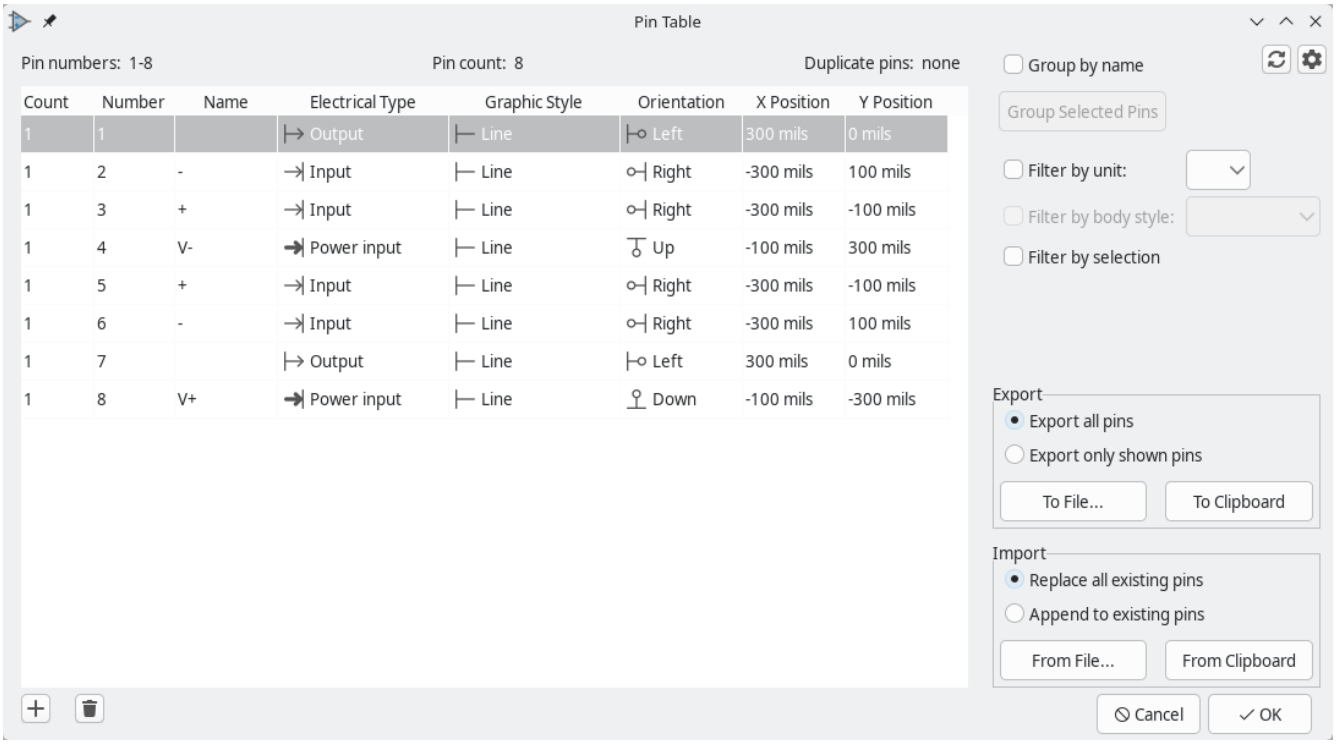
Alternate pin functions are selected in the Schematic Editor once the symbol has been placed in the schematic. For information on using alternate pin functions in the schematic, see the schematic editor symbol documentation.

Pushing pin properties to other pins

You can apply the length, name size, or number size of a pin to the other pins in the symbol by right clicking the pin and selecting **Push Pin Length**, **Push Pin Name Size**, or **Push Pin Number Size**, respectively. All other pins in the symbol will be updated.

Pin table

Another way to edit pins is to use the Pin Table, which is accessible via the ▦ button. The Pin Table displays all of the pins in the symbol and their properties in a table view, so it is useful for making bulk pin changes.



Pin numbers: 1-8 Pin count: 8 Duplicate pins: none

Count	Number	Name	Electrical Type	Graphic Style	Orientation	X Position	Y Position
1	1		Output	Line	Left	300 mils	0 mils
1	2	-	Input	Line	Right	-300 mils	100 mils
1	3	+	Input	Line	Right	-300 mils	-100 mils
1	4	V-	Power input	Line	Up	-100 mils	300 mils
1	5	+	Input	Line	Right	-300 mils	-100 mils
1	6	-	Input	Line	Right	-300 mils	100 mils
1	7		Output	Line	Left	300 mils	0 mils
1	8	V+	Power input	Line	Down	-100 mils	-300 mils

☐ Group by name

☐ Filter by unit:

☐ Filter by body style:

☐ Filter by selection

Export
● Export all pins
○ Export only shown pins

Import
● Replace all existing pins
○ Append to existing pins

Any pin property can be edited by clicking on the appropriate cell. Pins can be added and removed with the + and 🗑 icons, respectively.

NOTE Columns of the pin table can be shown or hidden by right-clicking on the header row and checking or unchecking additional columns. Some columns are hidden by default.

You can edit the same property for multiple pins simultaneously by grouping pins. Pins can be automatically grouped by name with the **Group by name** checkbox, or you manually group several pins by selecting them and clicking **Group Selected Pins**. Click the ⟳ button to clear the manual grouping. You can also filter the table to only display pins in certain units or body styles, or filter to the pins selected in the symbol editor.

If **Highlight on cross-probe** is enabled in the ⚙ menu, pins will be highlighted in the symbol editor when you select them in the pin table.

You can also import and export pins from the pin table. This lets you define and edit pins in an external spreadsheet tool. Especially for symbols with many pins, this may be more convenient than editing them within KiCad.

To export pins from the table, click either the **To File...** or **To Clipboard** button. The pins are exported in a comma-separated (CSV) format. All columns are exported, even if they are hidden in the table. If **Export all pins** is selected, all pins in the symbol are included. If **Export only shown pins** is selected, only the pins shown in the table are included.

To import pins into the table, click either the **From File...** or **From Clipboard** button. Data for import needs to be in either comma-separated (CSV) or tab-separated (TSV) format. A header row with column names is required: columns in the imported data are mapped to columns in the pin table by name. Any columns in the imported data that don't match a column in the pin table will be skipped, and any columns that are missing in the imported data will be filled with default values in the pin table. If **Replace all existing pins** is selected, the pins in the table are completely replaced by the imported pins. If **Append to existing pins** is selected, the imported pins are added to the table without modifying or replacing the existing pins.

Atribut-atribut Simbol

Symbols contain multiple fields, which are named values containing information related to the symbol. Fields can be displayed on the schematic or hidden and only shown in the symbol's properties. Some fields have special meaning to KiCad: `Reference` and `Footprint` are both critical for creating a PCB, 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 symbol will be included in the symbol when it is added to a schematic. You can also add new fields to symbols in the schematic. Whether they are in the library symbol or not, these fields can then be edited on a per-symbol basis in the schematic. They are also transferred to the symbol's corresponding footprint in the PCB.

NOTE Symbol fields are different than graphic text. In addition to being named, fields can be moved and edited in the schematic, while symbol text can only be edited in the symbol editor.

All library symbols are defined with five default fields: `Reference`, `Value`, `Footprint`, `Datasheet`, and `Description`, which are added whenever a symbol is created. These default fields cannot be deleted. Only the `Reference` field is required to have a value: the contents of a library symbol's `Reference` field is used as the reference designator prefix when the symbol is added to a schematic. In the schematic, the symbol's `Reference` field contains the entire reference designator.

The `Footprint` field, if used, contains a reference to a footprint for the symbol. The format is `LIBNAME:FOOTPRINTNAME`, where `LIBNAME` is the name of the footprint library in the footprint library table (see the Footprint Library Table section in the PCB Editor manual) and `FOOTPRINTNAME` is the name of the footprint in the library `LIBNAME`.

The `Description` field can contain text describing the symbol such as the component function, distinguishing features, and package options. Together with the symbol's name and keywords, text in this field is used when searching for symbols in the Symbol Chooser or Symbol Editor. Before KiCad version 8.0, this was a dedicated property (like the symbol name and keywords) rather than a symbol field.

Symbols defined in libraries are typically defined with only these five default fields. Additional fields such as vendor, part number, unit cost, etc. can be added to library symbols but generally this is done in the schematic editor so the additional fields can be added to every symbol in the schematic, not just all symbols of one type.

NOTE A convenient way to create additional empty symbol fields is to use define field name templates. Field name templates define empty fields that are added to each symbol when it is inserted into the schematic. Field name templates can be defined globally (for all schematics) in the Schematic Editor Preferences, or they can be defined locally (specific to each project) in the Schematic Setup dialog.

> **NOTE** If you want to manage a large amount of component data in symbol fields, consider using database libraries.

To edit an existing symbol 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 Symbol 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.

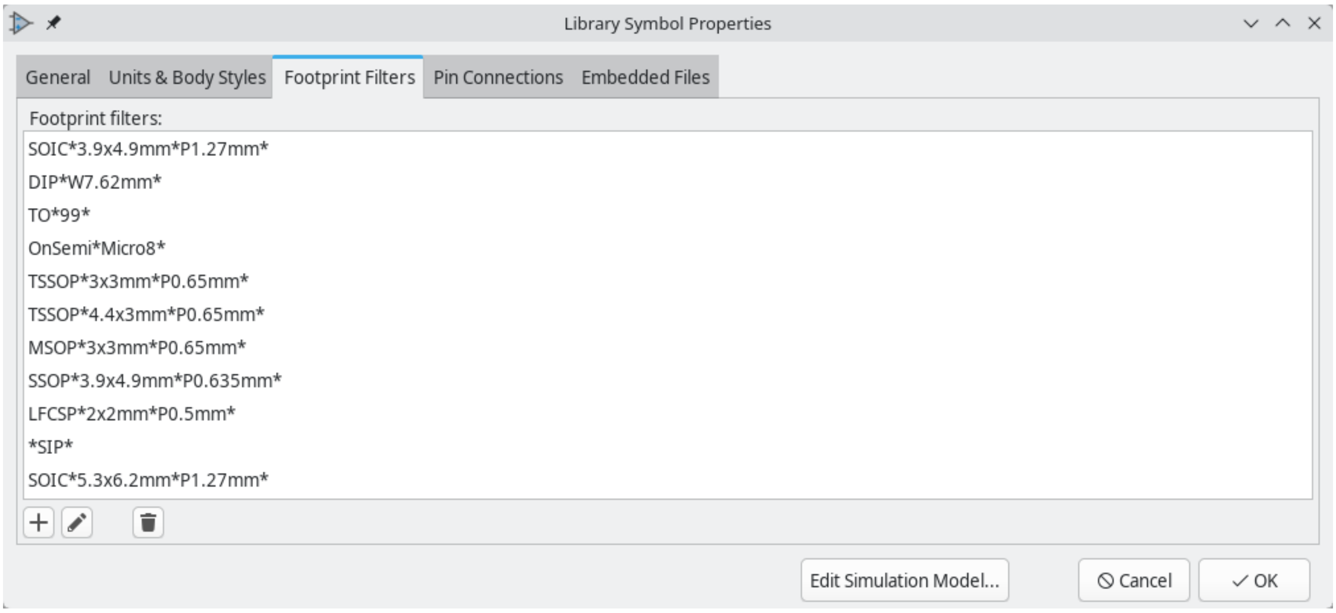
Fields have a number of properties, each of which is shown as a column in the properties grid. Not all columns are shown by default; columns can be shown or hidden by right clicking on the grid header and selecting or deselecting columns from the menu.

Footprint filters

The footprint filters tab is used to define which footprints are appropriate to use with the symbol. The filters can be applied in the Footprint Assignment tool so that only appropriate footprints are displayed for each symbol.

Multiple footprint filters can be defined. Footprints that match any of the filters will be displayed; if no filters are defined, then all footprints will be displayed.

Filters can use wildcards: `*` matches any number of characters, including zero, and `?` matches zero or one characters. For example, `SOIC-*` would match the `SOIC-8_3.9x4.9mm_P1.27mm` footprint as well as any other footprint beginning with `SOIC-`. The filter `SOT?23` matches `SOT23` as well as `SOT-23`.



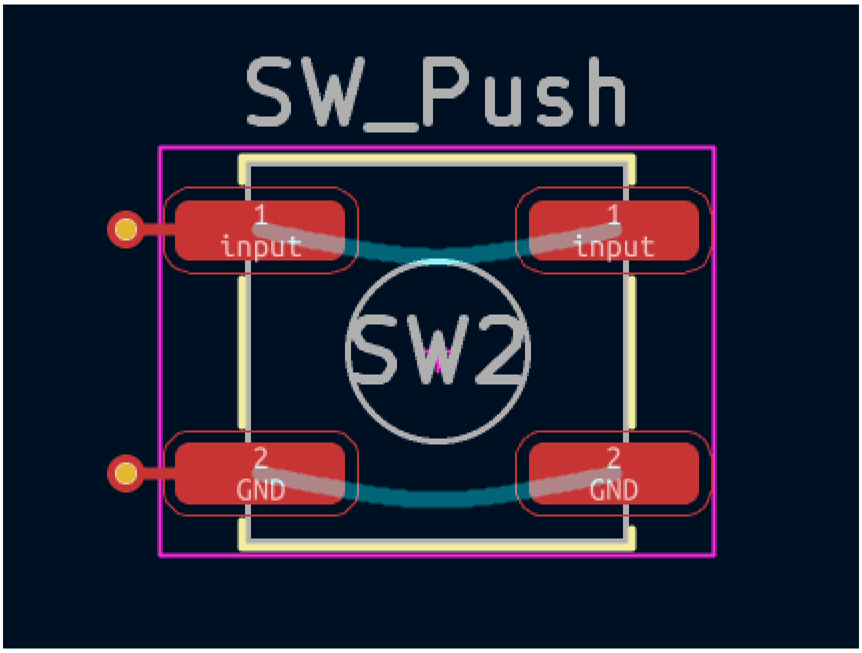
Footprint filters:

SOIC*3.9x4.9mm*P1.27mm*
DIP*W7.62mm*
TO*99*
OnSemi*Micro8*
TSSOP*3x3mm*P0.65mm*
TSSOP*4.4x3mm*P0.65mm*
MSOP*3x3mm*P0.65mm*
SSOP*3.9x4.9mm*P0.635mm*
LFCSP*2x2mm*P0.5mm*
SIP
SOIC*5.3x6.2mm*P1.27mm*

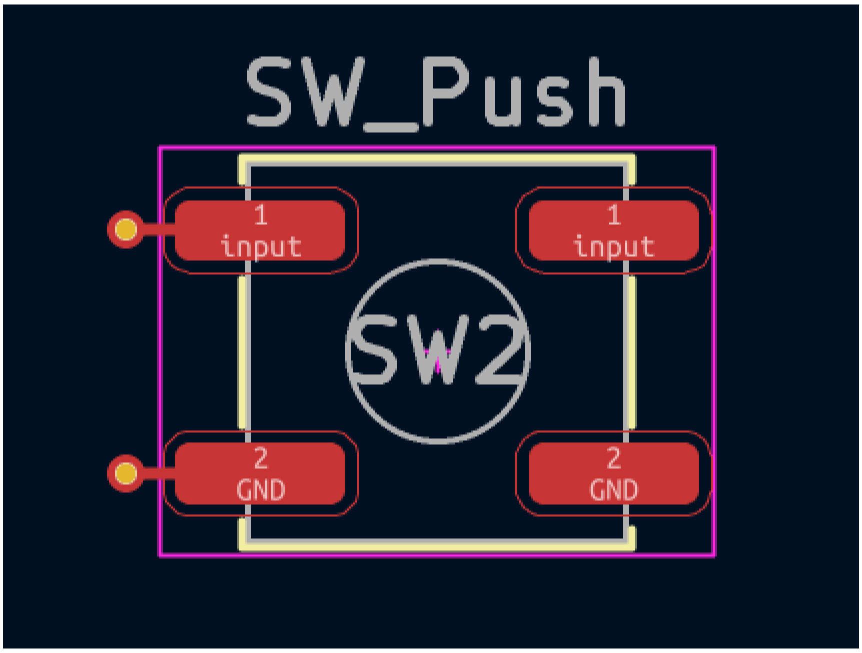
Jumper pins

You can configure a symbol to *jumper* some or all of its pins internally. Any jumpered pins will always have the same net; they are considered shorted together, even if they are not explicitly connected together in the schematic and PCB.

Symbols with jumper pins represent components that internally short together multiple pins. Examples of such components are wire jumpers, connectors with multiple connected shield pins, and switches with multiple shorted pins on each side of the switch.



A footprint without jumpered pads. The ratsnest shows missing connections between same-numbered pads.

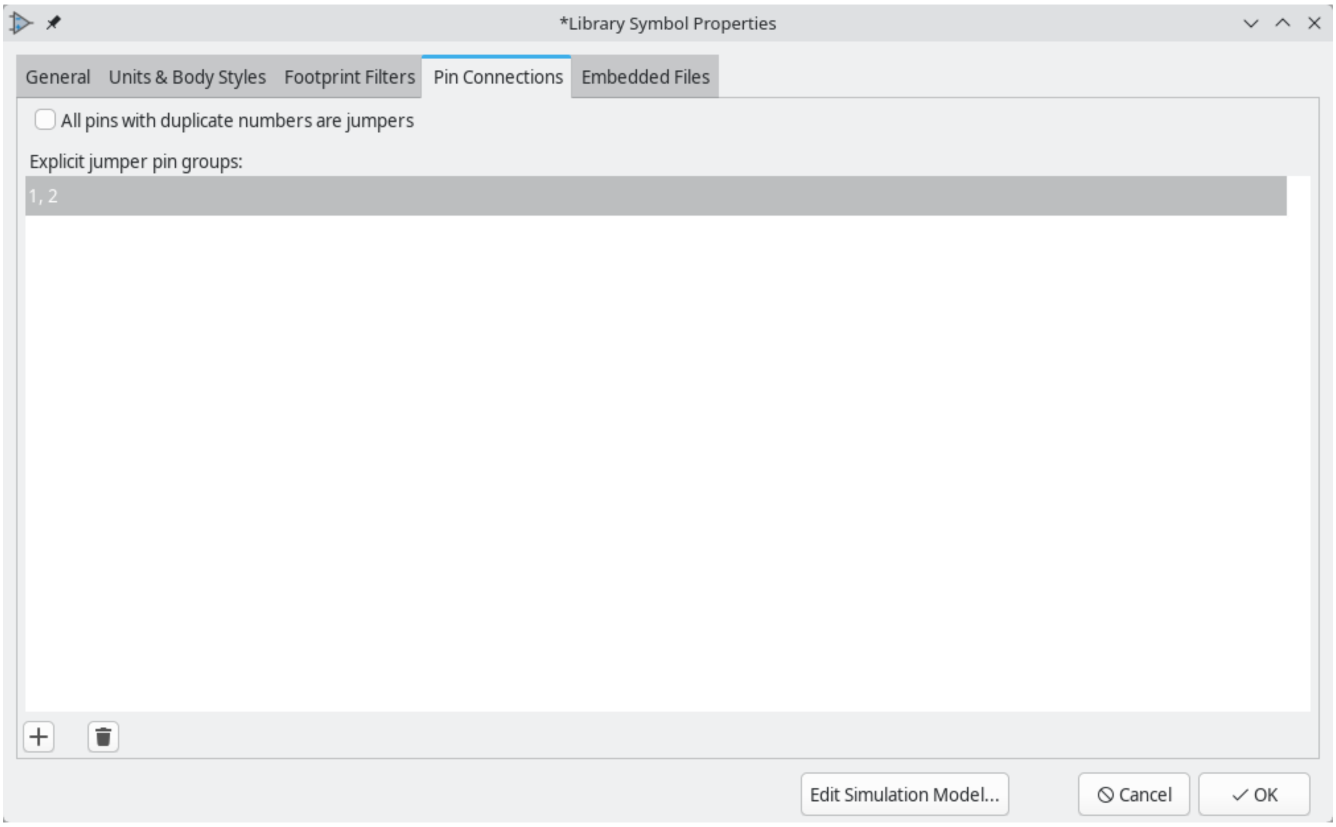


The same footprint in the same circuit with same-number pads jumpered (internally connected by the component). PCB connections between the same-numbered pads are no longer required.

Because KiCad considers the jumpered pins to be shorted by the component, you only need to connect to one of the shorted pins in the schematic and PCB.

NOTE Jumper pins 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 pins, each pin has the same net, and the electrical connection between jumper pins is assumed to be off-board in the attached component.

You can configure jumper pins in the **Pin Connections** tab of a symbol's Properties.



□ All pins with duplicate numbers are jumpers

Explicit jumper pin groups:

1, 2

When the **All pins with duplicate numbers are jumpers** setting is enabled, all pins (and pads in the linked footprint) will be jumpered to the other pins with the same number. For example, all pins numbered 1 will be considered connected to each other.

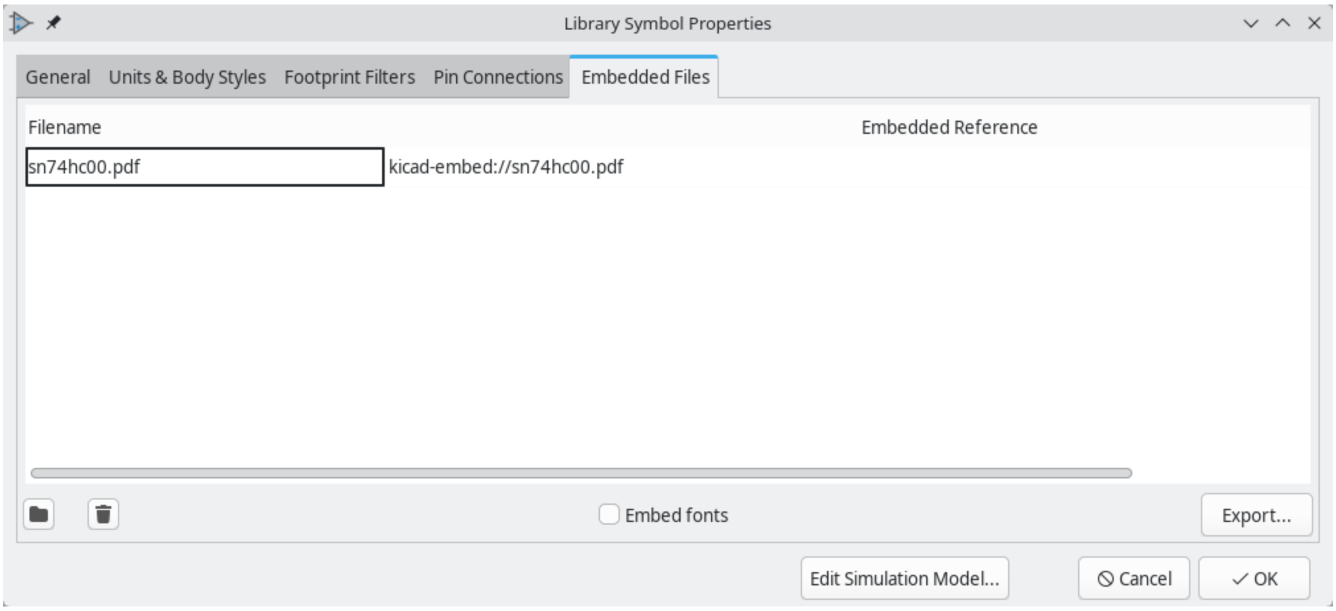
You can also configure jumper pin groups for pins that don't have the same number by adding **Explicit pin jumper groups** with the + button. Enter multiple pin numbers separated by spaces or commas. A pin jumper group with the text `1 2` or `1, 2` will jumper together pins 1 and 2. A group with `3 4 5` or `3, 4, 5` will jumper together pins 3, 4, and 5.

NOTE Like many other symbol properties, a symbol's jumper pin settings are transferred to the linked footprint. You can also configure a footprint's jumper pads using the Footprint Editor.

Embedding files

External files can be embedded within a symbol. Embedding a file stores a copy of the file inside the symbol. The symbol can then refer to the embedded copy of the file instead of the external file, which makes the symbol more portable as it doesn't rely on an external file, although the symbol library's filesize is increased as a result. In symbols this is especially useful for embedding datasheets and SPICE models.

In the symbol editor, you can embed files into library symbols in the **Embedded Files** tab of a symbol's properies.



NOTE You can add a datasheet, or a SPICE model, to a symbol and embed it in one step. To do so, browse for a datasheet (in the Symbol Properties dialog) or a SPICE model (in the SPICE Model Editor) and enable the **Embed File** checkbox in the file browser while choosing a file. This embeds the file and automatically uses the embedded reference as the file path instead of the path to the external file.

Files embedded in a symbol are deduplicated when the symbol is added to a schematic: if a file is embedded in a symbol, and multiple instances of that symbol are added to the schematic, only one copy of the file will be embedded, and all of the schematic instances will refer to the same embedded file.

When a symbol is derived from a symbol with embedded files, the derived symbol can access both its own embedded files and the files embedded in the parent symbol. Files embedded in a symbol cannot be referred to in the parent schematic outside of the symbol.

File embedding is explained in more detail in the Schematic Setup documentation.

Creating power symbols

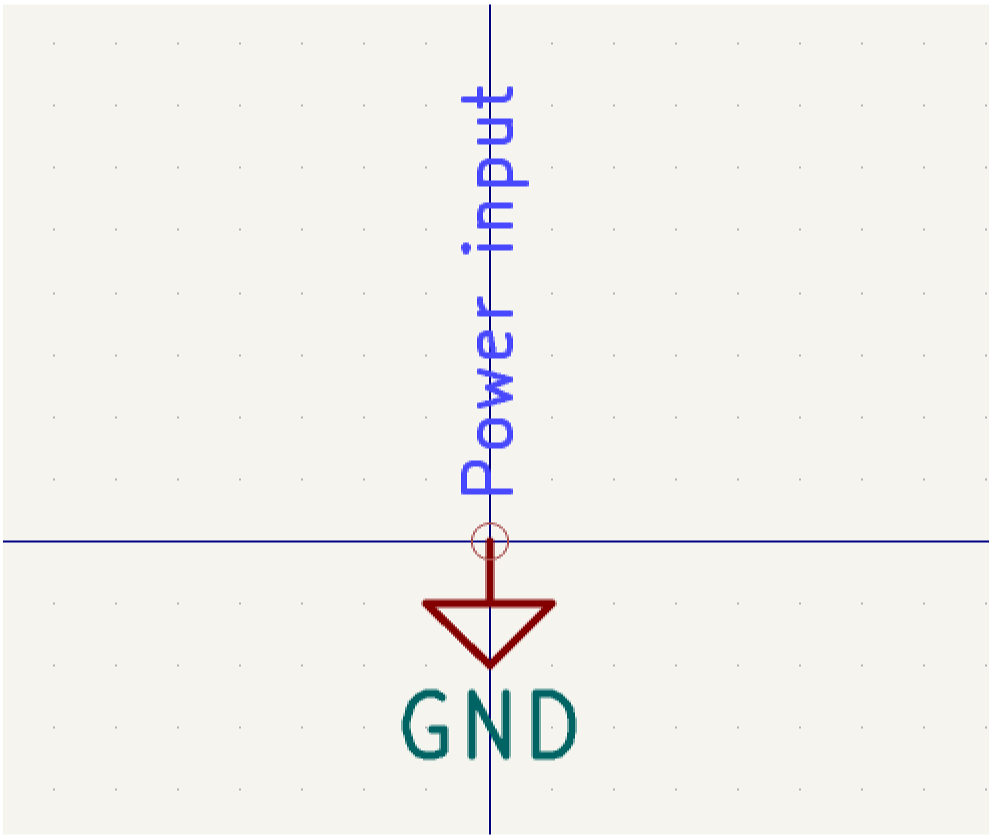
Power symbols are symbols that are used to label a wire as part of a global power net, like `VCC` or `GND`. The power symbol's `Value` field determines the net label. The behavior of power symbols is described in the electrical connections section. Power symbols are handled and created the same way as normal symbols, but there are several additional considerations described below.

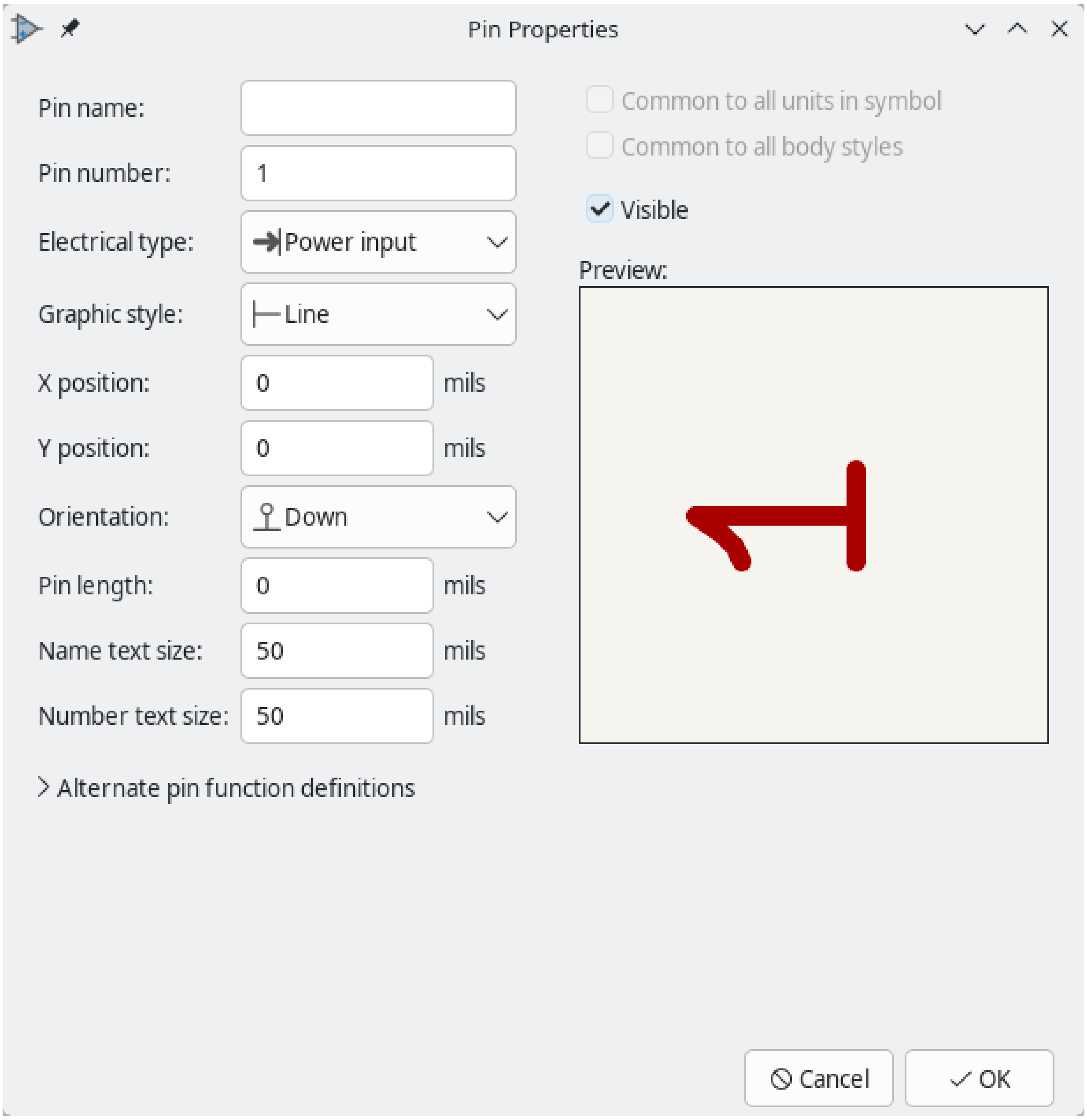
It may be useful to place power symbols in a dedicated library. KiCad's symbol library places power symbols in the `power` library, and users may create libraries to store their own power symbols. If the **Define as power symbol** box is checked in a symbol's properties, that symbol will appear in the Schematic Editor's **Add Power Symbol** dialog for convenient access.

Power symbols consist of a single pin of type Power Input. They must also have the **Define as power symbol** property checked.

NOTE In previous versions of KiCad, a power symbol's pin needed to be both a power input pin and invisible, and the pin's name determined the name of the net that the power symbol connected with. Beginning in KiCad version 8, the pin in a power symbol does not need to be invisible, and the net is determined by the power symbol's `Value` field.

Below is an example of a `GND` power symbol.





Pin Properties

Pin name:

Pin number: 1

Electrical type: Power input

Graphic style: Line

X position: 0 mils

Y position: 0 mils

Orientation: Down

Pin length: 0 mils

Name text size: 50 mils

Number text size: 50 mils

□ Common to all units in symbol

□ Common to all body styles

☑ Visible

Preview:

Untuk membuat sebuah simbol *power*, lakukan langkah-langkah berikut:

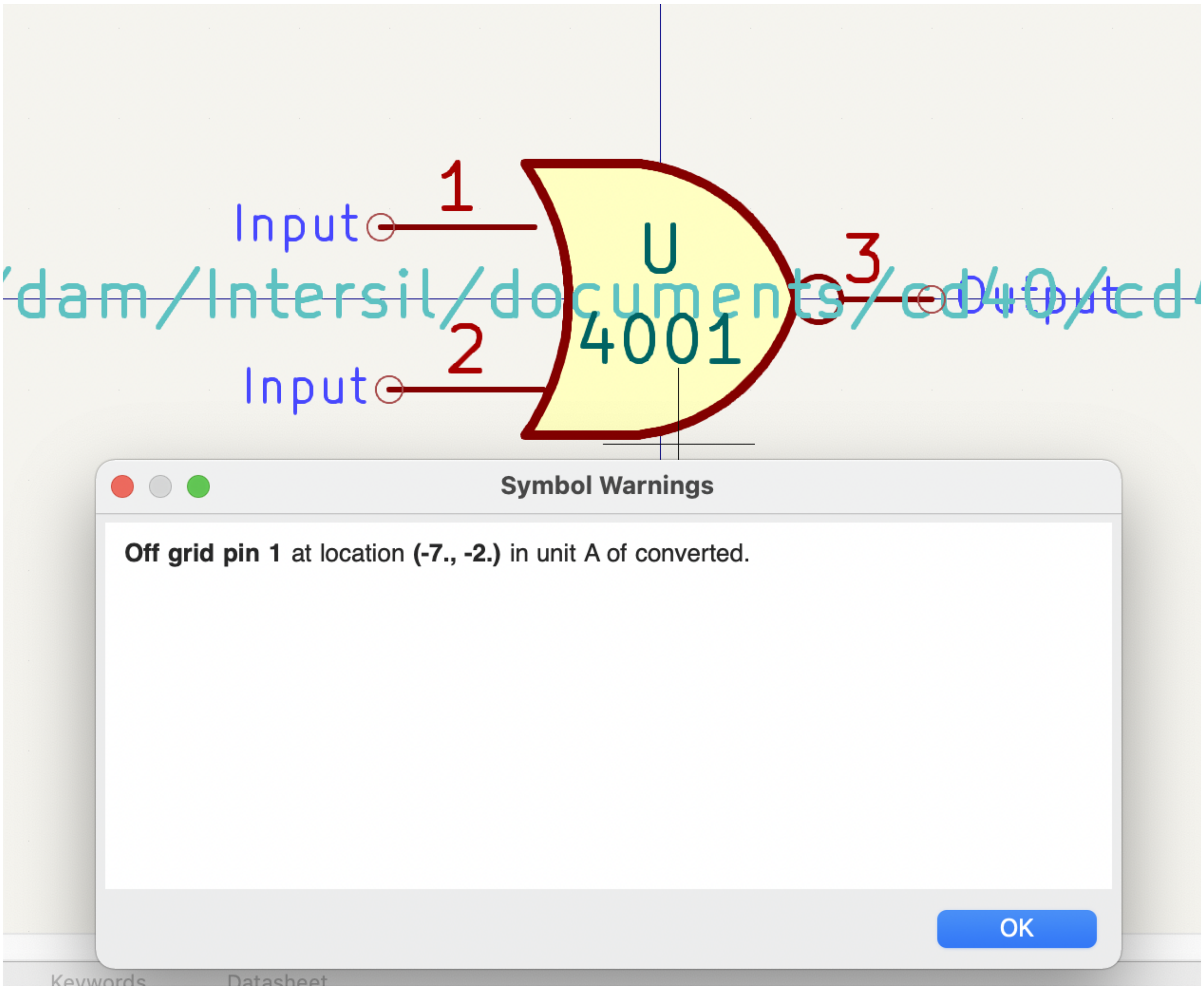
- Add a pin of type **Power input**. Make the pin number `1`, the length `0`, set the graphic style to **Line**, and make the pin **visible**. The pin number, name, length, and line style do not matter electrically.
- Place the pin on the symbol anchor. This is not required but makes it easier to place the power symbol in the schematic.
- Use the shape tools to draw the symbol graphics.
- Set the symbol value to the desired net name. The symbol value is electrically important: it determines the symbol's connected net name. This field can be changed later, after the symbol has been placed in the schematic, which will change which net the symbol connects to.
- Check the **Define as power symbol** box in Symbol Properties window. This makes the symbol appear in the **Add Power Symbol** dialog, prevents the symbol from being assigned a footprint, and excludes the symbol from the board, BOM, and netlists.
- Also deselect the **Show pin number** and **Show pin name** options in the Symbol Properties window. This is not necessary but improves the symbol's appearance.

- Set the symbol reference and uncheck the **Show** box. The reference text is not important except for the first character, which should be `#`. For the power symbol shown above, the reference could be `#GND`. Symbols with references that begin with `#` are not added to the PCB, are not included in Bill of Materials exports or netlists, and they cannot be assigned a footprint in the footprint assignment tool. If a power symbol's reference does not begin with `#`, the character will be inserted automatically when the annotation or footprint assignment tools are run.

An easier method to create a new power symbol is to use another symbol as a starting point.

Checking symbols

The Symbol Editor can check for common issues in your symbols. Run the symbol checker using the button in the top toolbar.



Off grid pin 1 at location **(-7., -2.)** in unit A of converted.

The symbol checker checks for:

- Pins that are off-grid (pins are considered off grid if their position is not a multiple of the current symbol editor grid. It is strongly recommended to use a 50 mil grid for symbol pins)
- Pins that are duplicated
- Issues with graphical shapes, such as zero-sized shapes
- Illegal reference designator prefixes: reference designator prefixes should not end with a number or `?`

- Incorrectly designed power symbols. Power symbols should have:
  - A single unit
  - No alternate body styles
  - A single pin which is either of type Power Output (see PWR_FLAG) or visible and of type Power Input (see power symbols)
- Hidden Power Input pins in non-power symbols: these create implicit connections and are not recommended

NOTE	In previous versions of KiCad, power symbols required an invisible power input pin so that they would make a global connection. In KiCad 8, the power input pin does not need to be invisible. Therefore the symbol checker will report if invisible power input pins are detected.

Importing schematics from other applications

KiCad can import complete schematics saved in several third-party EDA tool formats, converting them to KiCad's native format in a single step. This allows you to migrate existing designs without having to recreate them from scratch.

NOTE

Importing a schematic replaces the current schematic with the contents of the imported file. It does not merge the foreign schematic into an existing KiCad design. After import, save the schematic using **File** → **Save** to create a KiCad `.kicad_sch` file.

How to import a schematic

To import a schematic from a supported third-party format, use **File** → **Import** → **Non-KiCad Schematic…**.

An "Import Schematic" file dialog will open. The dialog presents an "All supported formats" filter that includes all importable file types, as well as individual format filters. Select the desired source file and click **Open**.

KiCad reads the foreign file, converts it to KiCad format, and loads the result into the schematic editor. If the importer encounters any issues during conversion, a report dialog will appear listing warnings and unsupported features.

TIP

If you open a file using **File** → **Open…** and the file is not a KiCad schematic, the Schematic Editor will display a message directing you to use **File** → **Import** → **Non-KiCad Schematic…** instead.

Supported formats

KiCad supports importing schematics from the following EDA tools:

Source EDA Tool	File Extension(s)	Notes
Altium Designer	`.SchDoc`	Multi-sheet designs are supported; each `.SchDoc` file corresponds to one sheet.
Eagle (Autodesk)	`.sch` (XML)	Eagle version 6.x and later (XML format).
LTspice	`.asc`	LTspice schematic and simulation files.
PADS Logic	`.asc`, `.txt`	PADS Logic ASCII export format.
CADSTAR Schematic Archive	`.csa`	Zuken CADSTAR schematic archive files.
gEDA / Lepton EDA	`.sch`	gEDA gschem and Lepton EDA schematic files.
EasyEDA (JLCEDA) Standard	`.json`	EasyEDA Standard edition JSON export.
EasyEDA (JLCEDA) Professional	`.epro`, `.zip`	EasyEDA Professional edition project files.

NOTE The Eagle `.sch` extension and the gEDA `.sch` extension are both used by their respective tools. KiCad identifies the correct format automatically based on the file contents.

NOTE The PADS Logic `.asc` extension is shared with LTspice. KiCad inspects the file header to determine which format applies.

Format-specific notes

Altium Designer

KiCad imports Altium Designer schematic files (`.SchDoc`). Altium projects can contain multiple sheets, each saved as a separate `.SchDoc` file. When you import one sheet, KiCad will also attempt to load the other sheets referenced by that project, reconstructing the sheet hierarchy.

Symbols found in the source file are placed into a generated symbol library that is saved alongside the imported schematic.

NOTE If you have a complete Altium project (`.PrjPcb`) with both schematic and PCB files, the KiCad Project Manager can import both in a single step. Use **File** → **Import Non-KiCad Project…** → **Altium Project…** and select the `.PrjPcb` file. The `.SchDoc` import described here is for schematic files imported on their own.

Eagle (Autodesk)

KiCad imports Eagle XML schematic files (`.sch`), which are the format produced by Eagle version 6.x and later. Eagle schematics store all sheets in a single file.

Symbols are extracted from the embedded Eagle libraries and placed into a generated KiCad symbol library.

NOTE

Binary Eagle files created by versions earlier than 6.x are not supported. Open the file in Eagle first and re-save it to convert to the XML format before importing.

LTspice

KiCad imports LTspice schematic files (`.asc`). These are the simulation schematics created by LTspice, which uses its own symbol library and netlist conventions.

After import, the schematic will reflect the circuit topology from LTspice, but simulation directives and LTspice-specific elements such as SPICE parameters and behavioral sources may not convert fully. Review the imported schematic and the KiCad SPICE simulator documentation before attempting to run simulations on an imported LTspice design.

PADS Logic

KiCad imports PADS Logic schematic files (`.asc` or `.txt`). These are the ASCII text exports produced by the PADS Logic schematic editor. To generate this file from PADS Logic, use **File** → **Export** and select the ASCII format.

CADSTAR Schematic Archive

KiCad imports CADSTAR Schematic Archive files (`.csa`) from Zuken's CADSTAR tool.

In addition to full schematic import, KiCad can use CADSTAR Parts Library files (`.lib`) as read-only symbol libraries through the Symbol Library Manager.

gEDA / Lepton EDA

KiCad imports gEDA gschem and Lepton EDA schematic files (`.sch`). These are ASCII text files produced by the gEDA suite or its fork, Lepton EDA.

Symbols in gEDA schematics reference external `.sym` files from the gEDA system symbol library. KiCad will search standard gEDA library locations for these symbol files during import. Symbols that cannot be found are replaced with a fallback rectangular symbol. Hierarchical sheets are supported; KiCad will recursively import sub-schematics referenced by `source=` attributes on component instances.

EasyEDA (JLCEDA) Standard

KiCad imports EasyEDA Standard edition schematic files (`.json`). To obtain a file for import, export your design from EasyEDA using the **Export EasyEDA** option, which produces a JSON file.

Symbols are extracted from the embedded library data within the JSON file and placed into a generated KiCad symbol library.

EasyEDA (JLCEDA) Professional

KiCad imports EasyEDA Professional edition project files (`.epro` or `.zip`).

For projects containing multiple schematics, KiCad will present a project chooser dialog so you can select which schematic within the project to import.

After importing

After importing a schematic from a third-party tool, review the result and perform any necessary cleanup:

1. **Save the schematic:** Use **File** → **Save** to save the imported design as a KiCad `.kicad_sch` file. The file will be placed in the current project directory with the project name.
2. **Run the Electrical Rules Check (ERC):** Select **Inspect** → **Electrical Rules Checker** to identify any connectivity or symbol issues introduced during import.
3. **Review the import report:** If the importer produced warnings, the report dialog will have been shown automatically. Review any warnings about unsupported features or items that could not be converted.
4. **Remap symbols:** Imported symbols are stored in a generated library created alongside the schematic. If you want to use KiCad standard library symbols instead, use **Tools** → **Edit Symbol Fields** to reassign symbols to your preferred library. The symbol remapping tool may also be useful.
5. **Assign footprints:** Third-party schematics rarely use KiCad footprint references. Use **Tools** → **Assign Footprints** to link each symbol to an appropriate KiCad footprint before proceeding to board layout.
6. **Check the drawing sheet:** Imported schematics do not carry a KiCad drawing sheet (title block). The schematic editor will use an empty drawing sheet after import. To add a title block, open **File** → **Page Settings** and choose a drawing sheet template.

Design blocks

Design blocks are a design-reuse feature that lets you save a portion of a schematic (and optionally a corresponding PCB layout) as a named, reusable fragment stored in a design block library. You can instantiate a design block any number of times within the same project or across different projects. Design blocks are organized in libraries in the same way that symbols and footprints are organized in their respective libraries.

A single design block can contain a schematic fragment, a PCB layout fragment, or both. When you place a schematic design block, the saved schematic fragment is inserted into the current schematic sheet or into a new hierarchical subsheet. If the design block also contains a layout fragment, you can later apply that layout to the footprints that were placed as a result of the schematic fragment. This makes design blocks well suited for circuits that appear repeatedly across projects, such as power supply subcircuits, filter networks, protection circuits, and interface conditioning stages.

Typically, design blocks are created first as a schematic fragment, with the layout added later once the circuit has been placed and routed on a board. Reusing layout fragments in design blocks is covered in the PCB Editor manual.

Design block concepts

Two ways to place a design block

When you place a design block into a schematic you choose between two fundamentally different placement modes:

Place inline (no sheet): The contents of the design block are inserted directly into the current schematic sheet alongside all other schematic objects. The content becomes part of the current sheet and can be edited in place.

Place as sheet: The contents of the design block are placed into a new hierarchical subsheet. The current schematic sheet receives a hierarchical sheet symbol whose contents are the design block's schematic. Placing as a sheet is useful when the design block represents a logically distinct functional block, when you want the design block contents to be visually separate from the parent schematic, or when you plan to reuse the same sheet multiple times as a repeated hierarchical sheet. Default fields defined in the design block's properties are applied as hierarchical sheet fields when placing as a sheet.

In either mode, also enable the **Place as group** option to wrap the placed content in a group with a *library link*. Without a group, features including saving changes back to the library, placing additional linked instances, and applying the stored layout to the corresponding footprints in the PCB editor are not available. The **Place as group** option works with both placement modes: when enabled alongside **Place as sheet**, a group wrapping the sheet symbol is added to the parent schematic.

NOTE	Design blocks do not support nested hierarchical sheets. If the sheet you are saving as a design block contains hierarchical sheet symbols (subsheets), the save operation will be rejected.

The library link

When a design block is placed as a group, the group stores a *library link* in the format `<library>:<block>` — for example, `analog_filters:sallen_key_lowpass`. This link connects the placed instance to its source in the library and enables the following operations:

- **Save to Linked Design Block** — push edits made to the placed group back to the library, updating the stored design block schematic.
- **Place Linked Design Block** — place another instance of the same design block as a new linked group, without reopening the Design Blocks panel.
- **Apply Design Block Layout** — in the PCB editor, apply the layout stored in the linked design block to the group of footprints corresponding to this design block instance. As an alternative, the multichannel tool can replicate a manually completed layout from one group to another without requiring a stored layout fragment in the design block.

A group that does not have a library link cannot use any of these operations. If you manually grouped the contents of a design block without using the Design Blocks panel, you can add the library link retroactively in the group's properties dialog (see Setting a library link manually).

Setting a library link manually

If you have a group in your schematic that contains the same content as an existing design block but the group does not have a library link, you can add the link without re-placing the design block:

1. Select the group and open its properties (right click → **Group Properties...**, or press E with the group selected).
2. In the **Library link** field, enter the design block identifier in the format `<library>:<block>`. For example, `power_circuits:ldo_3v3`.
3. Click **OK**.

The group will now behave as a linked group and all library-linked operations will be available.

If you want the design block's schematic contents to also be grouped in the PCB, either for organizational purposes or so you can apply a design block layout, use the Update PCB from Schematic tool with the **Group footprints based on symbol groups** option enabled.

Design block library structure

Design block libraries are stored as directories on the filesystem. The directory structure has two levels of nesting:

- The library itself is a directory whose name ends with `.kicad_blocks` (for example, `my_power_circuits.kicad_blocks`).
- Each design block within the library is a subdirectory inside the library directory, with the suffix `.kicad_block` (for example, `ldo_3v3.kicad_block`).
- Inside each design block directory, KiCad stores:
    -

  - A schematic file (`<block_name>.kicad_sch`) — present if the design block contains a schematic fragment.
  - A board file (`<block_name>.kicad_pcb`) — present if the design block contains a layout fragment.
  - A metadata file (`<block_name>.json`) — always present; stores the design block's description, keywords, and default fields.

Because libraries and design blocks are plain filesystem directories, they can be managed with ordinary file manager tools, version control systems, or shared over a network filesystem. The library tables (see below) must be updated if library directories are moved or renamed outside of KiCad.

Global and project library tables

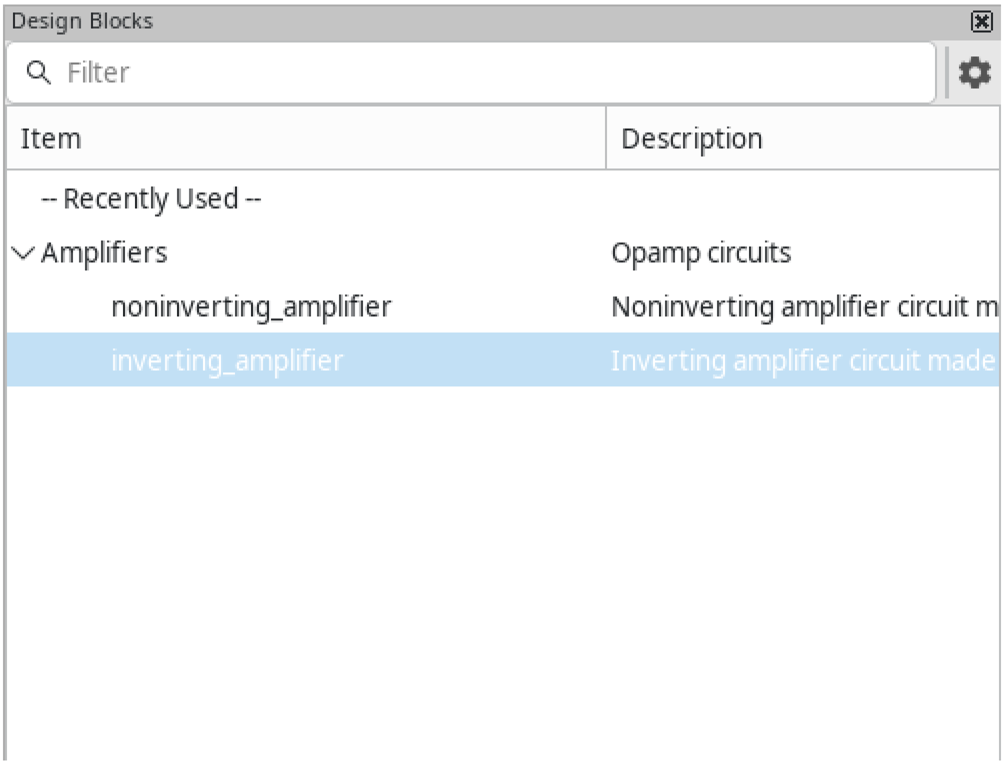
As with symbol and footprint libraries, design block libraries are registered in library tables before KiCad can use them. There are two library tables:

- The **global design block library table** lists libraries that are available in every KiCad project. The global table is shared across all projects on the same machine.
- The **project design block library table** lists libraries that are available only in the current project. The project table is stored alongside the project files.

Both tables are managed through **Preferences** → **Manage Design Block Libraries…**. The dialog behaves identically to the symbol and footprint library table dialogs. See the symbol library table documentation for a full description of the library table interface.

The Design Blocks panel

To use schematic 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**.



Item	Description
-- Recently Used --	
Amplifiers	Opamp circuits
noninverting_amplifier	Noninverting amplifier circuit m
inverting_amplifier	Inverting amplifier circuit made

The Design Blocks panel lets you create design block libraries and create, edit, and place design blocks in the schematic.

The panel has three main areas:

- A **filter textbox** at the top, for searching across all libraries by name, description, and keywords. By default results are ranked by best match. To sort alphabetically instead, select **Sort Alphabetically** under the ⚙ settings menu.
- A **library tree** that lists all registered design block libraries and the design blocks they contain. Libraries can be expanded or collapsed. A **Recently Used** pseudo-library at the top of the tree records design blocks you have placed recently. Right click a library and choose **Pin Library** to keep it at the top of the list regardless of recent usage.
- A **preview area** below the tree that shows the name, description, keywords, and a graphical preview of the currently selected design block.

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 symbol 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

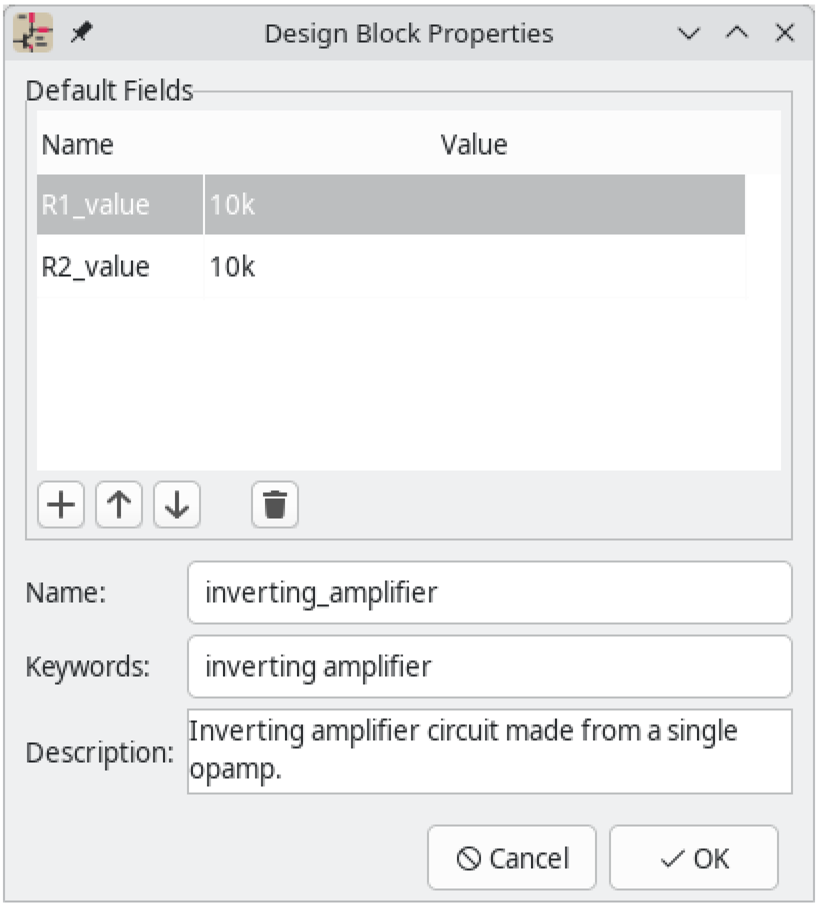
In the schematic editor, design blocks can be created either from the entire contents of a schematic sheet or from a selection of schematic objects. To create and save a new design block, select the desired source objects, either by opening the desired sheet or selecting the objects in the editing canvas. Then right click the design block library that will contain the block and select **Save Current Sheet as Design Block…** or **Save Selection as Design Block…** as appropriate.

The two creation commands differ in what they capture:

- **Save Current Sheet as Design Block** captures everything on the current schematic sheet. Navigate to the sheet you want to save before invoking this command. This is the most convenient option when the circuit you want to capture already exists on its own dedicated sheet.

- **Save Selection as Design Block** captures only the objects that are currently selected in the canvas. Select the symbols, wires, labels, and other schematic objects you want to include before invoking this command. If the selection is a single hierarchical sheet symbol, KiCad saves the contents of that subsheet (equivalent to **Save Current Sheet** for that sheet). If the selection contains multiple items or a single group, those items are saved as the design block content.

After invoking either command, the **Design Block Properties** dialog opens.



Name	Value
R1_value	10k
R2_value	10k

Name: inverting_amplifier

Keywords: inverting amplifier

Description: Inverting amplifier circuit made from a single opamp.

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. The name may not contain the characters `/ : ; \ | < > " * ?` or any character that is illegal in a library item name.
- **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. Use default fields to pre-populate sheet-level parameters such as supply voltage, channel number, or other values that are expected to be overridden when the design block is placed as a sheet.

After filling in the properties and clicking **OK**, the design block is saved to the selected library. If a design block with the same name already exists in the library, KiCad will ask for confirmation before overwriting it.

When you save a selection as a design block (as opposed to saving an entire sheet), KiCad automatically groups the selected objects and sets the group's library link to point to the newly created design block. This means the selection is ready for further design block operations immediately after saving.

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…**.

Deleting a design block

To delete a design block from a library, right click the design block in the library tree and select **Delete Design Block**. KiCad will ask for confirmation before permanently removing the design block and its directory from the filesystem. This operation cannot be undone.

Deleting a design block from the library does not affect any schematic objects that were previously placed from that design block. Existing placed groups that had a library link to the deleted design block retain their content; they simply lose the ability to use library-linked operations.

Updating an existing design block

There are multiple ways to push schematic changes back to an existing design block in the library. All of them replace the schematic content stored in the design block with new content; they do not merge changes.

Updating from a selection

This is the most general approach and works regardless of how the design block was originally placed:

1. Select the schematic objects that represent the new content for the design block.
2. In the Design Blocks panel, right click the design block you want to update and choose **Update Design Block from Selection**.

The stored schematic of the design block is replaced by the selection. If the design block previously had a schematic but you are updating with a new one, KiCad will ask for confirmation before overwriting.

Updating from the current sheet

If the current sheet contains exactly the content you want to store in the design block (for example, if the design block was originally placed as a hierarchical sheet and the sheet has since been edited):

1. Navigate to the sheet whose contents you want to store.
2. In the Design Blocks panel, right click the design block you want to update and choose **Update Design Block from Current Sheet**.

The stored schematic of the design block is replaced by everything on the current sheet.

Saving from a linked group

If the design block was placed as a group and the group has a library link:

1. Edit the schematic contents inside the group.
2. Right click the group in the canvas and choose **Save to Linked Design Block**.

This is the most direct workflow when iterating on a placed design block instance: edit in place, then push back to the library with a single right click.

> **NOTE** Updating a design block in the library does not automatically update previously placed instances of that design block. Placed instances are independent copies of the design block content at the time they were placed. To bring an existing placed instance in line with the current library version, delete it and place the updated design block again.

Using design blocks in a schematic

You can place a design block into a schematic 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**.

Placement options

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 Esc 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 schematic, and the group's library link will be set to point to the source design block and library. Grouping a design block keeps its contents together as a single unit in the schematic and enables library-linked operations such as saving changes back to the library, placing additional linked instances, and applying the design block layout in the PCB editor. The group can also be copied to the corresponding footprints in the board.
- If the **Place as sheet** checkbox is enabled, you will need to click twice in the editing canvas to place two corners of a hierarchical sheet. The design block contents will be placed in the new sheet. If the **Place as sheet** checkbox is not enabled, clicking once in the canvas will place the contents of the design block directly into the current schematic. **Place as group** and **Place as sheet** can be enabled simultaneously: when both are enabled, a new hierarchical sheet is created and a group containing the sheet symbol is added to the parent schematic.
-

If the **Keep annotations** checkbox is enabled, KiCad will insert the design block without changing the symbol annotations as defined in the saved design block. If it is not enabled, KiCad will reset the symbol annotations while inserting the design block and reannotate all of the symbols in the block according to the current annotation settings. Keeping annotations is useful when you intend to place only a single instance of the design block in the schematic and the original reference designators are meaningful. When placing multiple instances, disable this option so that KiCad assigns unique reference designators to each instance automatically.

After placement

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. If the design block was placed as a group, the group boundary is visible in the canvas. You can enter the group to edit individual items inside it, or select the entire group as a unit to move, copy, or delete all of its contents together.

Practical workflows

Workflow: creating a reusable power circuit

This workflow demonstrates saving a power supply subcircuit as a design block and reusing it in multiple projects.

1. **Draw the circuit.** In the schematic editor, draw the power supply circuit including symbols, wires, net labels, and power symbols. Annotate the schematic and verify it passes ERC.

2. **Create or select a library.** Open the Design Blocks panel (**View** → **Panels** → **Design Blocks**). If you do not already have a suitable library, right click in the library tree and choose **New Library…**. Give the library a descriptive name (for example, `power_circuits`) and save it to an appropriate location. Add it to the global library table so it is available in all projects.

3. **Select the circuit.** In the canvas, draw a selection box around all the symbols, wires, and labels that make up the power supply circuit.

4. **Save as design block.** Right click the target library in the Design Blocks panel and choose **Save Selection as Design Block…**. In the Design Block Properties dialog, enter a meaningful name, a brief description, and relevant keywords (for example, `LDO 3.3V linear regulator`). Click **OK**. KiCad saves the design block and automatically groups the selected objects with a library link to the new design block.

5. **Reuse in another project.** Open a different project. Open the Design Blocks panel and expand the `power_circuits` library. Find the design block you created (use the filter if needed). Enable **Place as group** and, if the reference designators should be reassigned, disable **Keep annotations**. Double click the design block to begin placement and click in the canvas to place it.

6. **Update the design block later.** If you improve the power supply circuit, select the placed group, edit it as needed, then right click the group and choose **Save to Linked Design Block** to push the changes back to the library.

Workflow: sheet-based reuse with hierarchical parameters

This workflow is suited for a design block that represents a complete functional block with parameters that vary between instances, such as a per-channel analog conditioning circuit.

1. **Draw the circuit on a dedicated sheet.** Create a hierarchical sheet in your project and draw the channel circuit on it, using hierarchical labels for signals that connect to the parent sheet. If the circuit has configurable values (such as a gain-setting resistor value), note which sheet fields you will use to communicate these.

2. **Define default fields.** Open the Design Blocks panel and save the sheet as a design block using **Save Current Sheet as Design Block…**. In the Design Block Properties dialog, add default fields for each hierarchical parameter. For example, add a field named `GAIN_R` with a default value of `10k`.

3. **Place as a sheet.** In the parent schematic, open the Design Blocks panel, select the design block, enable **Place as sheet**, and click twice in the canvas to define the sheet symbol boundary. KiCad creates a new hierarchical sheet symbol and populates it with the default fields from the design block.

4. **Customize each instance.** For each placed hierarchical sheet, open the sheet properties and adjust the field values to match the requirements for that particular instance (for example, different resistor values for different gain settings). The sheet contents are shared (all instances use the same underlying schematic file from the design block), but the field values are per-instance.

5. **Updating.** If the underlying circuit changes, save the modified sheet as an update to the design block using **Update Design Block from Current Sheet**. All instances of the hierarchical sheet in the project will automatically reflect the updated schematic, because they all reference the same subsheet file.

Simulator

KiCad provides an embedded electrical circuit simulator using ngspice as the simulation engine. ngspice is a SPICE simulator derived from the original widely used Berkeley SPICE program. KiCad's simulator can also run simulations using IBIS models of device pins.

The process of creating and running a simulation in KiCad has two main parts:

1. Drawing a simulation schematic in KiCad's Schematic Editor. Schematics for simulations are similar to normal schematics (and can even be identical), but they typically include simulation-specific devices, such as sources, and may exclude devices that are irrelevant for simulation, such as connectors. Creating a schematic simulation requires ensuring all symbols in the schematic have appropriate models assigned. Finding or creating simulation models, and then validating them, can be a significant portion of the process. KiCad includes some simple simulation models for basic devices such as sources, passive devices, and generic semiconductor devices, but beyond those you will need to find or create your own models.
2. Running the simulation using the simulator tool. This includes choosing the type of analysis (transient, AC, etc.) and configuring its options. Multiple different analyses can be performed. The simulator provides a plot window to view and analyze simulation results.

When drawing schematics for simulation, the `Simulation_SPICE` symbol library, installed with KiCad by default, may be useful. It contains common elements used for simulation such as voltage and current sources, generic semiconductor symbols, and other simulation-specific devices. The symbols in this library are described in detail below.

While these elements enable a great variety of simulation work, users familiar with SPICE in other environments will be used to incorporating models of commercially-available semiconductors, integrated circuits, and other devices as more complex SPICE models. Indeed, semiconductor manufacturers often freely supply these to help users simulate and develop circuits using their parts. Note that while KiCad does not include any commercial SPICE models in its distribution, you are free to use any models you may have, or have used with other circuit simulators, in KiCad's simulator.

In general, if a model works with other SPICE simulators, it should work with the KiCad simulator, although some SPICE simulators implement extensions that are unsupported by ngspice. ngspice offers several compatibility modes to improve compatibility with other simulators.

Finally, to quickly showcase the capabilities of the KiCad simulator, some demonstration projects are included in the KiCad distribution. They can be found in the `demos/simulation` directory.

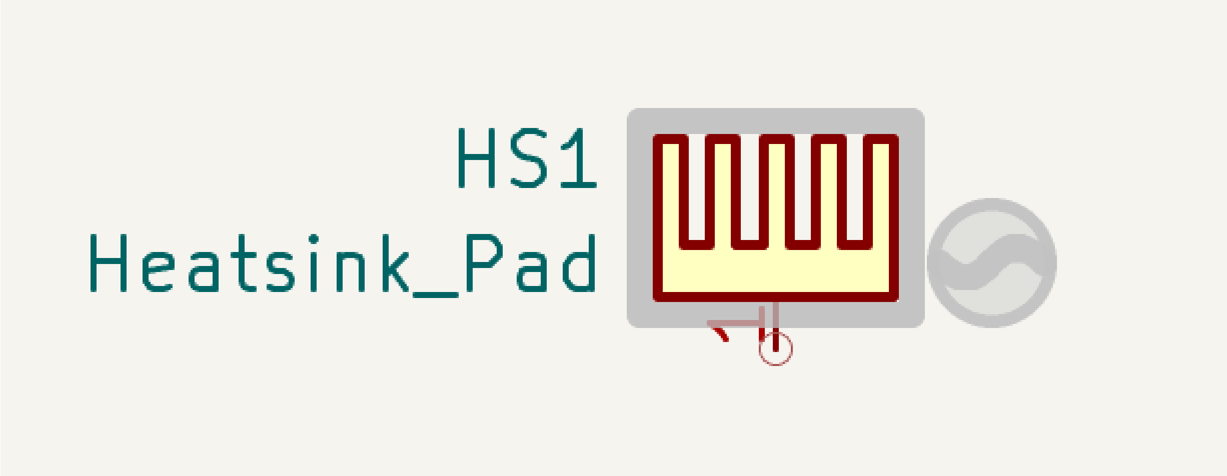
Menerapkan Model

You need to assign simulation models to symbols before you can simulate your circuit.

Each symbol can have only one model assigned, even if the symbol consists of multiple units. For symbols with multiple units, you should assign the model to the first unit.

SPICE model information is stored as text in symbol fields. Therefore you may define it in either the symbol editor or the schematic editor. To assign a simulation model to a symbol, open the Symbol Properties dialog and click the **Simulation Model…** button, which opens the Simulation Model Editor dialog.

You can exclude a symbol from simulation entirely by checking the **exclude from simulation** checkbox in the Symbol Properties dialog. Symbols with this attribute set are drawn with a grey outline and a small simulation icon next to them, as shown below.



Inferred models

Resistor, inductor, and capacitor models can be inferred, which means that KiCad will detect that they are passives and automatically assign an appropriate simulation model. Therefore they do not require any special settings; users only need to set the `Value` field of the symbol.

KiCad infers simulation models for symbols based on the following criteria:

- The symbol has exactly two pins,
- The reference designator begins with `R`, `L` or `C`.

Inferred models are ideal models. If the simulation requires a non-ideal model, for example an inductor with parasitic capacitance included, you must explicitly assign a model that includes it.

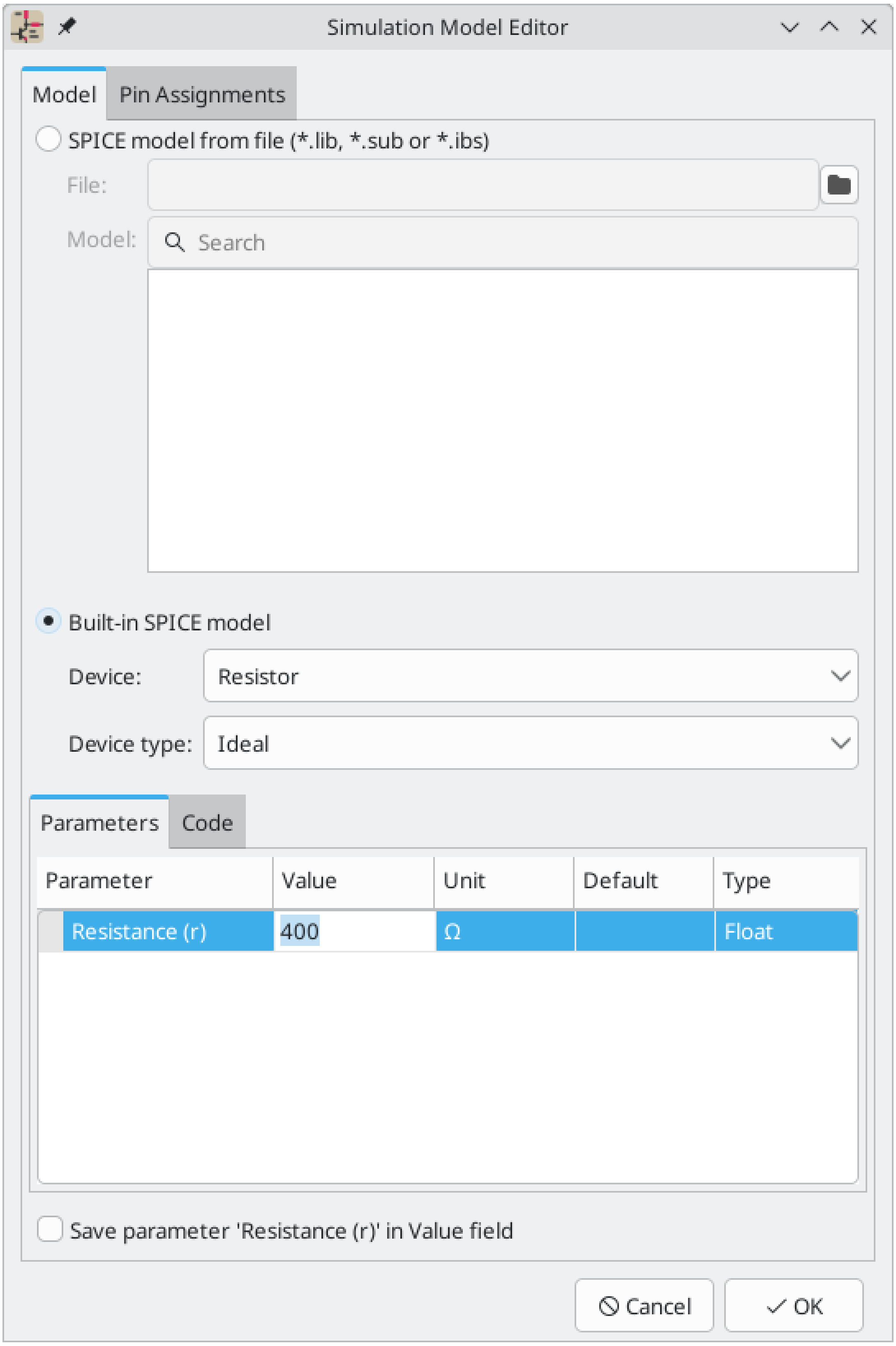
Built-in models

KiCad offers several standard simulation models. They do not require an external model file, and their parameters can be edited in KiCad's Simulation Model Editor GUI. The following devices are available:

- Resistors (including potentiometers)
- Capacitors
- Inductors
- Transmission lines
- Switches
- Voltage and current sources
- Diodes
- Transistors (BJTs, MOSFETs, MESFETs, and JFETs)
- XSPICE code models
- Raw SPICE elements

To add a built-in model to a symbol, open the Simulation Model Editor dialog (**Symbol Properties** → **Simulation Model…**) and select **Built-in SPICE model**. You can then select the kind of device from the **device** dropdown and the device subtype from the **device type** dropdown.

Refer to the ngspice documentation for more details about these models and their parameters.



○ SPICE model from file (*.lib, *.sub or *.ibs)

File:

Model: Search

● Built-in SPICE model

Device: Resistor

Device type: Ideal

Parameter	Value	Unit	Default	Type
Resistance (r)	400	Ω		Float

□ Save parameter 'Resistance (r)' in Value field

Device sets the type of device to simulate: a resistor, BJT, voltage source, etc. This value is stored in the symbol's `Sim.Device` field.

Device type selects the type of model to use for the device. Most devices have several types of models to choose from. Models may vary in their degree of accuracy, which characteristics they are optimized for, what parameters they have available, and how many pins they have. For example, the **ideal** resistor type models a simple resistor with two terminals and a single `resistance` parameter, while the **potentiometer** resistor type models an adjustable resistor with three terminals and an additional parameter for wiper position. Some devices have an especially large number of types to choose from: N-channel MOSFETs, for example, have 17 available types, each of which uses a different mathematical model to simulate the transistor behavior. One model may be more or less appropriate than another for simulating a specific device or circuit or for performing a particular analysis. Refer to the ngspice documentation for detailed information about models and their parameters. The **device type** value is stored in the symbol's `Sim.Type` field.

The **parameters** tab displays the parameters of the model and lets you edit them. For example, a resistor's resistance, a voltage source's waveform, a MOSFET's width and length, etc. Any parameters that differ from the model's defaults are stored in the symbol's `Sim.Params` field.

The **code** tab displays the generated SPICE model as it will be written to the SPICE netlist for simulation.

The **Save parameter '<parameter name>' in Value field** checkbox uses the symbol's `Value` field for storing parameters instead of the `Sim.Params` field. This may make it easier to edit simple models from the schematic, without opening the Simulation Model Editor. This option is only available for ideal passive models (R, L, C) and DC sources. If the field `Sim.Params` exists in the symbol, it will take priority over the `Value` field.

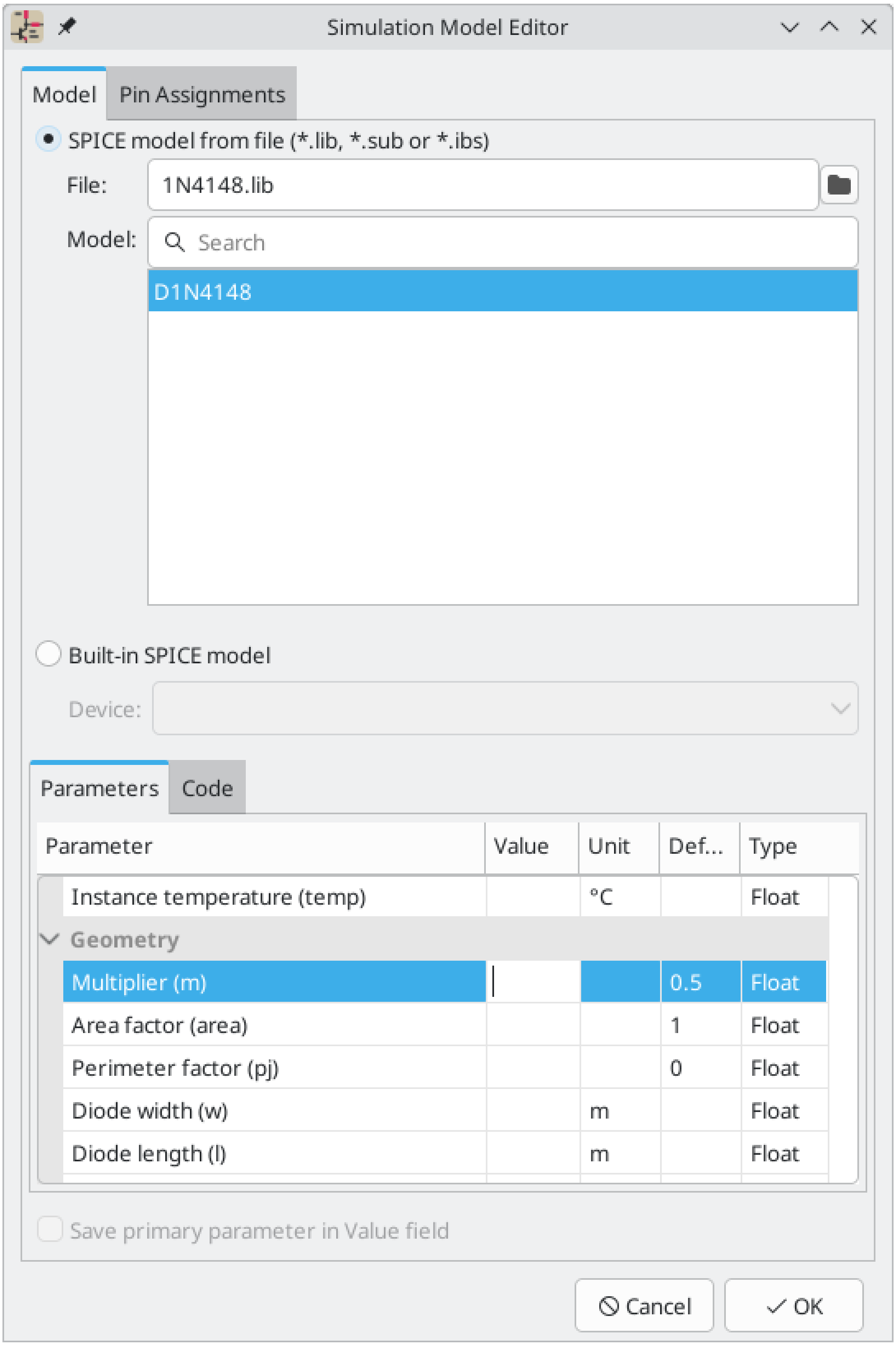
External models

KiCad can also load SPICE models from external files. This is typically how you will add a SPICE model of a specific commercially-available part (for example, a 555 timer or a TL071 operational amplifier) to your simulation. Models such as these are readily available from numerous sources, including manufacturers' web sites. These models must be in a standard SPICE format and must not be encrypted.

An external model can be one of the following types:

- A device model (`.model`). This is an intrinsic device (a passive, diode, transistor, etc.) with a set of parameter values defining its behavior. The parameters for a device model are editable in the Simulation Model Editor GUI.
- A subcircuit model (`.subckt`). This is a model that uses a collection of other ngspice circuit elements to define its behavior. If a subcircuit model contains parameters (in a `params:` sequence in its definition), the parameters are editable in the Simulation Model Editor GUI.

To load a model from an external file, open the Simulation Model Editor dialog (**Symbol Properties** → **Simulation Model...**) and select **SPICE model from file**.



● SPICE model from file (*.lib, *.sub or *.ibs)

File: 1N4148.lib

Model: Search

D1N4148

○ Built-in SPICE model

Device:

Parameter	Value	Unit	Def...	Type
Instance temperature (temp)		°C		Float
Geometry				
Multiplier (m)			0.5	Float
Area factor (area)			1	Float
Perimeter factor (pj)			0	Float
Diode width (w)		m		Float
Diode length (l)		m		Float

□ Save primary parameter in Value field

File is the path to the model file to use. Unencrypted model files are plain, human-readable text files and often have extensions such as `.lib`, `.sub` etc., although KiCad will accept a valid model with any extension.

The path to the file can be absolute, or relative to the project folder. The path can also be relative to the value of `SPICE_LIB_DIR` if you have defined that path variable. If you enable the **Embed File** checkbox in the file browser, the library file will be embedded in the schematic (or symbol library). This makes the schematic (or schematic) more portable as it doesn't rely on an external library file. The library filename is saved in the symbol's `Sim.Library` field.

Model is the name of the desired model in the model file. A model file may contain multiple models, and if so they will all be shown in the list. You can filter the list of models using the search box. The selected model is listed in the symbol's `Sim.Name` field.

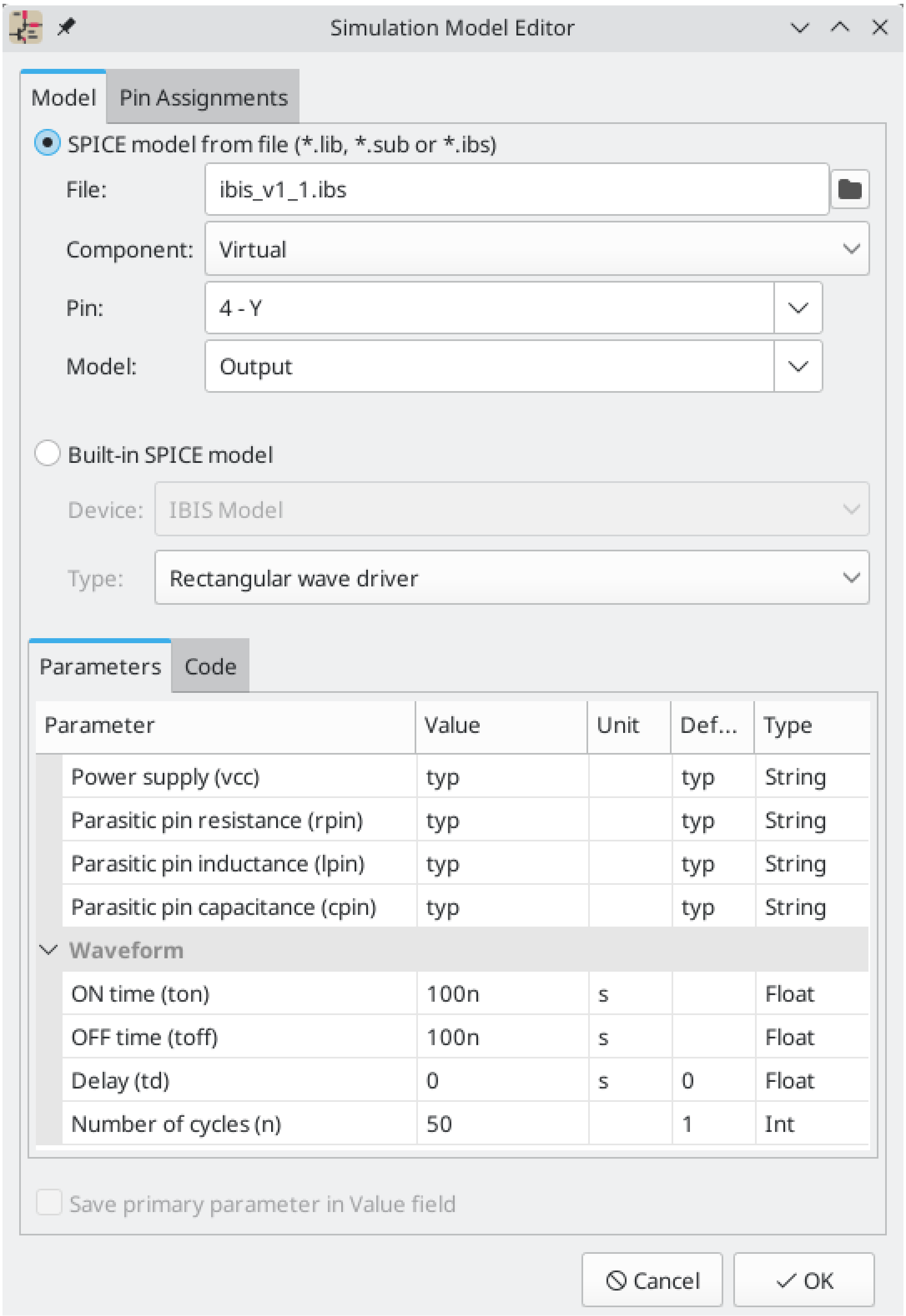
Parameters can be overridden (or additional parameters specified) using the **Parameters** tab. For device models, all parameters for that type of device are editable. For subcircuit models, any parameters included in the subcircuit definition are editable. Any parameters that are overridden in the **Parameters** tab are stored in the symbol's `Sim.Params` field.

The **Code** tab displays the generated SPICE model as it will be written to the SPICE netlist for simulation.

NOTE KiCad is not distributed with SPICE models for specific commercial devices. These models are usually available from device manufacturers or other internet sources.

IBIS models

IBIS (I/O Buffer Information Specification) files are an alternative to SPICE models for modeling the behavior of input/output buffers on digital parts. In order to load an IBIS file, users should follow the procedure for SPICE library models, but provide a `.ibs` file.



● SPICE model from file (*.lib, *.sub or *.ibs)

File: ibis_v1_1.ibs

Component: Virtual

Pin: 4 - Y

Model: Output

○ Built-in SPICE model

Device: IBIS Model

Type: Rectangular wave driver

Parameter	Value	Unit	Def...	Type
Power supply (vcc)	typ		typ	String
Parasitic pin resistance (rpin)	typ		typ	String
Parasitic pin inductance (lpin)	typ		typ	String
Parasitic pin capacitance (cpin)	typ		typ	String
Waveform				
ON time (ton)	100n	s		Float
OFF time (toff)	100n	s		Float
Delay (td)	0	s	0	Float
Number of cycles (n)	50		1	Int

□ Save primary parameter in Value field

File is the path to the model file to use. The path can be absolute or relative to the project folder. The path can also be relative to the value of `SPICE_LIB_DIR` if you have defined that path variable. The library filename is saved in the symbol's `Sim.Library` field. If an IBIS model file is loaded, the remaining fields in the dialog will relate to the IBIS model.

Component selects which component from the IBIS file to use, as IBIS files can contain multiple components. The component name is saved in the symbol's `Sim.Name` field.

Pin selects which pin in the IBIS model to simulate. The selected pin must be mapped to a symbol pin in the **Pin Assignments** tab. The chosen pin's number is saved in the symbol's `Sim.Ibis.Pin` field.

Model is the list of models available for the selected pin, for example an input or an output. The chosen model name is saved in the symbol's `Sim.Ibis.Model` field.

Type selects what the pin should do in the simulation. A pin can be a passive **device** that doesn't drive any value; it can be a **DC driver** that drives high, low, or high-impedance; or it can be a **rectangular wave** or **PRBS driver**. This value is stored in the symbol's `Sim.Type` field.

The **Parameters** tab lets you see and edit the parameters of the model. For each parameter, you can switch between a minimum, typical, or maximum value, as defined in the IBIS file. You can also choose the parameters of the driven waveform, depending on the pin's chosen **type**. Any parameters that differ from the defaults are stored in the symbol's `Sim.Params` field.

NOTE KiCad does not ship with IBIS models for symbols. IBIS models are usually available from device manufacturers.

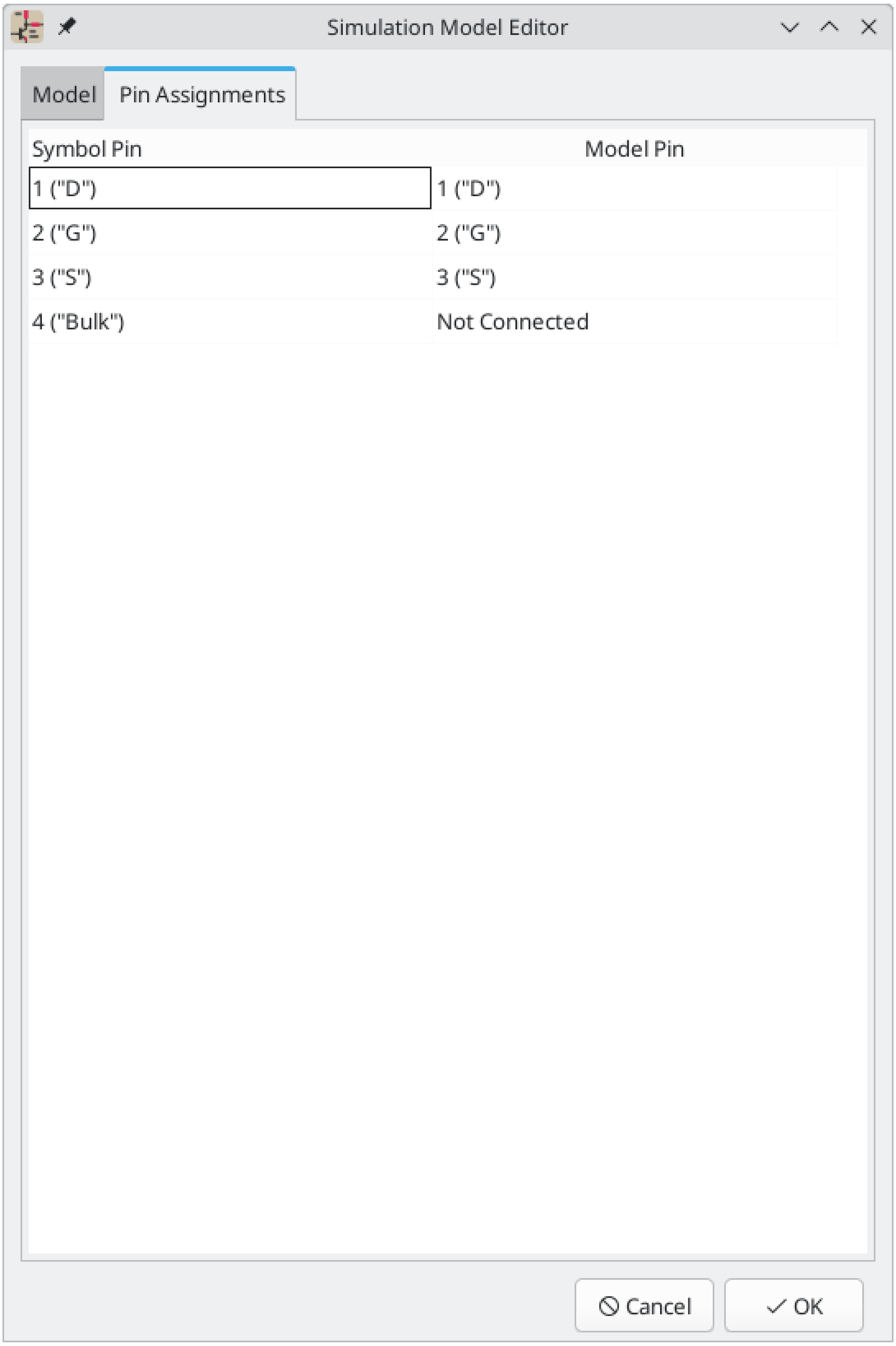
NOTE KiCad's `Simulation_SPICE` symbol library provides several symbols that may be useful for IBIS simulations. `IBIS_DEVICE` can be used for device (input) pins, while `IBIS_DRIVER` can be used for simulating driver pins. There are also variants of each for differential pins.

Pin Assignment

Simulation models may have their pins numbered differently than the corresponding symbol. For example, SPICE models for diodes usually consider pin 1 to be the anode, while schematic symbols are usually drawn with pin 1 as the cathode. Operational amplifier models are also very likely to have model pin assignments that do not match package or schematic pin numbers.

You can use the Simulation Model Editor's **Pin Assignments** tab to map the symbol's pins to the simulation model pins.

NOTE Always make sure symbol pins are correctly mapped to simulation model pins. Mistakes here can lead to erroneous or confusing simulation results, or a failure to simulate at all.



Symbol Pin	Model Pin
1 ("D")	1 ("D")
2 ("G")	2 ("G")
3 ("S")	3 ("S")
4 ("Bulk")	Not Connected

The left column displays the name and number of each symbol pin, i.e. the pin numbers and names that appear on the schematic part in KiCad. The right column displays the corresponding pin as defined in the model file in use. For each symbol pin, you can select the corresponding pin from the simulation model in the dropdown in the right column. In the cases where a schematic part has pins that are not in the model, as in the case of an operational amplifier with 'nulling' pins that are not modeled, the schematic part pin may be assigned to the 'Not Connected' option in the Pin Assignments dropdown. Unlike other pin assignments, 'Not Connected' may be assigned to multiple pins if necessary.

When you use a subcircuit model, the dialog displays the model's code under the pin assignments for use as a reference while assigning pins. A well-written model will often include a helpful reference section (as a set of comments) to inform the user how the model pins are mapped.

Value notation

The simulator supports several notations for writing numerical values in simulation model parameters, simulation analysis setup options, and SPICE directives:

- Plain notation : `10100`, `0.003`,
- Scientific notation: `1.01e4`, `3e-3`,
- Prefix notation: `10.1k`, `3m`.
- RKM notation: `4k7`, `10R`.

You can mix prefix and scientific notations. As such, `3e-4k` is a valid input and is equivalent to `0.3`. The list of valid prefixes is shown below. They are case sensitive.

Prefix	Name	Multiplier
a	atto	10^{-18}
f	femto	10^{-15}
p	pico	10^{-12}
n	nano	10^{-9}
u	micro	10^{-6}
m	milli	10^{-3}
k	kilo	10^{3}
M	mega	10^{6}
G	giga	10^{9}
T	tera	10^{12}
P	peta	10^{15}
E	exa	10^{15}

NOTE

Raw SPICE Element models and directives are passed to ngspice directly, without KiCad reformatting the values for ngspice to consume. ngspice uses a different, case-insensitive notation: 1 mega (10^{6}) is denoted there as `1Meg`, while `1M` is 1 milli (10^{-3}). Depending on the compatibility mode selected, ngspice may not support the same value notations as KiCad, so care should be taken when using raw SPICE elements and simulation directives.

SPICE directives

It is possible to add SPICE directives by placing them in text fields on a schematic sheet. This approach is convenient for defining the default simulation type. The list of supported directives in text fields is:

- Directives starting with a dot (e.g. `.tran 10n 1m`)
- Coupling coefficients for inductors (e.g. `K1 L1 L2 0.89`)

It is not possible to place additional components using text fields.

If a simulation command is included in schematic text, the simulator will use it as the simulation command when you open the simulator. However, you can override it by choosing a different simulation type in the Simulation Command dialog. You can also specify simulation commands with raw SPICE commands by using the Custom analysis type.

Refer to the ngspice documentation for more details about SPICE directives.

Running simulations

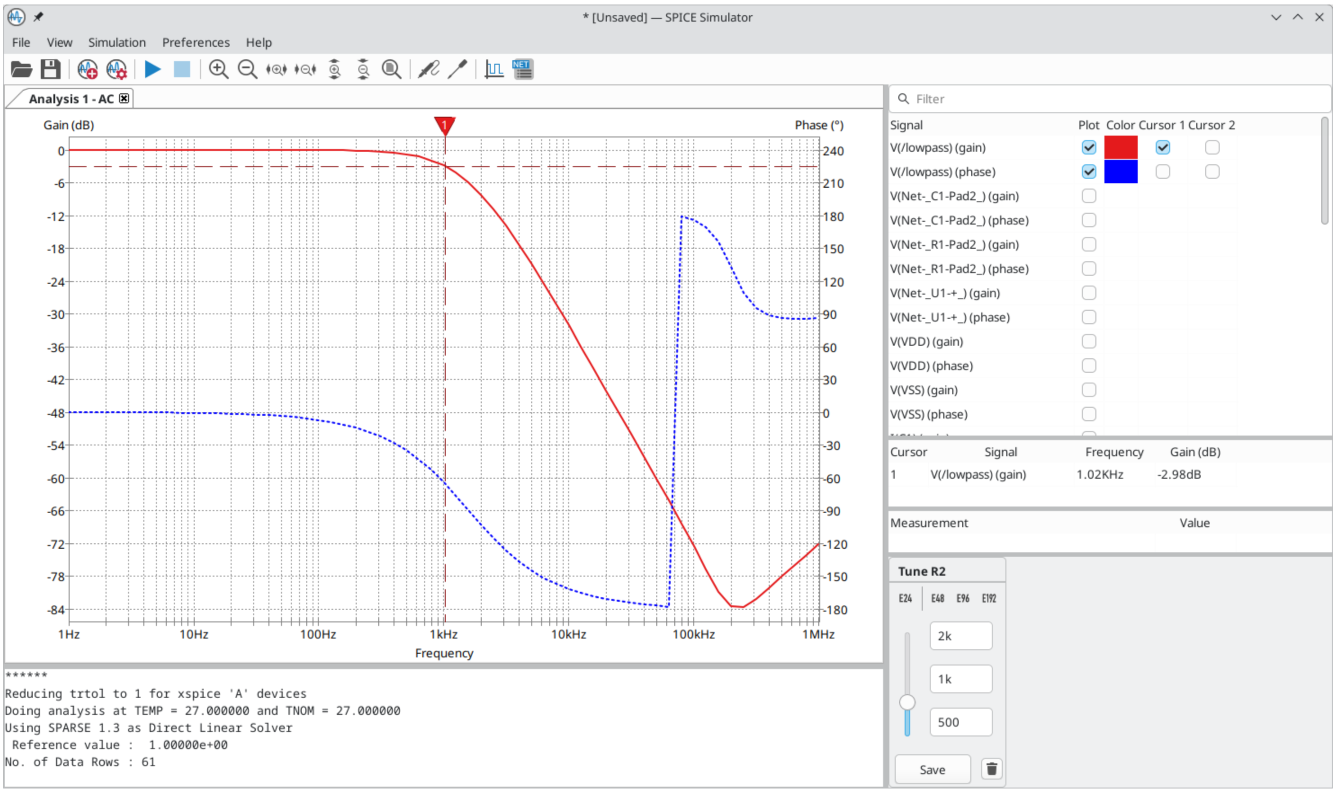
Circuits for simulation are drawn in the Schematic Editor, but simulations are run in the Simulator window.

After creating a schematic for simulation, the following steps are needed to run a simulation:

1. Open the simulator window by clicking **Inspect** → **Simulator** in the Schematic Editor or using the button in the top toolbar.
2. Create at least one analysis by clicking **Simulation** → **New Analysis Tab...** (`Ctrl` + `N`) or using the button. This lets you choose a type of simulation and configure its options. These options are explained below. Each analysis has its own tab, and multiple analyses can be created. The options for an analysis can be adjusted after it is created with **Simulation** → **Edit Analysis Tab...** or by clicking on the button.
3. Run the simulation by clicking **Simulation** → **Run Simulation** (`R`) or by clicking the ▶ button. This only runs the simulation in the active analysis tab. You can stop a running simulation at its current point by clicking the ■ button.
4. Examine the simulation results. Most analyses result in plots; for these you will need to select the signals to be plotted. Other analyses print their results in the output log window.

NOTE Once a simulation has been set up, the configuration can be saved in a workbook.

Simulator window



```
******
Reducing trtol to 1 for xspice 'A' devices
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
Using SPARSE 1.3 as Direct Linear Solver
 Reference value :  1.00000e+00
No. of Data Rows : 61
```

Signal	Plot	Color	Cursor 1	Cursor 2
V(/lowpass) (gain)	☑	🟥	☑	□
V(/lowpass) (phase)	☑	🟦	□	□
V(Net-_C1-Pad2_) (gain)	□			
V(Net-_C1-Pad2_) (phase)	□			
V(Net-_R1-Pad2_) (gain)	□			
V(Net-_R1-Pad2_) (phase)	□			
V(Net-_U1-+_) (gain)	□			
V(Net-_U1-+_) (phase)	□			
V(VDD) (gain)	□			
V(VDD) (phase)	□			
V(VSS) (gain)	□			
V(VSS) (phase)	□			

Cursor	Signal	Frequency	Gain (dB)
1	V(/lowpass) (gain)	1.02KHz	-2.98dB

Measurement	Value

The simulator window is divided into several sections:

- The top of the window has a toolbar with buttons for commonly used actions.
- The main part of the window graphically shows the simulation results. The simulation needs to run and signals need to be selected from the list of available signals or probed before they are displayed in the plot.
- Below the plot panel, the output log window shows logs from the ngspice simulation engine. Some types of analyses print their results here.
- The right side of the window displays a list of signals, a list of active cursors, measurements, and a tuning tool for adjusting component values based on simulation results.

Simulation types

Each simulation is a specific type of analysis. The following analysis types are available:

- **OP** — DC Operating Point
- **DC** — DC Sweep Analysis
- **AC** — AC Small-signal Analysis
- **TRAN** — Transient Analysis
- **PZ** — Pole-zero Analysis
- **NOISE** — Noise Analysis
- **SP** — S-parameter Analysis

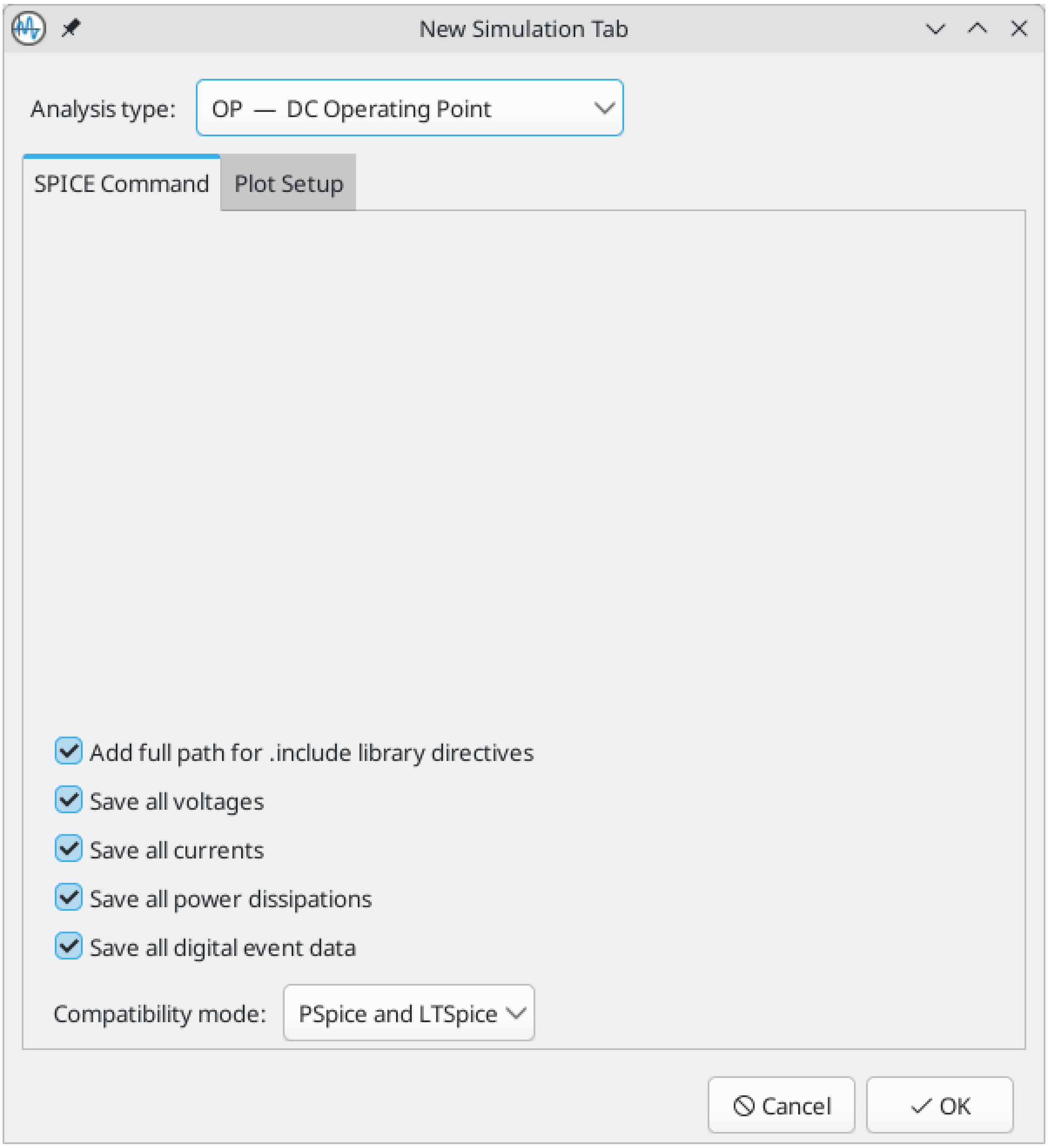
FFT — Frequency-content Analysis

- **Custom** — Custom SPICE Analysis (raw SPICE command)

Each analysis type and its options are explained below. You can configure an analysis when you create it (button) or by editing an existing analysis (button). These analysis types are explained in more detail in the ngspice documentation.

> **NOTE** Another way to configure a simulation is to type SPICE directives into text fields on schematics. Any text field directives related to a simulation command are overridden by the settings selected in the dialog. This means that once a simulation has run, the dialog overrides the schematic directives until the simulator is reopened.

Operating point analysis (OP)



Analysis type: OP — DC Operating Point

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

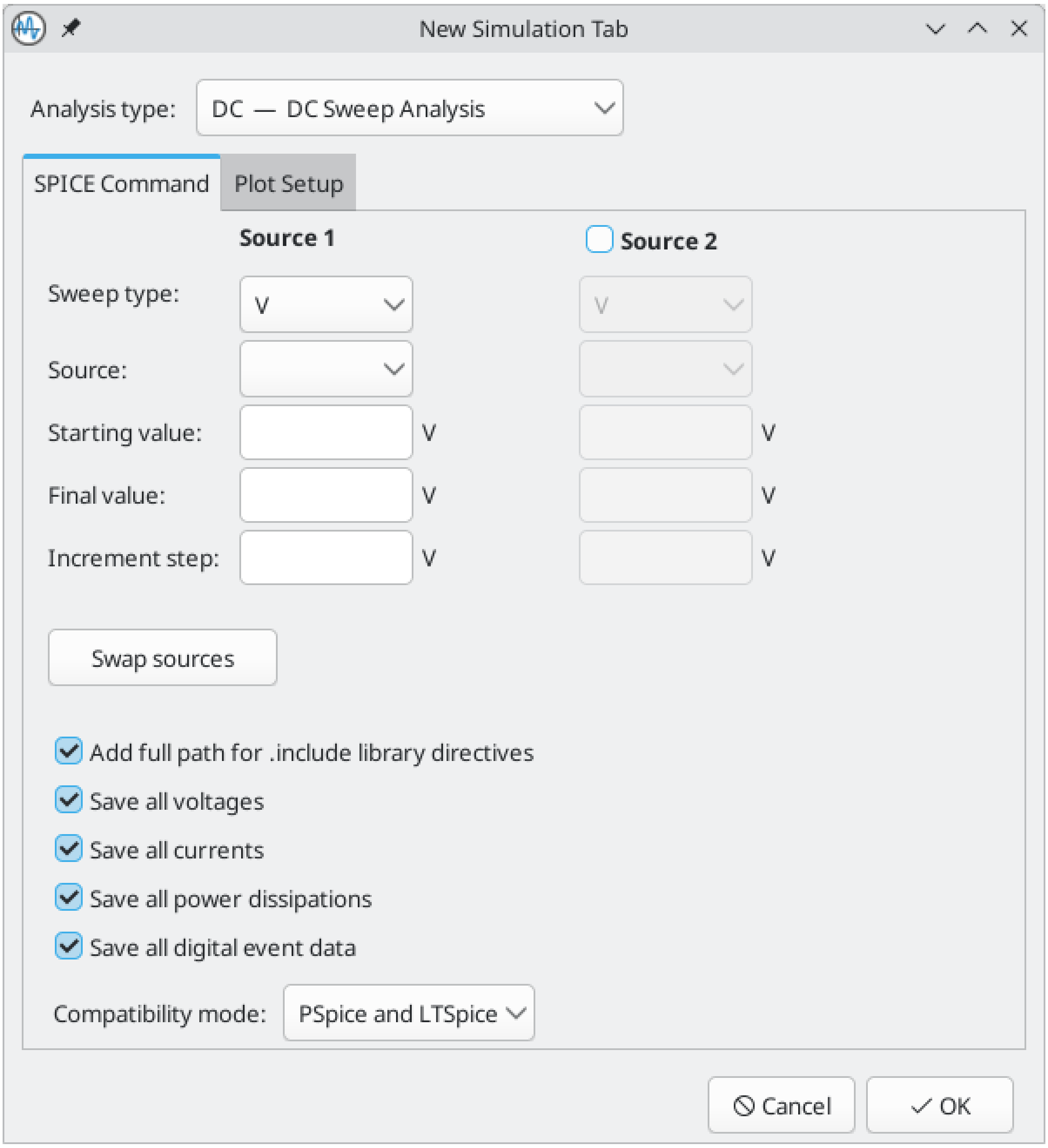
☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the DC operating point of the circuit. This analysis has no options.

Operating point analyses do not have any plotted results. Results are printed in the output log window. You can also display the node voltages and device currents calculated by this analysis as annotations in the schematic.

DC sweep analysis (DC)



Analysis type: DC — DC Sweep Analysis

Source 1 | □ Source 2

Sweep type: V | V

Source:

Starting value: V | V

Final value: V | V

Increment step: V | V

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the DC behavior of the circuit while sweeping one or two parameters. The following parameters can be swept:

- value of an independent voltage source
- value of an independent current source
- value of a resistor
- simulation temperature

DC analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Sweep type**: the type of variable to sweep. This can be a voltage source, a current source, a resistor, or the simulation temperature.
- **Source**: the particular voltage source, current source, or resistor to sweep (`srcnam`). The list is populated with each item in the schematic of the relevant type. It is disabled for temperature sweeps.
- **Starting value**: the starting value for the sweep (`vstart`).
- **Final value**: the ending value for the sweep (`vstop`).
- **Increment step**: the amount to increase the value at each step of the sweep (`vincr`). Smaller increments result in more output points.

If **Source 2** is enabled, the same options are available to simultaneously sweep a second source. When sweeping two sources, Source 1 is swept over its entire range for each value of Source 2. In other words, each value for Source 2 results in a separate curve, and each curve shows a full sweep of Source 1 with Source 2 held constant at a particular value.

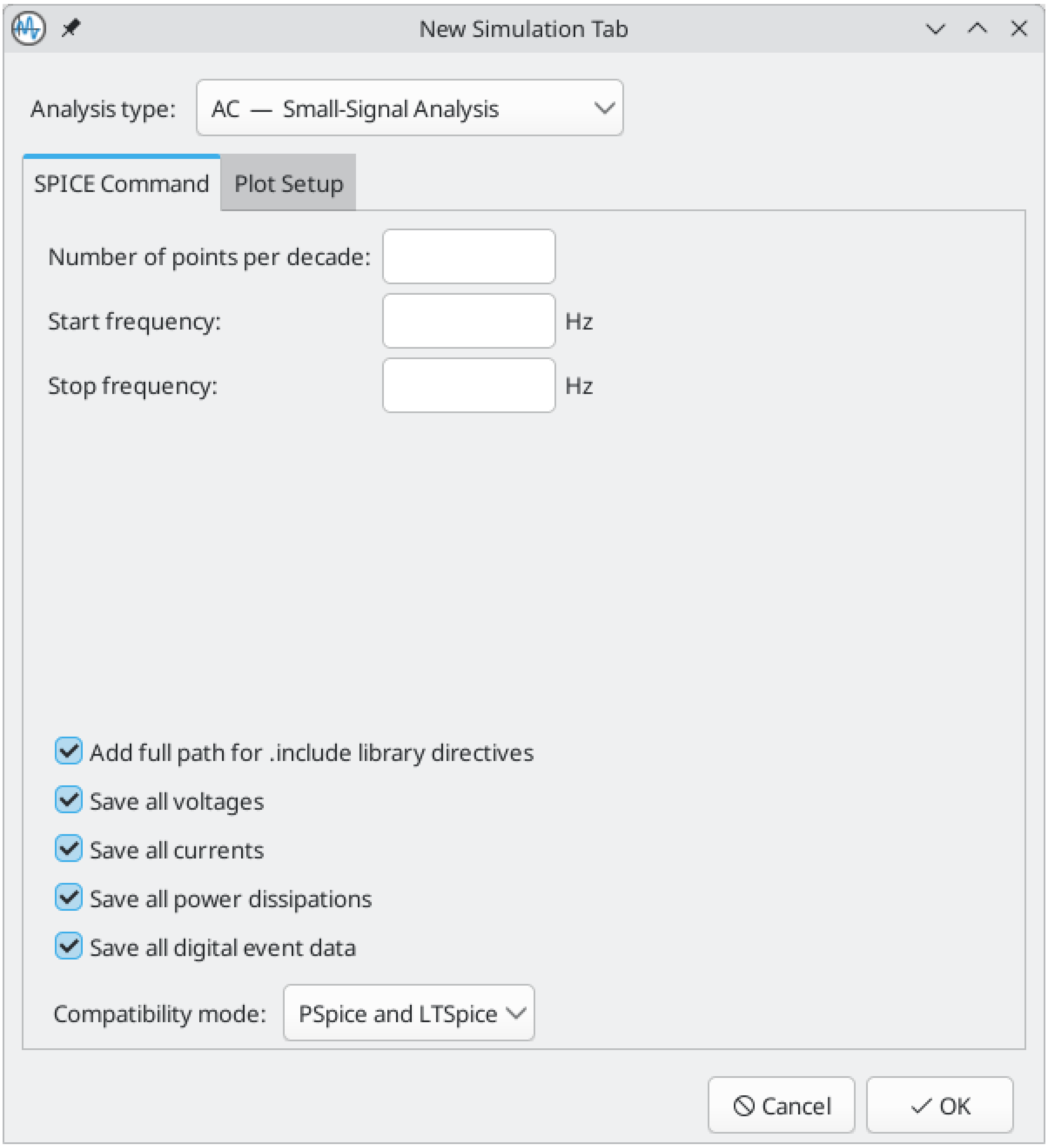
Clicking the **Swap sources** buttons swaps Source 1 with Source 2.

The output is displayed as a plot.

TIP

The tuning tool is another way to see how component values affect circuit behavior. Unlike DC sweeps, the tuning tool can be used with other types of simulations, like transient or AC analyses.

AC small-signal analysis (AC)



Analysis type: AC — Small-Signal Analysis

Number of points per decade:

Start frequency: Hz

Stop frequency: Hz

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the small-signal AC behavior of the circuit in response to a stimulus. Performs a decade sweep of stimulus frequency.

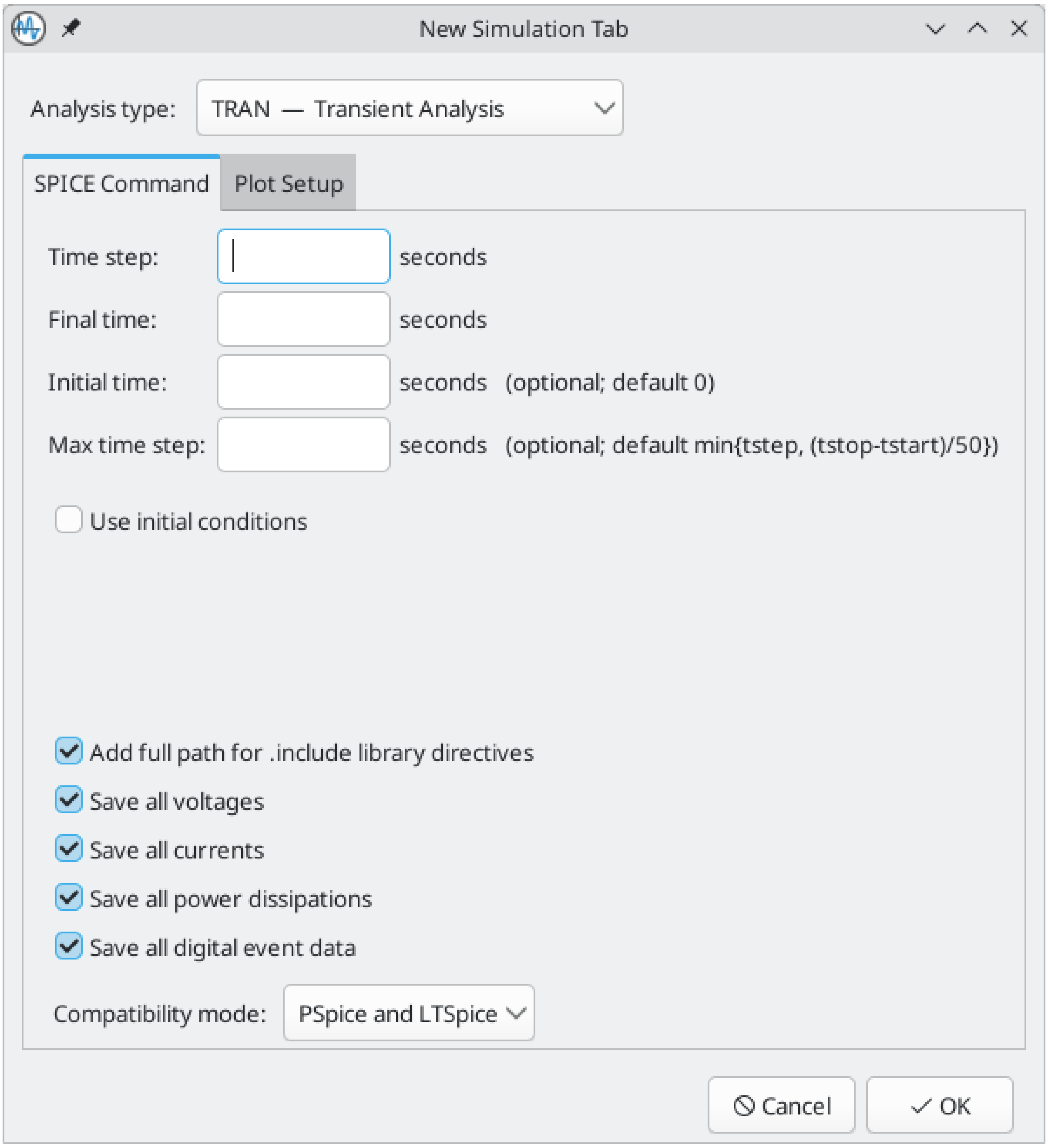
To run an AC analysis you must choose a number of points to measure per decade and the start and end frequencies for the decade sweep.

AC analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Number of points per decade**: the number of points to calculate per decade (`nd`).
- **Start frequency**: the lower bound of the frequency range to analyze (`fstart`).
- **Stop frequency**: the upper bound of the frequency range to analyze (`fstop`).

The output is displayed as a Bode plot (output magnitude and phase vs. frequency).

Transient analysis (TRAN)



Analysis type: TRAN — Transient Analysis

Time step: seconds

Final time: seconds

Initial time: seconds (optional; default 0)

Max time step: seconds (optional; default min{tstep, (tstop-tstart)/50})

□ Use initial conditions

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the time-varying behavior of the circuit.

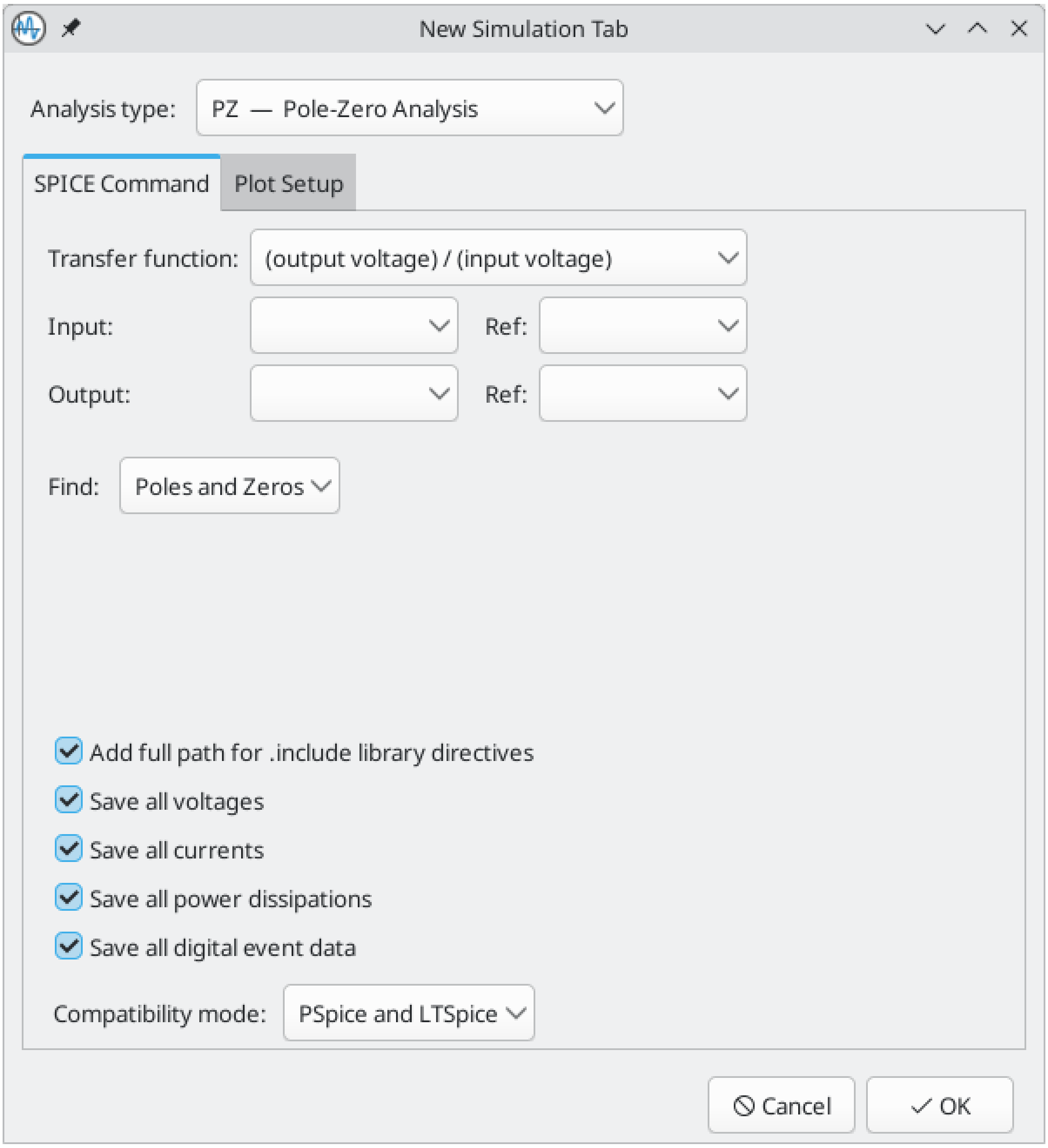
Transient analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Time step**: a suggested time step (`tstep`).
- **Final time**: the time at which the simulation will end (`tstop`).
- **Initial time**: the time at which the simulation will start (`tstart`). If not specified, this defaults to 0.
- **Max time step**: the maximum time step (`tmax`). If not specified, this defaults to the suggested time step or the total simulation duration divided by 50, whichever is smaller.
- **Use initial conditions**: if enabled, the simulator will not calculate the quiescent operating point before starting the transient simulation. Instead, the simulation will use the initial conditions specified in a `.ic`

directive and element `IC` parameters. This corresponds to ngspice's `uic` option.

The output is displayed as a plot.

Pole-zero analysis (PZ)



New Simulation Tab

Analysis type: PZ — Pole-Zero Analysis

Transfer function: (output voltage) / (input voltage)

Input: Ref:

Output: Ref:

Find: Poles and Zeros

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the poles and zeroes of the small-signal (AC) transfer function of the circuit.

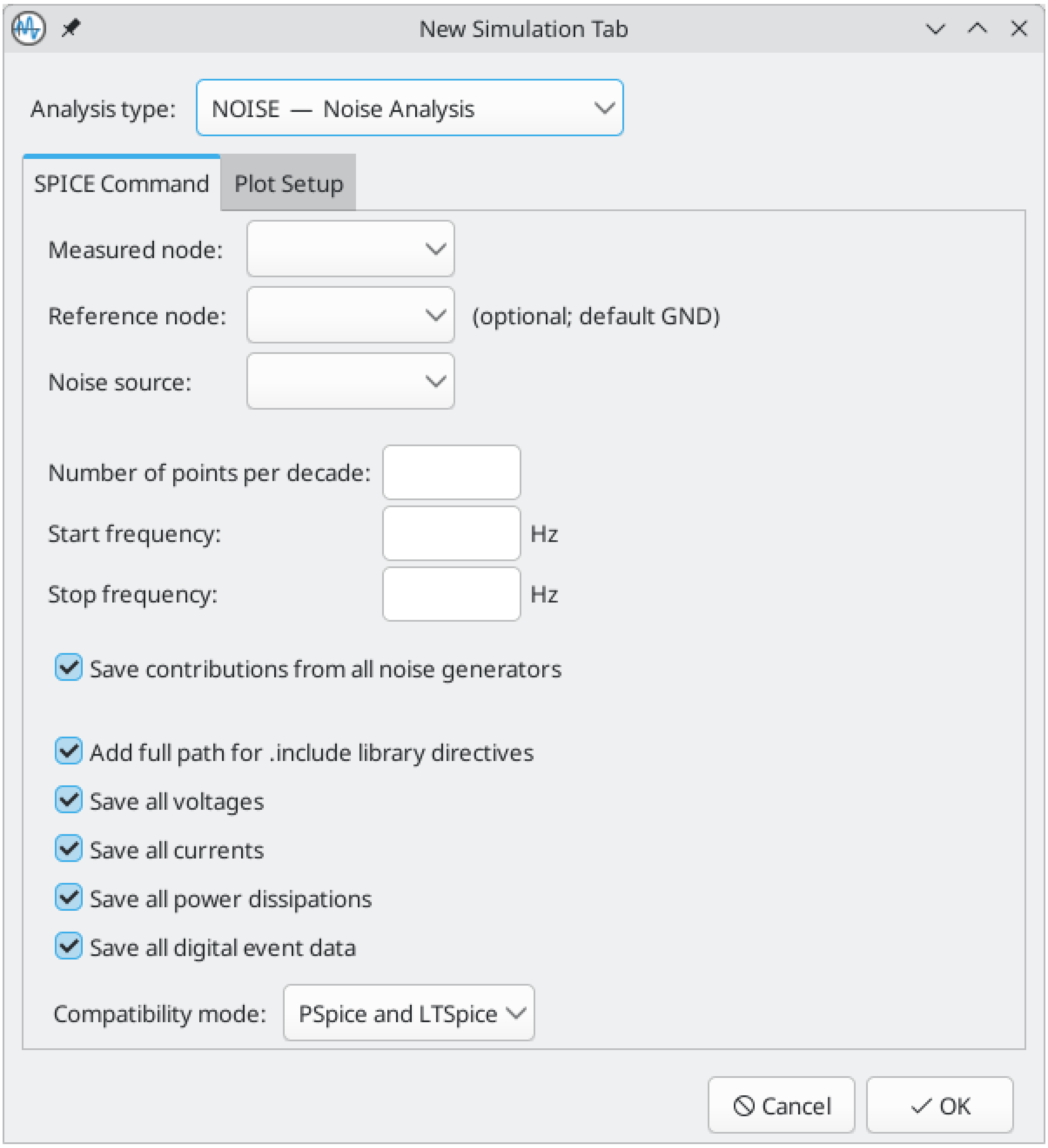
Pole-zero analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Transfer function**: selects between calculating the transfer function as output voltage divided by input voltage or output voltage divided by input current. These correspond to ngspice's `vol` and `cur` options, respectively.
- **Input**: the input node (`node1`) and the reference node for the input (`node2`).
- **Output**: the output node (`node3`) and the reference node for the output (`node4`).

Find: selects between calculating poles and zeroes, only poles, or only zeroes. These correspond to ngspice's `pz`, `pol`, and `zer` options, respectively.

Operating point analyses do not have any plotted results. Results are printed in the output log window.

Noise analysis (NOISE)



Analysis type: NOISE — Noise Analysis

Measured node:

Reference node: (optional; default GND)

Noise source:

Number of points per decade:

Start frequency: Hz

Stop frequency: Hz

☑ Save contributions from all noise generators

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Calculates the noise generated by the devices in the circuit, both as output noise and input-referred noise. The noise spectrum (V/√Hz or A/√Hz) of the circuit can be plotted, and the total noise over the specified frequency range is reported. Optionally, the noise contributions of each device are reported individually.

Noise analyses have the following options, which are listed here with the corresponding ngspice parameter name:

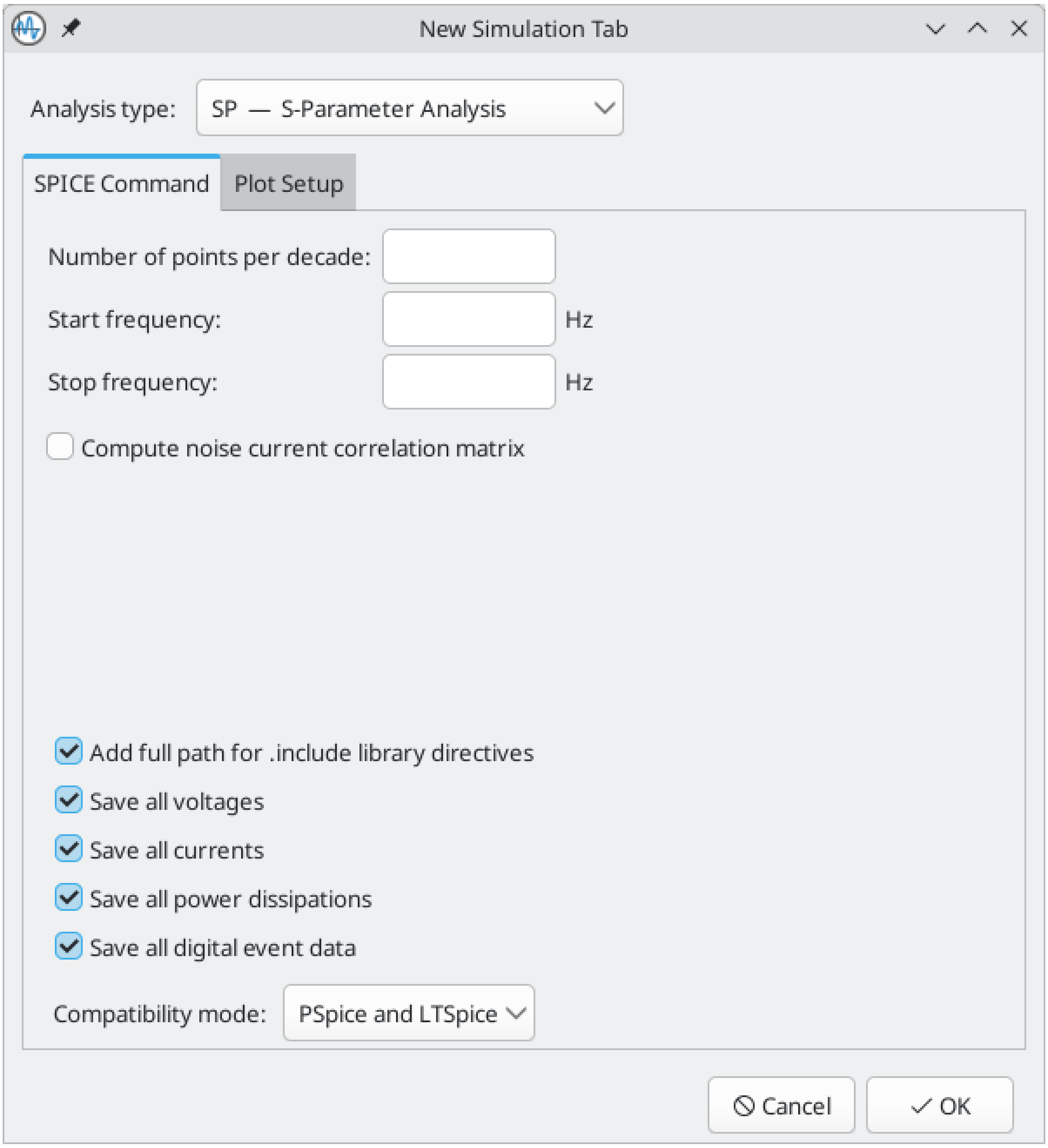
- **Measured node**: the node at which output noise is measured (`output`).
- **Reference node**: the reference node for measuring the output noise (`ref`). The output noise is calculated as the noise at the measured node minus the noise at the reference node. If it is not specified,

the default is the ground node.

- **Noise source**: a source that is considered the circuit’s input for the purposes of calculating input-referred noise (`src`). The source must specify an AC magnitude, i.e. the source must have an `ac` parameter.
- **Number of points per decade**: the number of points to calculate per decade (`pts`).
- **Start frequency**: the lower bound of the frequency range to analyze (`fstart`).
- **Stop frequency**: the upper bound of the frequency range to analyze (`fstop`).
- **Save contributions from all noise generators**: if enabled, noise magnitudes of individual noise generators are saved and reported. If disabled, only the overall output and input-referred noise over the specified frequency range are reported.

The output and input-referred noise spectra can be plotted. Total noise values for the specified frequency range are printed in the output log window.

S-parameter analysis (SP)



Analysis type: SP — S-Parameter Analysis

Number of points per decade:

Start frequency: Hz

Stop frequency: Hz

□ Compute noise current correlation matrix

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

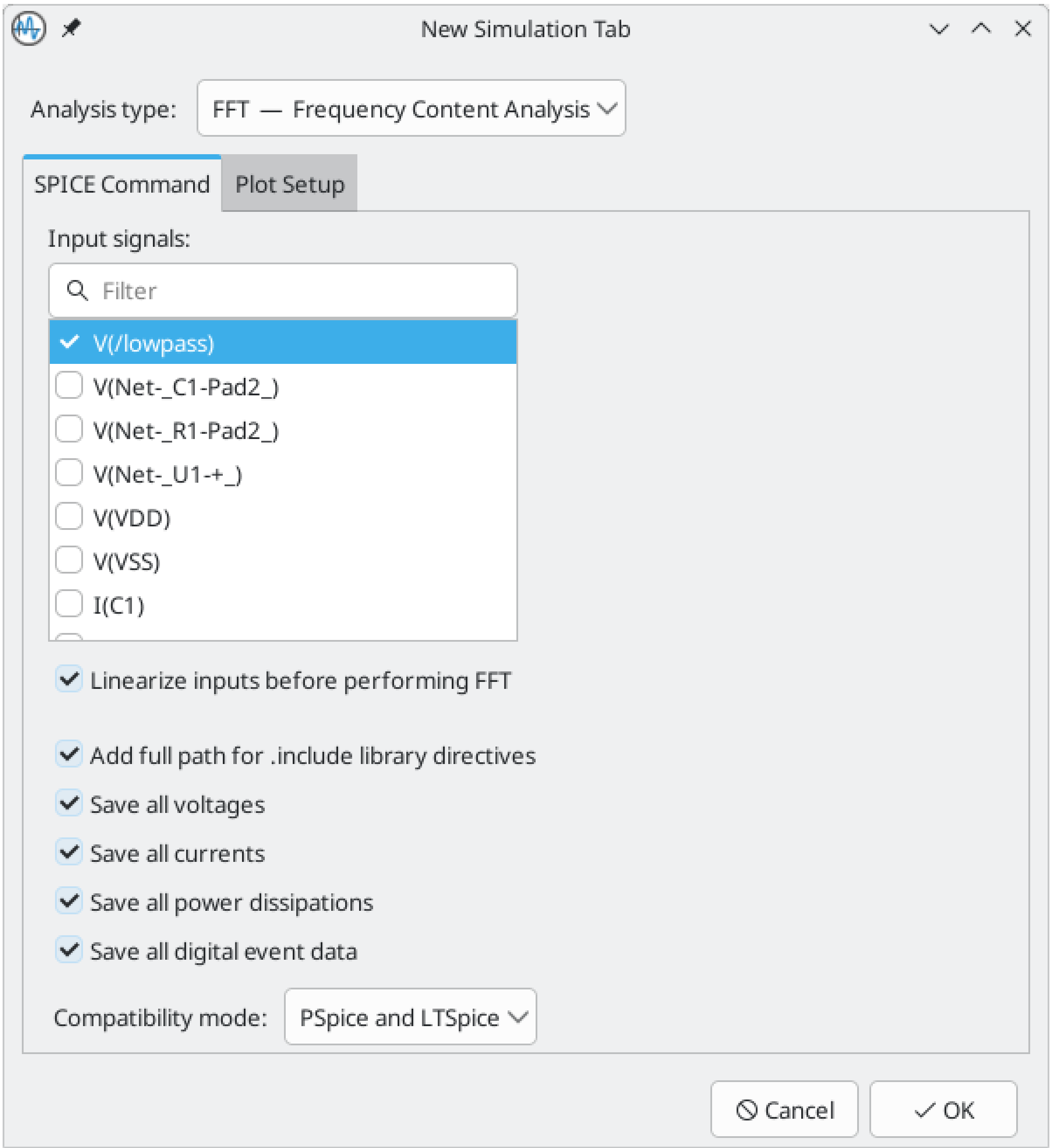
Calculates the scattering parameters, admittance matrix, and impedance matrix for the circuit. Optionally calculates the noise current correlation matrix. Note that this analysis requires at least two voltage sources configured as RF ports as described in the ngspice manual.

S-parameter analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Number of points per decade**: the number of points to calculate per decade (`nd`).
- **Start frequency**: the lower bound of the frequency range to analyze (`fstart`).
- **Stop frequency**: the upper bound of the frequency range to analyze (`fstop`).
- **Compute noise current correlation matrix**: if enabled, the noise current correlation matrix is also calculated. This corresponds to ngspice's `donoise` option.

The output is displayed as a plot.

Frequency content analysis (FFT)



Analysis type: FFT — Frequency Content Analysis

Input signals:

☑ V(/lowpass)
□ V(Net-_C1-Pad2_)
□ V(Net-_R1-Pad2_)
□ V(Net-_U1-+_)
□ V(VDD)
□ V(VSS)
□ I(C1)

☑ Linearize inputs before performing FFT

☑ Add full path for .include library directives
☑ Save all voltages
☑ Save all currents
☑ Save all power dissipations
☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

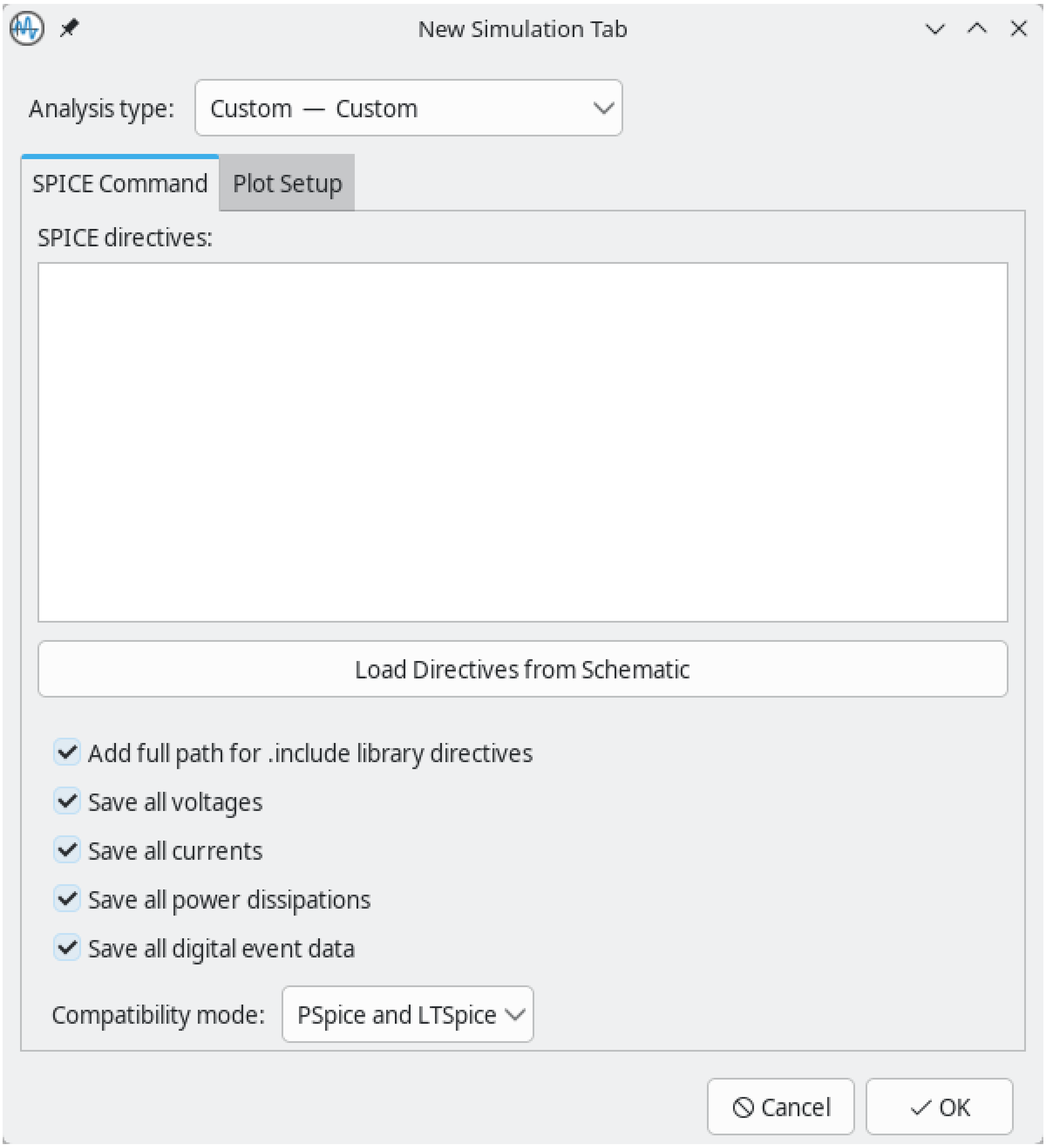
Calculates an FFT based on an existing analysis tab. Any of the signals from an existing analysis can be used as inputs to the FFT. The signals are windowed using a Hanning window.

FFT analyses have the following options, which are listed here with the corresponding ngspice parameter name:

- **Input signals**: the signal(s) to perform an FFT on. An FFT is independently applied to each selected signal.
- **Linearize inputs before performing FFT**: When enabled, the input vector is converted to have equidistant time points prior to performing the FFT. Corresponds to ngspice's `linearize` command.

The output is displayed as a plot.

Custom analysis



Analysis type: Custom — Custom

SPICE directives:

☑ Add full path for .include library directives

☑ Save all voltages

☑ Save all currents

☑ Save all power dissipations

☑ Save all digital event data

Compatibility mode: PSpice and LTSpice

Performs a SPICE analysis defined by raw SPICE analysis commands (`.tran`, `.ac`, `.dc`, `.noise`, etc.). For details on such analysis commands and their syntax, refer to the ngspice documentation.

The analysis commands given on each line of the **SPICE directives** textbox are run as separate analyses. If you click the **Load Directives from Schematic** button, any SPICE directives in schematic text are copied into the textbox.

Additional simulation settings

There are several simulation options that apply to all types of simulations. These are located at the bottom of the dialog.

- **Add full path for .include library directives**: if enabled, relative paths to SPICE models will be converted to absolute paths in the SPICE netlist.
- **Save all voltages**: if enabled, the simulator will save voltages for each node in the simulation results so that they can be plotted. This corresponds to ngspice's `.save all` command in the SPICE netlist. If disabled, voltages will not be saved in the results, and therefore cannot be plotted, unless a SPICE directive is used to manually probe a node voltage.
- **Save all currents**: if enabled, the simulator will save currents through each device pin in the simulation results so that they can be plotted. This corresponds to ngspice's `.probe alli` command in the SPICE netlist. If disabled, currents will not be saved in the results, and therefore cannot be plotted, unless a SPICE directive is used to manually probe a current.
- **Save all power dissipations**: if enabled, the simulator will save power dissipations for each device in the simulation results so that they can be plotted. This corresponds to ngspice's `.probe P(<device>)` command in the SPICE netlist for each device in the schematic. If disabled, power dissipations will not be saved in the results, and therefore cannot be plotted, unless a SPICE directive is used to manually probe a power dissipation.
- **Save all digital event data**: if enabled, the simulator will save digital event data for digital (event-driven) simulation models. This corresponds to ngspice's `.esave all` command. If disabled, digital event data will be discarded.
- The **Compatibility mode** dropdown selects the compatibility mode that the simulator uses to load models. The **User configuration** option refers to the user's `.spiceinit` ngspice configuration file. Compatibility modes are described in the ngspice documentation.

Viewing simulation results

Each analysis has its own tab, containing its own separate plot, signal list, and output log window. Only the active tab is updated when a simulation is run. In this way it is possible to compare simulation results between different runs.

Simulation results from most analysis types are visualized as plots. However, DC Operating Point (OP) and Pole-zero (PZ) analyses do not generate plots. Instead, they print their results in the output log at the bottom of the simulator window. OP analysis results can additionally be displayed as annotations on the schematic canvas.

Plotted results

Most types of analyses display their results in a plot. The type of plot depends on the analysis: transient simulations display signal values over time, for example, while AC simulations display results in a Bode plot.

You can zoom and move a plot using the following gestures:

- Scroll mouse wheel to zoom in/out. `Shift`, `Ctrl`, and `Alt` can modify the scroll action depending on the configuration in the Simulator panel in the Schematic Editor Preferences.
- Right click to open a context menu to adjust the view.
- Draw a selection rectangle to zoom in the selected area.
- Drag a cursor marker to move the cursor.

The list of signals that can be plotted in the active plot is shown in the Signals pane on the right side of the simulator window. The following types of signals can be plotted:

- Node voltages: the voltage of each net in the schematic, displayed as `V(<net>)`.
- Device node currents: the current for each device in the schematic, displayed as `I(<device>)` or `I(<device:terminal>)`. For two-terminal devices, the device's current is listed as a single signal corresponding to the current into the device's pin 1. For devices with more than two terminals, the current into each terminal is a separate signal.
- Device power dissipations: the power dissipated by each component, displayed as `P(<device>)`.
- User-defined signals: custom signals defined as an expression based on other signals. User-defined signals can be arbitrary mathematical expressions. One common use for user-defined signals is to plot a voltage differential, i.e. the voltage between two points.

To plot a signal, check the box in the plot column next to the signal of interest. To remove a signal from the plot, clear its checkbox.

You can also interactively select signals to plot by using the Probe Schematic tool. To activate the tool, use **Simulation** → **Probe Schematic...** (`P`) or click the ( ) button. When activated, the tool lets you click elements in the schematic to plot the corresponding signal. Different types of signals are probed depending on what you click:

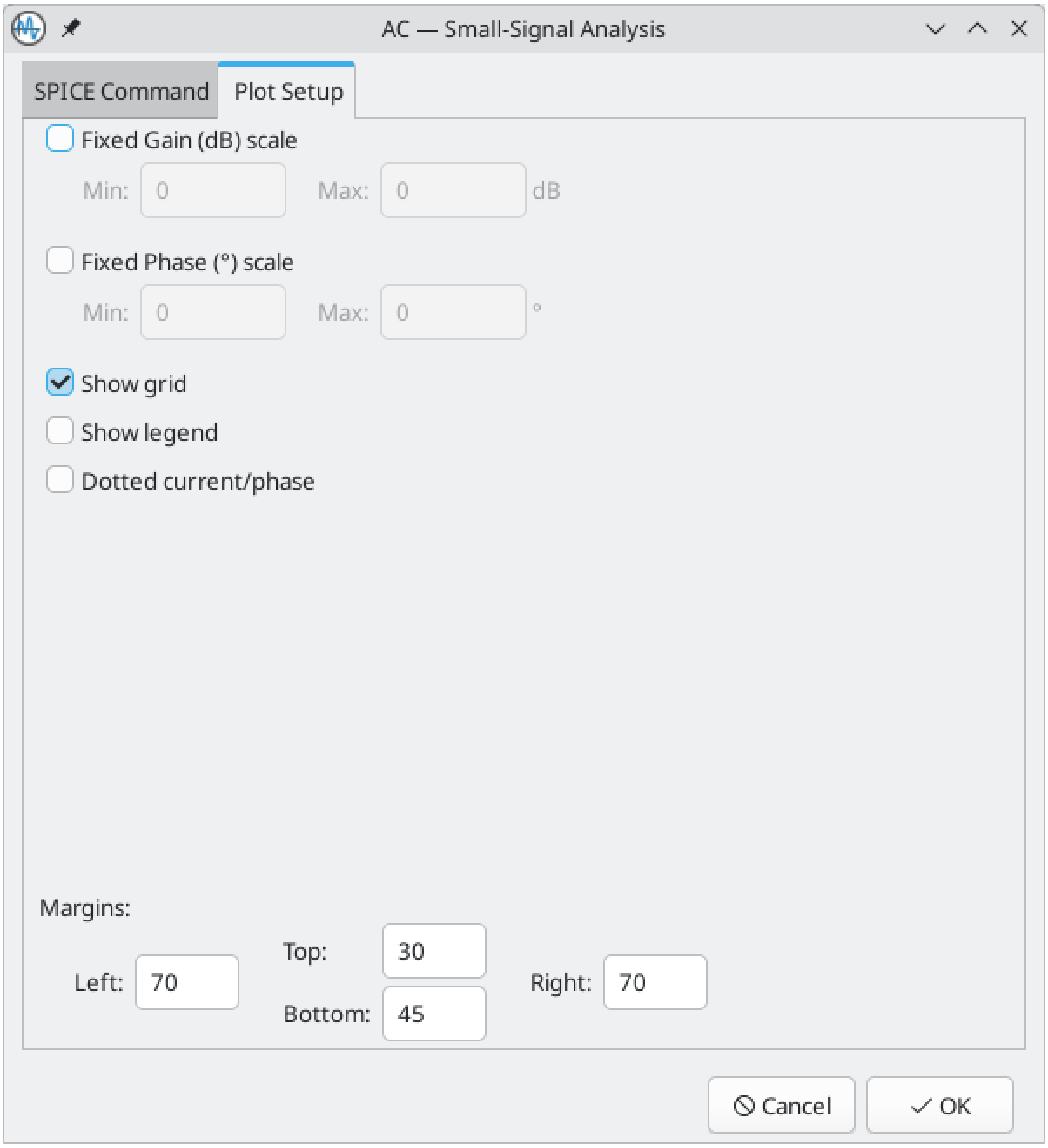
- Clicking on a wire plots the voltage of that net. When you hover over a wire, the net is highlighted to indicate that its voltage can be probed.
- Clicking on a symbol pin plots the current going into that pin. When you hover over a pin, the cursor changes to a current clamp to indicate that clicking will probe the current.

> **NOTE** KiCad does not support ngspice's `.plot` directive. This directive has no effect when a simulation is run using KiCad.

Plot settings and appearance

Colors of individual signals may be set by clicking the color field associated with each signal in the Signals grid. You may choose from a predefined palette (**Defined Colors**), or select a custom color (**Color Picker**). You can toggle the color of the plot background from black to white with **View** → **Dark Mode Plots** (this affects all analysis tabs, not just the active tab).

Many plot settings can be configured in the **Plot Setup** tab of the Analysis Setup window (button).



☐ Fixed Gain (dB) scale

Min: 0 Max: 0 dB

☐ Fixed Phase (°) scale

Min: 0 Max: 0 °

☑ Show grid

☐ Show legend

☐ Dotted current/phase

Margins:

Left: 70 Top: 30 Bottom: 45 Right: 70

- **Fixed scale**: when enabled, this fixes the vertical and/or horizontal plot scales to the specified ranges. Manually zooming will not affect an axis if its scale is fixed.
- **Show grid**: when enabled, a grid will be shown behind the plotted signals. You can also turn the grid on or off with **View** → **Show Grid**.
- **Show legend**: when enabled, a legend will be shown on the plot for the enabled signals. You can reposition the legend by dragging it.
- **Dotted current/phase**: when enabled, plotted signals representing current (transient simulations) or phase (AC simulations) are displayed using dotted lines instead of solid lines. You can also enable this setting with **View** → **Dotted Current/Phase**.
- **Margins**: this controls the padding on each side of the plot.

Cursors

For precise measurement, cursors are available in the plot window. You can add a cursor to a signal by checking the **Cursor 1** or **Cursor 2** checkbox for the signal in the signal grid on the right.

Once cursors have been added, the horizontal position of each cursor is shown by a number in a triangular marker at the top of the plot display. You can reposition each cursor by clicking and dragging its marker. The vertical position of each cursor tracks its assigned signal. The horizontal and vertical value for each cursor are shown in the cursor grid on the right of the simulator window. If cursors 1 and 2 are both enabled, the difference between them is also shown.

To precisely position a cursor, you can directly edit its horizontal position in the cursors grid. You can also modify the display format (unit range and number of significant digits) by right clicking a value in the cursors grid and clicking **Format** for the relevant quantity.

You can also add more cursors beyond the two default cursors. To create more cursors, right click on a signal in the signals grid and click **Create new cursor**. To remove an extra cursor, right click in a one of the extra cursor columns and click **Delete cursor 3** (or whichever cursor you want to delete).

Measurements

You can add automatically calculated measurements for any plotted signal, such as a minimum, average, or peak-to-peak measurement.

NOTE	Measurements are made over all the data resulting from the simulation, not just the data that is visible in the plot window at the current zoom setting.

NOTE	It is not necessary to have selected a signal for plotting (by selecting its Plot checkbox) in order to make a measurement on it. Even unplotted signals can be measured.

To add a predefined measurement to a signal, right click on a signal in the signals grid and select a measurement. The following measurements are available:

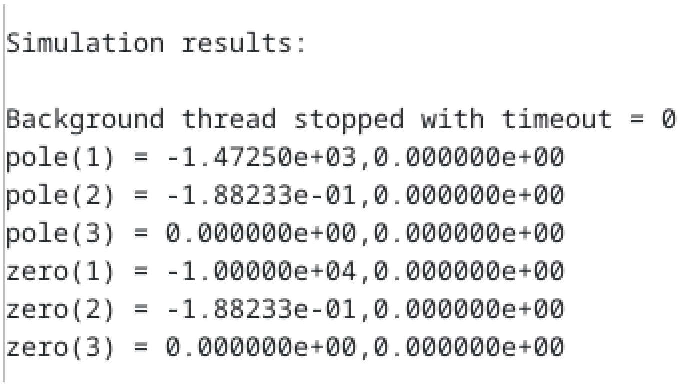
- **Measure Min**: measures the minimum value of the entire signal
- **Measure Max**: measures the minimum value of the entire signal
- **Measure RMS**: measures the root-mean-square value of the entire signal
- **Measure Peak-to-peak**: measures the peak-to-peak value of the entire signal
- **Measure Time of Min**: measures the time at which the minimum value of the entire signal occurs
- **Measure Time of Max**: measures the time at which the maximum value of the entire signal occurs
- **Measure Integral**: computes the time integral value of the entire signal
- **Perform Fourier Analysis**: performs a Fourier analysis of the selected signal based on a specified fundamental frequency. Calculates the amplitude of the harmonics of the fundamental as well as the total harmonic distortion. This measurement is only available for transient analyses, and its results are printed in the simulation log window instead of in the measurement panel.

Measurement results are displayed in the Measurement pane at the lower right of the simulator window. Multiple measurements will display as multiple rows in this area. You can delete a measurement by right clicking it and clicking **Delete Measurement**. To modify the display format of a measurement result (unit range and number of significant digits), right click the value and click **Format Value…**.

The above context menu options are a shortcut for directly specifying measurements in the the Measurement pane. To manually create a new measurement, click in an empty row in the Measurement pane and type in an ngspice measurement function. You can also edit existing measurements by clicking on them. For more information about measurements and their syntax, refer to the ngspice manual.

Numerical results

While most analyses display their results as plots, DC Operating Point (OP) and Pole-zero (PZ) analyses do not result in plots. Instead, these analyses print their results as text in the simulation log window.

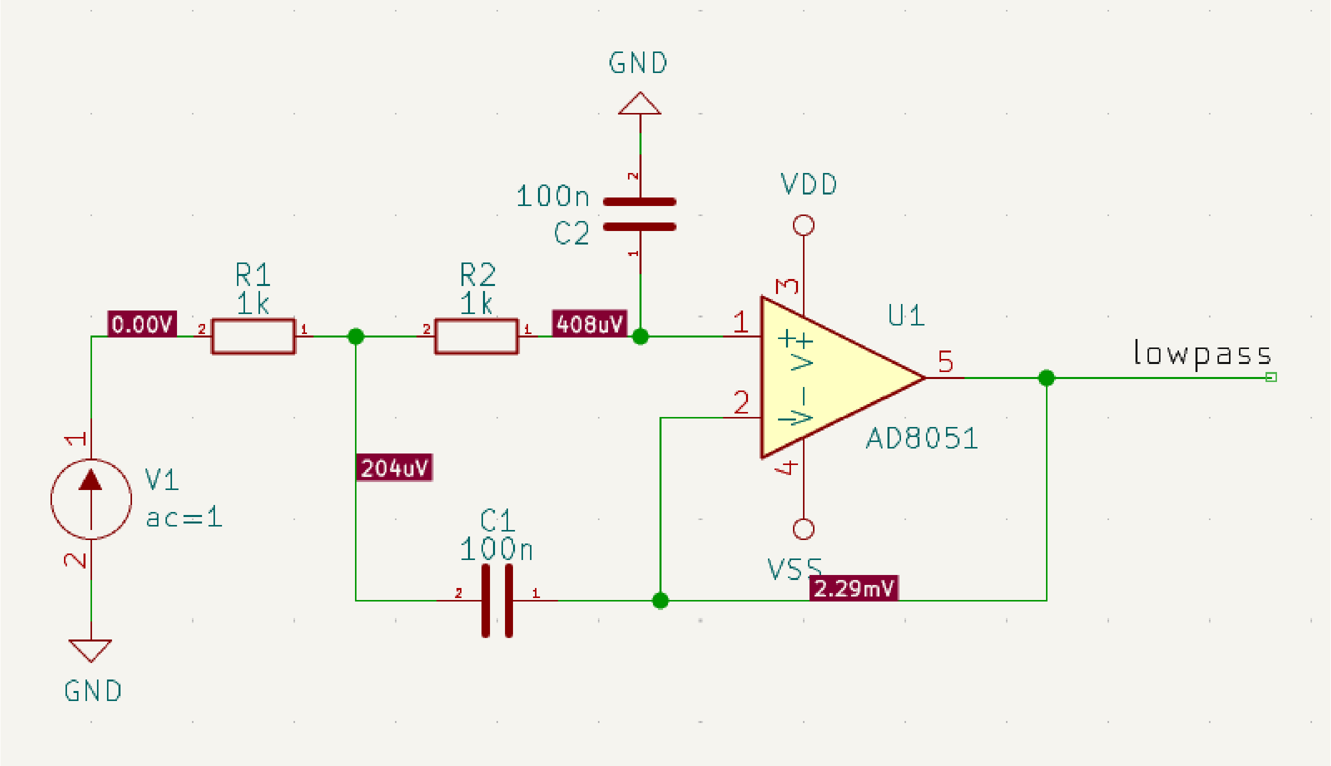


```
Simulation results:

Background thread stopped with timeout = 0
pole(1) = -1.47250e+03,0.000000e+00
pole(2) = -1.88233e-01,0.000000e+00
pole(3) = 0.000000e+00,0.000000e+00
zero(1) = -1.00000e+04,0.000000e+00
zero(2) = -1.88233e-01,0.000000e+00
zero(3) = 0.000000e+00,0.000000e+00
```

Operating point annotations

For the OP analysis, annotations can also be added to the schematic indicating the operating point voltage and current values at the nodes. Operating point voltage annotations can be shown or hidden for every node with **View → Show OP Voltages**. Operating point current annotations can be shown or hidden for every symbol pin with **View → Show OP Currents**. These annotations are globally shown or hidden for every node or every pin. You can control the formatting of these annotations in the Formatting panel of Schematic Setup.



You can also display operating point results using text variables. Using text variables requires more work to set up but provides more control over how the data is displayed. For example, you can use text variables to display only certain voltages or currents, or to control the display formatting of particular values. You can also use text variables to display power dissipation measurements.

To use text variables to display an operating point voltage for a net, add a label to the net, then add a field to that label. The label can contain any text as long as it contains the `${OP}` text variable. The text variable can contain formatting specifiers in the form `${OP.<precision><unit>}`, but both the precision and unit are optional. Precision is the number of significant digits to display, which is 3 by default. Unit is the unit to use, including a prefix, such as `mV` for millivolts.

As an example, a label field containing the text `Vout: ${OP}`, attached to a net with an operating point voltage of 5.123V, displays as "Vout: 5.12V". The text `${OP.4mV}` displays as "5123mV", `${OP.2}` displays as "5.1V", and `${OP.mV}` displays as "512mV".

Using text variables to display an operating point current into a device pin is similar, but uses symbol fields instead of label fields. The field can contain any text as long as it contains the `${OP:<pin>}` text variable. The pin can be specified as a pin name or a pin number. For two-terminal devices, the pin can be omitted, and the current into the device's pin 1 will be reported. The same optional formatting specifiers are supported as for voltages, in the form `${OP:<pin>.<precision><unit>}`.

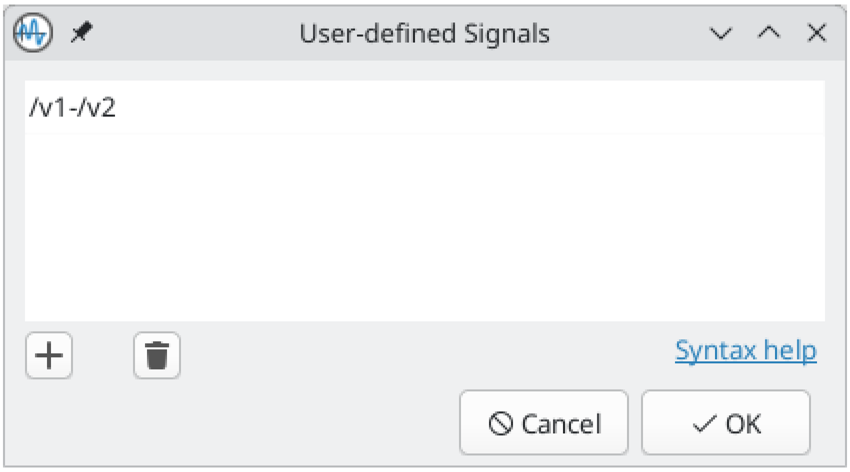
For example, in a symbol with an operating point current of 5.123mA through pin 1 (pin name `C`), a symbol field containing the text `Ic: ${OP:C}` displays as "Ic: 5.12mA". The text `${OP:C.2mA}` displays as "5.1mA", `${OP:1.4}` displays as "5.123mA", and `${OP:C.A}` displays as "0.00512A".

You can also use text variables to display a device's power dissipation. This works identically to current, but with `power` instead of a pin name or number. For example, in a symbol with an operating point power dissipation of 5.123mW, a symbol field containing `${OP:power.2mW}` displays as "5.1mW".

NOTE Don't forget to set the label field or symbol field to visible, or it will not be displayed.

User-defined signals

In addition to the list of signals that come from the nets in the schematic, you can define your own signals which behave like normal signals in most respects. User defined signals are defined as mathematical operations on one or more basic signals.



/v1-/v2

To add a user-defined signal, open the User-defined Signal dialog with **Simulation** → **User-defined Signals...** (button) and add a signal. The new signal will appear in the Signal grid, where it can be plotted and used like any other signal. User-defined signals are also included in OP results.

> **NOTE** Unlike normal signals, which are plotted incrementally as a simulation progresses, user-defined signals are not plotted until the simulation completes.

One use for user-defined signals is to plot a differential voltage, i.e. the voltage between two arbitrary nodes. For example, to plot the voltage difference between the nets `/v1` and `/v2`, you can create a user-defined signal with the expression `/v1-/v2` and then plot it. This expression could also be written in a number of different ways, for example `V(/v1)-V(/v2)` or `V(/v1,/v2)`, which are both equivalent.

A number of mathematical functions are available for use in user-defined signals. To see a list, click the **Syntax help** link in the User-defined Signals dialog. Refer to the ngspice manual for details on each of these functions.

> **NOTE** Net names may contain special characters that have particular meaning to the ngspice interpreter. In particular, ngspice interprets the dash character (`-`) as a subtraction operation. To ensure that net names are interpreted correctly, surround the net name with double quote (`"`) characters when referring to it in a user-defined signal.

Tuning components

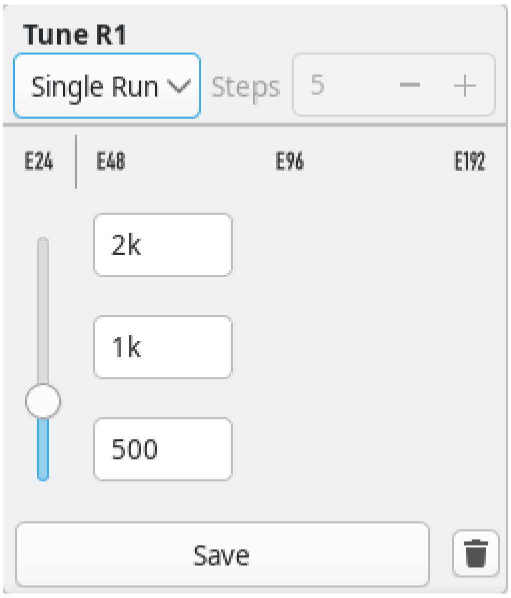
The simulator has a tuning tool, which lets you observe how changing a component value affects the simulation results. Each time the component value is tuned, the simulation is re-run with the new component value. You can then save the tuned component value back to the schematic if desired.

The following components can be tuned:

- Resistors
- Inductors
- Capacitors
- DC Voltage sources
- DC Current sources

> **NOTE** You can only tune components when running analyses that result in a plot. In other words, you cannot tune components when running an operating point analysis.

To tune a component, use **Simulation** → **Add Tuned Value...**, the keyboard shortcut `T`, or the button in the toolbar, and then click on the component to tune in the schematic. This adds a tuning control for that component in the bottom right corner of the simulator window. Multiple components can be tuned at once, with a separate tuning control per component.



Tune R1

Single Run Steps 5

2k

1k

500

- The dropdown menu at the top switches between **Single Run** and **Multi Run** modes.
  - In Single Run mode, the simulation is run using the tuned value from the middle text field. Each time this value changes, the simulation is re-run with the new value, and the new results replace the old results.
  - In Multi Run mode, the simulation is run multiple times using different values for the tuned value. All runs are plotted simultaneously to show the spread of results. The **Steps** control sets the number of runs. The minimum is two runs, as the minimum and maximum values of the tuning range are always included.
- The top text field sets the top end of the tuning range. In Multi Run mode, this sets the value for the last simulation run.
- The bottom text field sets the bottom end of the tuning range. In Multi Run mode, this sets the value for the first simulation run.
- The middle text field sets the actual component value that is used in the simulation in Single Run mode. This value can also be adjusted using the slider. In Multi Run mode, the middle text field is disabled.
- The **Save** button updates the schematic symbol with the tuned value. Until you press the **Save** button, the schematic symbol will keep its original value. In Multi Run mode, the **Save** button is disabled because there is no single value to save back to the schematic.
- The 🗑 button removes the component from the Tune panel and restores its original value.

In addition, it is possible to restrict the component values to those from a particular series of Preferred Values — either of the E24, E48, E96 or E192 series. This is particularly useful when it is necessary to restrict component values to commercially available parts.

Saving simulation setups

You can save a simulation setup in a workbook. Workbooks are files that store information about a simulation setup, including which analyses are configured and what their settings are, which signals are plotted for each analysis, user-defined signals, measurements, cursors, and display settings. A workbook can be saved and reloaded later to restore the previously configured settings in a new session. Workbooks can include more than one simulation: for example, it may contain separate tabs for transient, operating point, and AC analyses which you added as you worked.

You can save a workbook using **File** → **Save Workbook** and load one using **File** → **Open Workbook**. The most recently used workbook is automatically loaded when the simulator is opened.

NOTE Workbooks store simulation setup information, but they do not store simulation results. You can export simulation results to PNG (graphics) or CSV (simulation data values) with **File** → **Export Current Plot as PNG…** and **File** → **Export Current Plot as CSV…**. To recreate the results of simulations stored in a workbook, select the appropriate analysis tab in the Simulator window and run the simulation again (R or **Simulation** → **Run Simulation**).

Exporting simulation results

KiCad's simulator offers two ways to export results:

- as a PNG (Portable Network Graphics format) image of a simulation data plot,
- as a plain text file of simulation data values, in CSV (Comma-Separated Value) format.

Only simulations that produce plots (see above) can be exported as an image or a CSV file. The results of OP and PZ analyses cannot be exported in this way.

Exporting results as graphics

The currently-visible Simulator plot may be exported as a PNG file using the command **File** → **Export Current Plot as PNG…**.

The size and aspect-ratio of the saved image will match that of the displayed plot in the Simulator.

You can also copy an image of the active plot to the clipboard using **File** → **Export Current Plot to Clipboard**, or insert an image of the active plot into the schematic using **File** → **Export Current Plot to Schematic**. Exporting a plot to the schematic is equivalent to exporting the plot to the clipboard and then pasting it into the schematic.

Exporting results as numerical data

The currently-visible Simulator plot may be exported as a CSV file using the command **File** → **Export Current Plot as CSV…**.

The data in the simulation plot is exported as multiple columns. The precise format is dependent upon the analysis type. In general, there will be multiple columns of data, one corresponding to each variable selected for plotting. The first row of the file is a header row, containing the name of the variable in the column (i.e. 'time', 'V(/Vout)1' or similar).

NOTE When exporting to CSV, only variables selected using their Plot checkbox will be included in the exported file.

NOTE The data in the exported file contains all the data that would be plotted if the entire plot were displayed. If the plot is zoomed in to show a particular region this will still be the case, resulting in the output file containing more data than the user might expect.

NOTE

This function exports the data with data separated by semicolons (`;`) rather than commas. Programs reading this data may need to be configured to expect a semicolon as a delimiter.

Troubleshooting simulations

Sometimes a simulation will fail, either with or without errors being reported. Paying attention to the error messages reported, taking care in the development and entry into KiCad of the circuit to be simulated, and making use of the KiCad, ngspice and general SPICE documentation and information from fellow users in forums is very worthwhile and can often point the way to a solution.

It’s worth noting, for users unfamiliar with SPICE, ngspice or circuit simulation in general that some 'common' and interesting circuits can sometimes be tricky to simulate accurately or reliably. These include apparently simple circuits such as oscillators, which in some cases may fail to oscillate at all! It is nearly always possible to build a working simulation but sometimes this can more require SPICE experience than might be initially apparent, or the guidance of someone who already has it. The ngspice documentation, once again, is worth reading for insights into good and effective simulation practices if you are encountering difficulty.

Incorrect netlist

It is possible to inspect the SPICE netlist with **Simulation** → **Show SPICE netlist…** (NET button). This method of troubleshooting requires some SPICE knowledge, but spotting errors in the netlist can help determine the cause of simulation problems, as well as providing confirmation of what input ngspice is actually acting on.

Simulation error messages

The output console displays messages from the simulator. It is advisable to check the console output to verify there are no errors or warnings. Messages appearing in the console may be conveniently selected, copied, and pasted if you wish to share them.

A common error message is "timestep too small". This message means that the simulation engine is unable to calculate the next point in the simulation, even when using the minimum possible time increment. This error can have many causes, including numerical convergence issues with a simulation model used in the circuit or with the circuit itself. It can also be caused by mistakes in drawing the circuit, such as incorrectly assigning pins to the simulation model or forgetting to provide a voltage supply.

Convergence problems

In case the simulation does not converge in a reasonable amount of time (or not at all), it is possible to add the following SPICE directives. More information is available in the ngspice manual.

WARNING

Changing convergence options can lead to erroneous results. These options should be used with caution.

```
.options gmin=1e-10
.options abstol=1e-10
.options reltol=0.003
.options cshunt=1e-15
```

- `gmin` is the minimum conductance allowed by the program. The default value is `1e-12` (1 pS).
- `abstol` is the absolute current error tolerance of the program. The default value is `1e-12` (1 pA).
- `reltol` is the relative error tolerance of the program. The default value is `0.001` (0.1%).
- `cshunt` adds a capacitor of the specified value from each voltage node in the circuit to ground.

Simulation pin assignments

If unexpected results are generated by a simulation despite it running without obvious errors, it is worth double-checking that the assignments between the pins of the part instance in the KiCad schematic and that of the associated model are correct. These have to be correct for each instance of the model in the schematic.

Helpful hints

Some helpful tips, hints and advice to help you get the most from using ngspice in KiCad for simulation.

The ngspice manual is your friend!

Fundamentally the KiCad Simulator is a user-friendly front-end to the powerful ngspice circuit simulator. Therefore problems with the details of simulation, the correct use of SPICE elements, models, etc. is beyond the scope of the KiCad documentation but is very likely to be fully and completely addressed in the ngspice documentation itself. It's therefore recommended not to overlook this valuable resource.

NOTE

KiCad updates may result in changes to the ngspice version used by the simulator. The current version of ngspice used in a particular version of KiCad is shown on the **Help** → **About KiCad** dialog, under the **Version** tab. Referring to the online ngspice documentation will ensure you always have access to the latest information for reference.

Organizing third-party models

Although it is certainly possible to keep simulation models (i.e. `.lib`, `.sub` files) in the directories associated with individual KiCad projects, this will likely result in unnecessary copies of model files proliferating as you create simulations. Consider creating a dedicated storage location (directory, folder) for models, perhaps organized by manufacturer or device type, to hold these files. Then they may simply be referenced in simulations at a common location.

Simulation symbols and models in KiCad's libraries

The KiCad libraries provide a number of symbols for simulation in the `Simulation_SPICE` symbol library. Most of these symbols have SPICE models already assigned as appropriate, although you may need to adjust the model parameters in order to obtain the desired simulation behavior.

The `Simulation_SPICE` symbol library contains the following symbols:

Symbol Name	Description
0	A ground (node 0) symbol. Note that a standard GND symbol from another library can also be used.
BSOURCE	A symbol for a SPICE B-source (nonlinear dependent voltage or current source). Refer to the ngspice manual for information on B-sources.

D	A diode symbol. Note that the pin assignment is appropriate for both simulation and PCB layout.
ESOURCE	A symbol for a SPICE E-source (linear voltage-controlled voltage source). By default it is set up with a gain of 1 V/V. Refer to the ngspice manual for more information on E-sources.
GSOURCE	A symbol for a SPICE G-source (linear voltage-controlled current source). By default it is set up with a gain of 1 A/V. Refer to the ngspice manual for more information on G-sources.
IAM	A symbol for a SPICE amplitude-modulated independent current source. Refer to the ngspice manual for more information on independent current sources.
IBIS_DEVICE	An IBIS device symbol representing a digital input pin. This symbol is not preconfigured with a simulation model. It is intended to be used with an external IBIS model for a device.
IBIS_DEVICE_DIFF	An IBIS device symbol representing a digital differential input pin. This symbol is not preconfigured with a simulation model. It is intended to be used with an external IBIS model for a differential device.
IBIS_DRIVER	An IBIS driver symbol representing a digital output pin. This symbol is not preconfigured with a simulation model. It is intended to be used with an external IBIS model for a driver.
IBIS_DRIVER_DIFF	An IBIS driver symbol representing a pair of digital differential output pins. This symbol is not preconfigured with a simulation model. It is intended to be used with an external IBIS model for a differential driver.
IDC	A symbol for a SPICE DC independent current source. Refer to the ngspice manual for more information on independent current sources.
IEXP	A symbol for a SPICE exponential independent current source. Refer to the ngspice manual for more information on independent current sources.
IPULSE	A symbol for a SPICE pulsed independent current source. Refer to the ngspice manual for more information on independent current sources.
IPWL	A symbol for a SPICE piecewise linear independent current source. The waveform is specified as space-separated time-current pairs. Refer to the ngspice manual for more information on independent current sources.
ISFFM	A symbol for a SPICE single-frequency frequency-modulated independent current source. Refer to the ngspice manual for more information on independent current sources.
ISIN	A symbol for a SPICE sinusoidal independent current source. Refer to the ngspice manual for more information on independent current sources.

ITRRANDOM	A symbol for a SPICE transient random independent current source. Refer to the ngspice manual for more information on independent current sources.
NJFET	A symbol for an N-channel JFET with drain, gate, and source terminals, suitable for use with 3-terminal N-JFET models.
NMOS	A symbol for an N-channel MOSFET with drain, gate, and source terminals, suitable for use with 3-terminal N-MOSFET models.
NMOS_Substrate	A symbol for an N-channel MOSFET with drain, gate, source, and substrate (bulk) terminals, suitable for use with 4-terminal N-MOSFET models.
NPN	A symbol for an NPN transistor with collector, base, and emitter terminals, suitable for use with 3-terminal NPN models.
NPN_Substrate	A symbol for an NPN transistor with collector, base, emitter, and substrate terminals, suitable for use with 4-terminal NPN models.
OPAMP	A generic single-pole operational amplifier symbol. There are parameters for pole frequency, open loop gain, offset voltage, and output resistance. These parameters can be edited in the Parameters grid of the SPICE Model Editor dialog.
PJFET	A symbol for a P-channel JFET with drain, gate, and source terminals, suitable for use with 3-terminal P-JFET models.
PMOS	A symbol for a P-channel MOSFET with drain, gate, and source terminals, suitable for use with 3-terminal P-MOSFET models.
PMOS_Substrate	A symbol for a P-channel MOSFET with drain, gate, source, and substrate (bulk) terminals, suitable for use with 4-terminal P-MOSFET models.
PNP	A symbol for a PNP transistor with collector, base, and emitter terminals, suitable for use with 3-terminal PNP models.
PNP_Substrate	A symbol for a PNP transistor with collector, base, emitter, and substrate terminals, suitable for use with 4-terminal PNP models.
SWITCH	A symbol for a voltage-controlled switch. Refer to the ngspice manual for more information on switches.
TLINE	A symbol for a transmission line. By default it is set up as a lossless transmission line but it can also be used for a lossy transmission line. Refer to the ngspice manual for more information on transmission lines.
VAM	A symbol for a SPICE amplitude-modulated independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.
VDC	A symbol for a SPICE DC independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.

VOLTMETER_DIFF	A differential voltmeter symbol that can be used to measure the voltage between two arbitrary nodes. The voltage difference between the `+` and `-` pins is output as a ground-referenced voltage on the `out` pin.
VPULSE	A symbol for a SPICE pulsed independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.
VPWL	A symbol for a SPICE piecewise linear independent voltage source. The waveform is specified as space-separated time-voltage pairs. Refer to the ngspice manual for more information on independent voltage sources.
VSFFM	A symbol for a SPICE single-frequency frequency-modulated independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.
VSIN	A symbol for a SPICE sinusoidal independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.
VTRNOISE	A symbol for a SPICE transient noise independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.
VTRRANDOM	A symbol for a SPICE transient random independent voltage source. Refer to the ngspice manual for more information on independent voltage sources.

Most of these symbols in the `Simulation_SPICE` library use built-in models, but several symbols use external models from the `Simulation_SPICE.sp` simulation model library. The simulation model library is a separate file in the same folder as the symbol library. This simulation model library also contains SPICE models that may be useful for use with other symbols.

The `Simulation_SPICE.sp` simulation model library contains the following models:

Model Name	Description
kicad_builtin_opamp	A generic single-pole operational amplifier model. This model is used by the OPAMP symbol in the `Simulation_SPICE` symbol library, and the pins are ordered appropriately for use with any standard single-unit opamp symbol. There are parameters for pole frequency, open loop gain, offset voltage, and output resistance. These parameters can be edited in the Parameters grid of the SPICE Model Editor dialog.
kicad_builtin_opamp_dual	A dual version of the kicad_builtin_opamp model, with the same parameters. This model is not assigned to any symbols in the `Simulation_SPICE` symbol library, but the pins are ordered appropriately for use with standard dual opamp symbols.
kicad_builtin_opamp_quad	A quad version of the kicad_builtin_opamp model, with the same parameters. This model is not assigned to any symbols in the `Simulation_SPICE` symbol library, but the pins are ordered appropriately for use with standard quad opamp symbols.
kicad_builtin_varistor	A generic varistor model. This model is not assigned to any symbols in the `Simulation_SPICE` symbol library. There are parameters for threshold voltage, dynamic resistance, and leakage resistance. These parameters can be edited in the Parameters grid of the SPICE Model Editor dialog.
kicad_builtin_vdiff	A differential voltmeter model. The voltage difference between the first two terminals is output as a ground-referenced voltage on the third terminal. This model is used by the VOLTMETER_DIFF symbol in the `Simulation_SPICE` symbol library.

Advanced Topics

Configuration and Customization

NOTE	This section of the KiCad documentation has not yet been written. We appreciate your patience as our small team of volunteer documentation writers work to update and expand the documentation.

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, label names, label fields, hierarchical sheet fields, 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 in the schematic, 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 project 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 Schematic 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 PCB Editor.

Variables used in hierarchical sheet fields refer to the properties of the hierarchical sheet, not the parent, unless otherwise noted. For example, `${#}` returns the subsheet's page number when used in a hierarchical sheet field, but the parent sheet's page number when used in graphic text in the parent sheet.

Variables can also be used for field names. A field with a variable as its name will automatically have its value set to the same variable. For example, in a project with a project variable `MY_VAR` set to `MY_VALUE`, a user-created symbol field named `${MY_VAR}` will automatically have its value set to `${MY_VAR}`, which will then resolve to `MY_VALUE`. If the field's **Show Name** property is set, the variable's name will be displayed as the field name, for example `MY_VAR: MY_VALUE`.

Variable name	Description
#	Sheet number.
##	Total number of schematic sheets.
ADDR	Column (letter) and row (zero-indexed number) address of current table cell. This variable is only available in table cells.

Variable name	Description
CELL(<row>, <col>)	Contents of the table cell indexed by <row> and <col> (zero-indexed). This variable is only available in table cells.
COL	Column number (zero-indexed) of current table cell. This variable is only available in table 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 **root schematic sheet**, with a file extension.
FILEPATH	Full file path of the **root schematic sheet**, 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.
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 table cells.
SHEETFILE	Filename of the **current sheet**, with a file extension.
SHEETNAME	Sheet name of the current sheet.
SHEETPATH	Sheet path of the current sheet.
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 project text variable <variablename>.

Variable name	Description
`<fieldname>`	Contents of symbol field, symbol attribute, hierarchical sheet field, or label field `<fieldname>`. Fields can only be accessed from within their parent object, so symbol fields can be accessed from other text or fields within the symbol, and hierarchical sheet fields can be accessed within the sheet or in other sheet fields of the sheet. Both built-in and user-defined fields are available. Built-in fields use all uppercase letters: for example, to access a symbol's value, use `${VALUE}`. Built-in symbol fields are `DATASHEET`, `DESCRIPTION`, `FOOTPRINT`, `FOOTPRINT_LIBRARY`, `FOOTPRINT_NAME`, `NET_CLASS(<pin_number>)`, `NET_NAME(<pin_number>)`, `OP`, `PIN_ALT_LIST(<pin_number>)`, `PIN_BASE_NAME(<pin_number>)`, `PIN_NAME(<pin_number>)`, `REFERENCE`, `REFERENCE(<pin_number>)`, `SHORT_NET_NAME(<pin_number>)`, `SHORT_REFERENCE`, `SHORT_REFERENCE(<pin_number>)`, `SYMBOL_DESCRIPTION`, `SYMBOL_KEYWORDS`, `SYMBOL_LIBRARY`, `SYMBOL_NAME`, `UNIT`, `UNIT(<pin_number>)`, and `VALUE`. Built-in symbol attributes are `DNP`, `EXCLUDE_FROM_BOARD`, `EXCLUDE_FROM_BOM`, and `EXCLUDE_FROM_SIM`. These attributes expand to the friendly name of the attribute if the attribute is set (e.g. `Excluded from board` for `EXCLUDE_FROM_BOARD` and `DNP` for `DNP`), or to an empty string if the attribute is not set. Built-in sheet fields are `SHEETFILE`, `SHEETNAME`, and `SHEETPATH`. These refer to the child sheet's filename, sheet name, and sheet path, respectively, rather than the parent sheet's. Built-in label fields are `CONNECTION_TYPE`, `NET_CLASS`, `NET_NAME`, `OP`, `SHORT_NET_NAME`, and `INTERSHEETREFS` (global labels only).
`<refdes>:<fieldname>`	Contents of field or attribute `<fieldname>` in symbol `<refdes>`. Both built-in and user-defined fields are available. Built-in fields use all uppercase letters: for example, to access the value of `U1`, use `${U1:VALUE}`. Built-in symbol fields are `DATASHEET`, `DESCRIPTION`, `FOOTPRINT`, `FOOTPRINT_LIBRARY`, `FOOTPRINT_NAME`, `NET_CLASS(<pin_number>)`, `NET_NAME(<pin_number>)`, `OP`, `PIN_NAME(<pin_number>)`, `REFERENCE`, `REFERENCE(<pin_number>)`, `SHORT_NET_NAME(<pin_number>)`, `SHORT_REFERENCE`, `SHORT_REFERENCE(<pin_number>)`, `SYMBOL_DESCRIPTION`, `SYMBOL_KEYWORDS`, `SYMBOL_LIBRARY`, `SYMBOL_NAME`, `UNIT`, `UNIT(<pin_number>)`, `VALUE`. Built-in symbol attributes are `DNP`, `EXCLUDE_FROM_BOARD`, `EXCLUDE_FROM_BOM`, and `EXCLUDE_FROM_SIM`. These attributes expand to the friendly name of the attribute if the attribute is set (e.g. `Excluded from board` for `EXCLUDE_FROM_BOARD` and `DNP` for `DNP`), or to an empty string if the attribute is not set. **Note:** If the symbol referenced by `<refdes>` is reannotated, i.e. receives a new reference designator, `<refdes>` in the text variable will be automatically updated to the new reference designator so that the text variable continues to reference the same symbol.

Variable name	Description
`ERC_ERROR <errorname>`	Generates an ERC error named `<errorname>`. Everything inside the braces resolves to an empty string, while everything after the braces is included in the descriptive text for the ERC violation. The text variable must be at the beginning of the text item. For example, a text item containing `${ERC_ERROR TODO}Calculate resistor value` would display as the text "Calculate resistor value" and generate a ERC error named "TODO" with the description "Calculate resistor value".
`ERC_WARNING <warningname>`	Generates an ERC warning named `<warningname>`. This behaves the same as `ERC_ERROR`, except a warning is generated rather than an error.

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

Arithmetic operator	Description	Example
`+`	Addition **Note:** `+` can also be used to concatenate two strings, or to concatenate a string and a number.	`@{1 + 1}` = `2`
`-`	Subtraction	`@{1 - 1}` = `0`
`*`	Multiplication	`@{2 * 2}` = `4`

`/`	Division (the expression is not evaluated if the divisor is 0)	`@{3 / 2}` = `1.5`
		`@{1 / 0}` = `@{1 / 0}`

`%`	Modulo (the expression is not evaluated if the divisor is 0)	`@{3 % 2}` = `1`
		`@{3 % -2}` = `1`
		`@{1 % 0}` = `@{1 % 0}`

`^`	Exponentiation (right-associative)	`@{2 ^ 3}` = `8`
		`@{2 ^ 3 ^ 2}` = `512`

The following expression functions are available for performing mathematical functions. For example, `@{abs(-1)}` evaluates to `1`.

Expression function	Description	Example
`abs(x)`	Evaluates to the absolute value of `x`.	`@{abs(-1)}` = `1`

`sqrt(x)`	Evaluates to the square root of `x`. The expression is not evaluated if `x` is negative.	`@{sqrt(4)}` = `2`
		`@{sqrt(-1)}` = `@{sqrt(-1)}`

`pow(x, y)`	Evaluates to `x` to the power of `y`.	`@{pow(2, 3)}` = `8`
`floor(x)`	Evaluates to the greatest integer less than or equal to `x`.	`@{floor(3.7)}` = `3`
`ceil(x)`	Evaluates to the smallest integer greater than or equal to `x`.	`@{ceil(3.2)}` = `4`

`round(x, [precision])`	Evaluates to `x` rounded to the number of decimal places given by `precision`. `precision` is optional; if it is not given, `x` is rounded to the nearest integer.	`@{round(3.7)}` = `4`
		`@{round(3.2)}` = `3`
		`@{round(3.14159, 2)}` = `3.14`

`min(…)`	Evaluates to the minimum of the given comma-separated values.	`@{min(5, 3, 8, 1)}` = `1`

Expression function	Description	Example
`max(…)`	Evaluates to the maximum of the given comma-separated values.	`@{max(5, 3, 8, 1)} = 8`
`sum(…)`	Evaluates to the sum of the given comma-separated values.	`@{sum(1, 2, 3, 4)}` = `10`
`avg(…)`	Evaluates to the average of the given comma-separated values.	`@{avg(2, 4, 6)} = 4`
`random()`	Evaluates to a random number between `0` (inclusive) and `1` (exclusive).	`0` ≤ `@{random()}` < `1`
`shunt(r1, r2)`	Evaluates to the equivalent resistance of the two parallel resistances `r1` and `r2`: `(r1*r2) / (r1+r2)`.	`@{shunt(10k, 10k)} = 5000`
`db(ratio)`	Converts the power ratio `ratio` to decibels: `10*log10(ratio)`.	`@{db(2)}` ≈ `3`
`dbv(ratio)`	Converts the voltage or current ratio `ratio` to decibels: `20*log10(ratio)`.	`@{dbv(2)} ≈ 6`
`fromdb(db)`	Converts the quantity `db`, in decibels, to a power ratio: `10^(db/10)`.	`@{fromdb(3)}` ≈ `2`
`fromdbv(db)`	Converts the quantity `db`, in decibels, to a voltage or current ratio: `10^(db/20)`.	`@{fromdbv(6)} ≈ 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.	`@{enearest(5)}` = `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.	`@{eup(5)} = 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.	`@{edown(5)}` = `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	`@{1 < 2}` = `1`
`<=`	Less than or equal to	`@{1 <= 2}` = `1`
`>`	Greater than	`@{1 > 2}` = `0`
`>=`	Greater than or equal to	`@{1 >= 2}` = `0`
`==`	Equal to	`@{1 == 2}` = `0`
`!=`	Not equal to	`@{1 != 2}` = `1`

The following expression functions are available for operating on the result of a boolean or mathematical expression.

Expression function	Description	Example

`if(condition, true_value, false_value)`	Evaluates to `true_value` if `condition` evaluates to a true (non-zero) value. Otherwise, evaluates to `false_value`.	`@{if(0, "true", "false")}` = `false`
		`@{if(1 < 2, 5, 10)}` = `5`

String manipulation

The following expression functions are available for manipulating strings.

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.	`@{upper("hello world")}` = `HELLO WORLD`
`lower(str)`	Evaluates to `str` converted to lowercase.	`@{lower("HELLO WORLD")}` = `hello world`

`concat(…)`	Evaluates to the given comma-separated strings or numbers concatenated together.
	Note: strings and numbers can also be concatenated using the `+` operator.
		`@{concat("Hello", " ", "World")}` = `Hello World`
		`@{concat("Count: ", 42, " items")}` = `Count: 42 items`
		`@{"Count: " + 42 + " items"}` = `Count: 42 items`

Expression function	Description	Example
`beforefirst(str, c)`	Evaluates to the portion of `str` before the first occurrence of the character `c` in `str`.	`@{beforefirst("hello.world.txt", ".")} = hello`
`beforelast(str, c)`	Evaluates to the portion of `str` before the last occurrence of the character `c` in `str`.	`@{beforelast("hello.world.txt", ".")}` = `hello.world`
`afterfirst(str, c)`	Evaluates to the portion of `str` after the first occurrence of the character `c` in `str`.	`@{afterfirst("hello.world.txt", ".")} = world.txt`
`afterlast(str, c)`	Evaluates to the portion of `str` after the last occurrence of the character `c` in `str`.	`@{afterlast("hello.world.txt", ".")}` = `txt`

`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.
	Note: `format()` behaves exactly the same as `fixed()`.
		`@{format(3.14159)} = 3.14`
		`@{format(3.14159, 3)} = 3.142`
		`@{format(1234.5)} = 1234.50`

`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.
	Note: `fixed()` behaves exactly the same as `format()`.
		`@{fixed(3.14159)}` = `3.14`
		`@{fixed(3.14159, 3)}` = `3.142`
		`@{fixed(1234.5)}` = `1234.50`

`currency(amount, [symbol])`	Formats `amount` as a string containing the currency symbol specified by `symbol` and a value with two decimal places. `symbol` is optional; if it is not given, `"$"` is used.	`@{currency(1234.56)} = $1234.56`
		`@{currency(999.99, "€")} = €999.99`

Date and time formatting

The following expression functions are available for manipulating dates and times.

Expression function	Description	Example
`today()`	Evaluates to the current date, represented as the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01). **Note:** this value can be formatted in various ways using the `dateformat()` function. You can add or subtract numbers to this value to get dates relative to the current date.	
`now()`	Evaluates to the current timestamp, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1970-01-01). **Note:** you can add or subtract numbers to this value to get timestamps relative to the current timestamp.	

`dateformat(days, [format])`	Formats `days` as a string depending on the specified date `format`. `days` is a number representing the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01). Valid values for `format` are `"ISO"`, `"US"`, `"EU"`, `"Chinese"`, `"Japanese"`, `"long"`, and `"short"`. `format` is optional; if it is not given, `"ISO"` is used.	`@{dateformat(0)}` = `1970-01-01`
		`@{dateformat(0, "ISO")}` = `1970-01-01`
		`@{dateformat(0, "short")}` = `Jan 1, 1970`

`weekdayname(days)`	Evaluates to the day of the week of the date represented by `days`. `days` is a number representing the number of days since the Unix epoch (00:00:00 UTC on 1970-01-01).	`@{weekdayname(0)}` = `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
`vcsidentifier([length])`	Evaluates to the current HEAD commit identifier, 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.

Expression function	Description
`vcsbranch()`	Evaluates to the current branch name.
`vcsauthor()`	Evaluates to the current HEAD commit's author's name.
`vcsauthoremail()`	Evaluates to the current HEAD commit's author's email address.
`vcscommitter()`	Evaluates to the current HEAD commit's committer's name.
`vcscommitteremail()`	Evaluates to the current HEAD commit's committer's email address.
`vcscommitdate([format])`	Evaluates to the current HEAD commit's commit date. Valid values for `format` are `"ISO"`, `"US"`, `"EU"`, `"Chinese"`, `"Japanese"`, `"long"`, and `"short"`. `format` is optional; if it is not given, `"ISO"` is used.
`vcsnearestlabel([match], [anyTags])`	Evaluates to the name of the matching tag closest to the HEAD commit. `match` 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 `*`. If `anyTags` is `0` or not given, only annotated tags will be considered. If `anyTags` is `1`, all tags will be considered.
`vcslabeldistance([match], [anyTags])`	Evaluates to the number of commits between HEAD and the matching tag closest to the HEAD commit. `match` 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 `*`. If `anyTags` is `0` or not given, only annotated tags will be considered. If `anyTags` is `1`, all tags will be considered.
`vcsdirty([includeUntracked])`	Evaluates to `1` if the repository is dirty, or `0` if it is not. If `includeUntracked` is `0` or not given, untracked files will not contribute to the repository's dirty status. If `includeUntracked` is 1, untracked files will contribute.
`vcsdirtysuffix([suffix], [includeUntracked])`	Evaluates to the given `suffix` if the repository is dirty or an empty string if it is not. `suffix` is optional; if it is not given, `-dirty` is used. If `includeUntracked` is `0` or not given, untracked files will not contribute to the repository's dirty status. If `includeUntracked` is 1, untracked files will contribute.
`vcsfileidentifier(file, [length])`	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.
`vcsfileauthor(file)`	Evaluates to the name of the author of the last commit that modified the specified `file`.
`vcsfileauthoremail(file)`	Evaluates to the email address of the author of the last commit that modified the specified `file`.

Expression function	Description
`vcsfilecommitter(file)`	Evaluates to the name of the committer of the last commit that modified the specified `file`.
`vcsfilecommitteremail(file)`	Evaluates to the email address of the committer of the last commit that modified the specified `file`.
`vcsfilecommitdate(file, [format])`	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.

Database Libraries

A database library is a type of KiCad symbol library that holds data about parts in an external SQL database. Database libraries do not contain any symbol or footprint definitions by themselves. Instead, they **reference** symbols and footprints found in other KiCad libraries. Each database library entry maps a KiCad symbol (from another library) to a set of properties (fields) and usually a KiCad footprint (from a footprint library).

Using database libraries allows you to create fully-defined parts (sometimes called **atomic parts**) out of KiCad symbols and footprints without needing to store all the part properties in a symbol library. The external database can be linked to third-party tools for managing part data and lifecycles. Database library workflows are generally more complex than the standard KiCad library workflows, and so this type of library is typically only used in situations where it makes managing a large library of fully-defined parts more efficient (such as in organization or team settings).

KiCad does not provide a GUI for editing a SQL database or defining a database library. It is up to the user to find the most appropriate workflow and toolchain for creating and updating the database itself. Some users may want to directly edit the database through a third-party database client, and some may use other third-party software such as a part lifecycle management (PLM) tool to create and edit data.

In a database library, there are one or more **tables** that generally represent a single type of part (such as Resistors or Capacitors). Each table can have an independent schema, meaning that different types of parts can have different properties that are translated into symbol fields in KiCad. Each table must have a unique ID column which is used as the identifier for a symbol placed from that table. This unique ID will typically be a part number (either a manufacturer's part number, or an internal organization part number). Each table must also have a column that contains a mapping to a KiCad symbol, in the form `LibraryNickname:SymbolName`. The `LibraryNickname` must match a symbol library that is present in the KiCad library tables. Tables may also contain a column containing a KiCad footprint, in the form `LibraryNickname:FootprintName`. If this column is present, symbols placed from the table will include a footprint mapping.

Tables may also contain arbitrary additional columns that may optionally be mapped to symbol fields in KiCad. The KiCad database library configuration file controls how these fields should be named, whether or not to make the fields visible, and whether or not to include the fields in the data displayed in the Symbol Chooser.

Database Library Configuration Files

and save it with a `kicad_dbl` extension. You can then add this file to your global symbol library table using the Configure Symbol Libraries dialog.

```
{
    "meta": {
        "version": 0
    },
    "name": "My Database Library",
    "description": "A database of components",
    "globally_unique_keys": false,
    "source": {
        "type": "odbc",
        "dsn": "",
        "username": "",
        "password": "",
        "timeout_seconds": 2,
        "connection_string": ""
    },
    "libraries": [
        {
            "name": "Resistors",
            "table": "Resistors",
            "key": "Part ID",
            "symbols": "Symbols",
            "footprints": "Footprints",
            "fields": [
                {
                    "column": "MPN",
                    "name": "MPN",
                    "visible_on_add": false,
                    "visible_in_chooser": true,
                    "show_name": true,
                    "inherit_properties": true
                },
                {
                    "column": "Value",
                    "name": "Value",
                    "visible_on_add": true,
                    "visible_in_chooser": true,
                    "show_name": false
                }
            ],
            "properties": {
                "description": "Description",
                "footprint_filters": "Footprint Filters",
                "keywords": "Keywords",
                "exclude_from_bom": "No BOM",
                "exclude_from_board": "Schematic Only"
            }
        }
    ]
}
```

NOTE Database library files are in JSON format. Standard JSON syntax rules apply. To check if your file contains syntax errors, you may use a JSON validator or linter (available online).

Configuring top level settings

`name` : The name of the library.

`description` : A description of the library.

`globally_unique_keys` : *optional, defaults to false*. Controls the format of the library link when placing parts in schematics. If `true` library link format is `LIB_NAME:KEY`. If `false` library link format is `LIB_NAME:SUB_LIB_NAME/KEY`. If `true` keys must be unique across all sub-libraries. When enabled this allows parts to be moved between sub-libraries without breaking the link in schematics (useful for library refactors and fixing miscategorized parts). When `false` keys only need to be unique within each individual sub-library.

NOTE Changing `globally_unique_keys` on an existing library will break library links for parts previously placed in schematics.

Configuring the source

KiCad currently only supports ODBC connections to SQL databases. You can either connect with a DSN or a connection string. If a DSN name is supplied, the optional `username` and `password` fields will be used to connect to the DSN. If a connection string is supplied, the `dsn` , `username` , and `password` fields are ignored. The connection string will be passed directly to the ODBC driver, so you can include any parameters your ODBC driver supports.

When using a DSN connection, leave the `connection_string` property blank or omit it from the file. When using a connection string, leave the `dsn` , `username` , and `password` fields blank or omit them from the file. Connection strings must start with a `Driver` key indicating to the ODBC manager which driver should be used, and may include other keys that depend on the specific driver. Check the documentation for your ODBC driver for details. You may also find a reference site like connectionstrings.com useful when configuring a database connection.

KiCad does not recommend or endorse any particular ODBC driver or database server, but has been tested to work with Sqlite, MySQL, MariaDB, and PostgreSQL.

NOTE Windows users: the backslash character (`\`) must be escaped with a second backslash when included in a JSON quoted string. If including a file path in your connection string, make sure to use double backslashes (`\\`).

NOTE Flatpak users: You need to install the corresponding ODBC drivers as Flatpak extensions. You can do this via the "Add-ons" section for KiCad in your software manager (i.e. GNOME Software), or via the command line: Run `flatpak install org.kicad.KiCad.ODBCDriver.sqliteodbc` for SQLite, `flatpak install org.kicad.KiCad.ODBCDriver.mariadb-connector-odbc` for MariaDB or MySQL, or `flatpak install org.kicad.KiCad.ODBCDriver.psqlodbc` for PostgreSQL.

NOTE

Flatpak users: Due to Flatpak sandboxing, a possible way to connect to database servers running on your local machine is via TCP/IP. Make sure that your database server allows TCP/IP connections, then add the required `Port` parameter to your connection string. For example, add `Port=3306;` for the default TCP port of MySQL/MariaDB, or `Server=localhost;Port=5432;` to force PostgreSQL to use a TCP connection to the local server. Using the default UNIX domain socket connections for MySQL, MariaDB, or PostgreSQL is only possible when overriding host file system permissions via `flatpak override`.

Configuring libraries

Each database library can contain "sub-libraries" mapped to a single database table. The `libraries` entry in the configuration file contains a list of objects that each define a single library. The following settings must exist for each library:

`name`: The name of the sub-library (table) that will be shown in the KiCad UI and included as a prefix in each symbol name placed from this sub-library. This name can include any valid characters for a symbol name except for a forward slash (`/`) because the slash character is used as a separator between the sub-library name and the symbol name. If this field is left blank, no prefix will be added to symbols in this sub-library.

`table`: The name of the table in the database.

`key`: The column name containing a unique key that will be used to identify parts from the table.

`symbols`: The column name containing KiCad symbol references.

`footprints`: The column name containing KiCad footprint references.

`fields`: A list of field definitions. Each field defined here will be added to the symbol when it is placed on the schematic. If a field with a matching name is already defined in the source symbol, the value from the database table will override whatever value was defined in the source symbol. Each field definition may contain:

`column`: The name of the database table column that should be mapped to a field.

`name`: The name of the KiCad field to populate from the database.

`visible_on_add`: If `true`, this field will be visible in the schematic when a symbol is added. If this setting is not specified, it will default to `false`.

`visible_in_chooser`: If `true`, this field will be shown in the Symbol Chooser as a column. If this setting is not specified, it will default to `false`.

`show_name`: If `true`, the field's name will be shown in addition to its value in the schematic. If this setting is not specified, it will default to `false`.

`inherit_properties`: If `true`, and a field with the given `name` already exists on the source symbol, only the field contents will be updated from the database, and the other properties (`visible_on_add`, `show_name`, etc) will be kept as they were set in the source symbol. If the given field name does not exist in the source symbol, this setting is ignored. If this setting is not specified, it will default to `false`.

`properties`: A map of symbol properties to database columns. All properties are optional; any that are not specified in the database library configuration will be inherited from the values set for the source symbol.

The following properties are supported:

`description`: The symbol’s Description property.

`footprint_filters`: Reserved for future expansion.

`keywords`: The symbol’s Keywords property.

`exclude_from_bom`: The symbol’s "Exclude from Bill of Materials" setting. The column named here must be a numeric type, and will be taken as a boolean (0 for false, 1 for true).

`exclude_from_board`: The symbol’s "Exclude from PCB" setting. The column named here must be a numeric type, and will be taken as a boolean (0 for false, 1 for true).

`exclude_from_sim`: The symbol’s "Exclude from simulation" setting. The column named here must be a numeric type, and will be taken as a boolean (0 for false, 1 for true).

Database columns may be mapped to custom (user-defined) fields, or to certain built-in KiCad fields, including `Value` and `Datasheet`.

> **NOTE** KiCad only supports text (string) fields. If you map a database column containing a numeric SQL data type, it will be converted to a string using a general-purpose conversion algorithm that will switch to scientific notation for very large or very small numbers. This format conversion cannot be fine-tuned by the user, so if explicit control over number-to-string conversion is needed, a new column or view should be used to do the conversion in the database.

Using database libraries

After creating your configuration file and adding it to your symbol library table, you can place parts from the database tables using the Symbol Chooser. Parts placed from a database library can be updated using the Update Symbols from Library function, which will update any fields that were changed in the database as well as updating the underlying symbol if it was changed in the source library.

Note that any source library referenced by a database table must also be present in the symbol library table for the database library to function. If you want to use a library only as a source of symbols for a database library, you can hide it from the Symbol Chooser by clearing the "Visible" checkbox in the Manage Symbol Libraries dialog.

HTTP Libraries

HTTP libraries are a type of KiCad symbol library that sources data about parts for an external source such as an ERP system. They do not contain any symbol or footprint definitions as standard KiCad libraries do. Instead, they **reference** symbols and footprints found in other KiCad libraries.

HTTP libraries are read only and support REST or REST-like APIs.

HTTP Library Configuration Files

To create an HTTP library, you must create a configuration file that contains the necessary information for KiCad to connect to the providing library (API) and to retrieve data from it.

Copy the template below into a new file and save it using the `.kicad_httplib` file extension. You should then edit this file and replace `root_url` and `token` values with your own. Once saved, add this file to your global symbol library table using the Configure Symbol Libraries dialog which can be found under **Preferences → Manage Symbol Libraries...**.

Users have the option to configure two timeout settings. The `timeout_parts_seconds` setting dictates the validity duration of a part's information, while the `timeout_categories_seconds` setting determines how long categories remain valid. The default values are set to 60 seconds and 600 seconds respectively, but if the data for either setting is anticipated to remain unchanged, users can opt for higher values. This will significantly speed up the opening of the symbol chooser. It's important to note that KiCad will re-cache the data on the initial startup regardless of these timeout settings.

```
{
    "meta": {
        "version": 1.0
    },
    "name": "KiCad HTTP Library",
    "description": "A KiCad library sourced from a REST API",
    "source": {
        "type": "REST_API",
        "api_version": "v1",
        "root_url": "http://localhost:8000/kicad-api",
        "token": "usertokendatastring",
        "timeout_parts_seconds": 60,
        "timeout_categories_seconds": 600
    }
}
```

Authentication

Authentication is done via an **Access Token** only. Users need to ask their administrators to get a valid token issued if the HTTP library is maintained externally.

Caching Behaviour for Categories

KiCad caches all available Categories once when opening the Symbol Chooser Dialog. Subsequently, any alterations made to the categories on the server side will remain undetected by KiCad until the user performs a program restart. This implementation is intentionally designed to conserve bandwidth resources, as it prevents KiCad from attempting to retrieve data from the API every time the user opens the Symbol Chooser Dialog. Such continuous data fetching, especially under constrained bandwidth conditions, would severely impede KiCad's performance.

Server Response Codes

If KiCad receives an API error, it will display an error message to the user. For more information about API errors and server responses, see the APIs and Bindings section at dev-docs.kicad.org.

Custom Netlist and BOM Formats

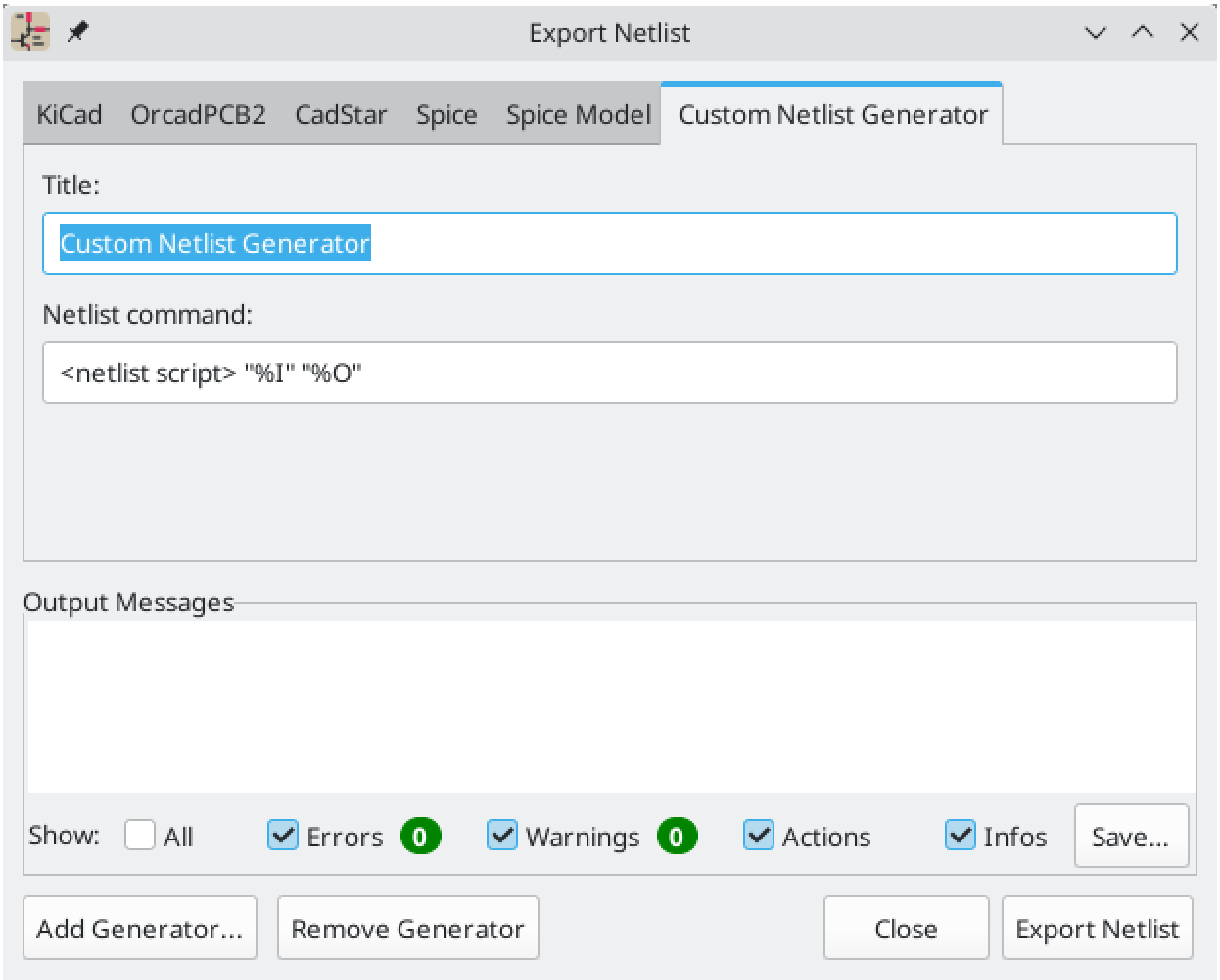
KiCad can output netlists and BOMs in various formats, and users can define new formats if desired.

The process of exporting a netlist is described in the netlist export section. BOM output is described in the BOM export section.

The following section describes how to create an exporter for a new output format.

Adding new netlist generators

New netlist generators are added to the **Export Netlist** dialog by clicking the **Add Generator…** button.



Title: Custom Netlist Generator

Netlist command: <netlist script> "%I" "%O"

Show: □ All ☑ Errors ☑ Warnings ☑ Actions ☑ Infos

New generators require a name and a command. The name is shown in the tab label, and the command is run whenever the **Export Netlist** button is clicked.

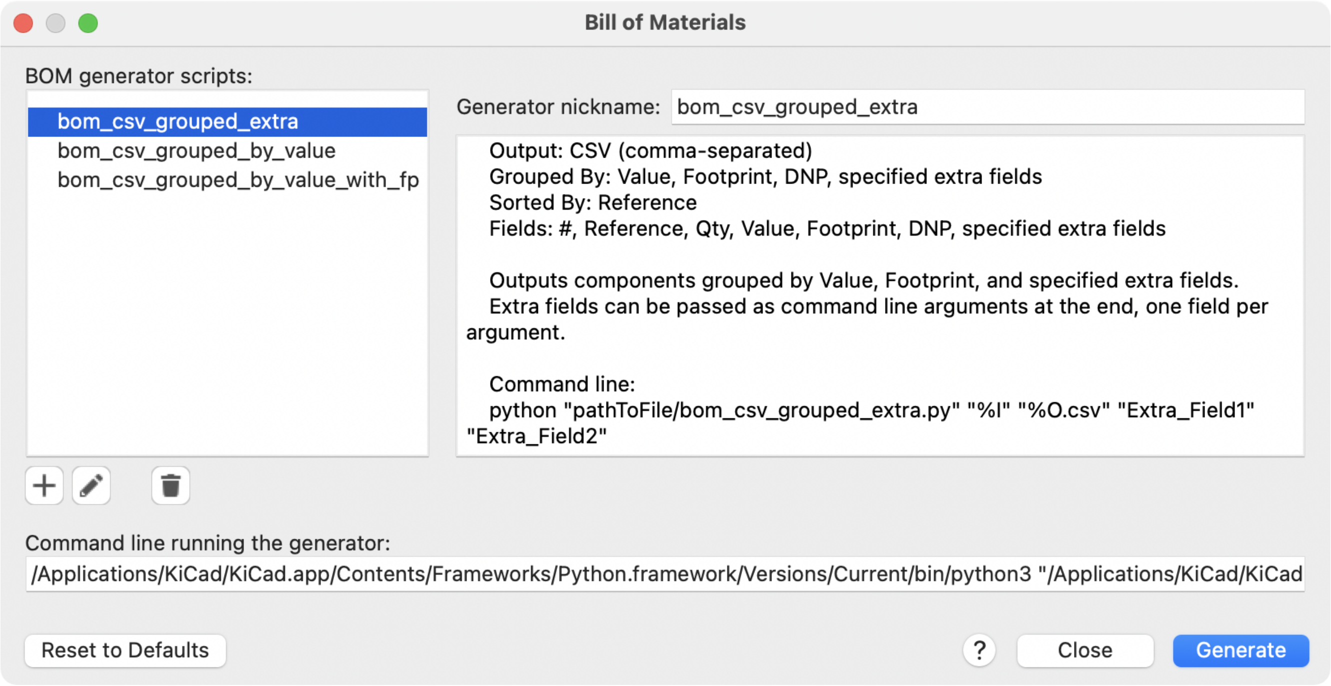
When the netlist is generated, KiCad creates an intermediate XML file which contains all of the netlist information from the schematic. The generator command is then run in order to transform the intermediate netlist into the desired netlist format.

The netlist command must be set up properly so that the netlist generator script takes the intermediate netlist file as input and outputs the desired netlist file. The exact netlist command will depend on the generator script used. The command format is described below.

Python and XSLT are commonly used tools to create custom netlist generators.

Adding a new BOM generator

KiCad also uses the intermediate netlist file to generate BOMs with the Generate BOM tool.



BOM generator scripts:

bom_csv_grouped_extra
bom_csv_grouped_by_value
bom_csv_grouped_by_value_with_fp

Generator nickname: bom_csv_grouped_extra

```
    Output: CSV (comma-separated)
    Grouped By: Value, Footprint, DNP, specified extra fields
    Sorted By: Reference
    Fields: #, Reference, Qty, Value, Footprint, DNP, specified extra fields

    Outputs components grouped by Value, Footprint, and specified extra fields.
    Extra fields can be passed as command line arguments at the end, one field per
argument.

    Command line:
    python "pathToFile/bom_csv_grouped_extra.py" "%I" "%O.csv" "Extra_Field1"
"Extra_Field2"
```

Command line running the generator:

```
/Applications/KiCad/KiCad.app/Contents/Frameworks/Python.framework/Versions/Current/bin/python3 "/Applications/KiCad/KiCad
```

Additional scripts can be added to the list of BOM generator scripts by clicking the + button. Scripts can be removed by clicking the 🗑 button. The ✏ button opens the selected script in a text editor.

Generator scripts written in Python and XSLT can contain a header comment that describes the generator's functionality and usage. This header comment is displayed in the BOM dialog as the description for each generator. The header comment must contain the string `@package`. Everything following that string until the end of the comment is used as the description for the generator.

KiCad automatically fills the command line field when a new generator script is added, but the command line might need to be adjusted by hand depending on the generator script. KiCad attempts to automatically determine the output file extension from the example command line in the generator script's header.

Generator command line format

The command line for a netlist or BOM exporter defines the command that KiCad will run to generate the selected output file.

For a netlist exporter using `xsltproc`, an example is:

```
xsltproc -o %O.net /usr/share/kicad/plugins/netlist_form_pads-pcb.asc.xsl %I
```

For a BOM exporter using Python, an example is:

```
/usr/bin/python3 /usr/share/kicad/plugins/bom_csv_grouped_by_value.py "%I" "%O.csv"
```

NOTE It is recommended to surround arguments in the command line with quotes (`"`) in case they contain spaces or other special characters.

Some character sequences like `%I` and `%O` have a special meaning in the command line, because KiCad replaces them with a filename or path before executing the command.

Parameter	Replaced with…	Description
`%I`	`<project path>/<project name>.xml`	Absolute path and filename of the intermediate netlist file, which is the input to the BOM or netlist generator plugin
`%O`	`<project path>/<project name>`	Absolute path and filename of the output BOM or netlist file (without file extension). An appropriate file extension may need to be specified after the `%O` sequence.
`%B`	`<project name>`	Base filename of the output BOM or netlist file (without path or file extension). An appropriate file extension may need to be specified after the `%B` sequence.
`%P`	`<project path>`	Absolute path of the project directory, without trailing slash.

Berkas Netlist Antara

When exporting BOM files and netlists, KiCad creates an intermediate netlist file and then runs a separate tool which post-processes the intermediate netlist into the desired netlist or BOM format.

The intermediate netlist uses XML syntax. It contains a large amount of data about the design. Depending on the output (BOM or netlist), different subsets of the complete intermediate netlist file will be included in the final output file.

The structure of the intermediate netlist file is described in detail below.

Because the conversion from intermediate netlist file to output netlist or BOM is a text-to-text transformation, the post-processing filter can be written using Python, XSLT, or any other tool capable of taking XML as input.

NOTE

XSLT is not recommended for new netlist or BOM exporters; Python or another tool should be used instead. Beginning with KiCad 7, `xsltproc` is no longer installed with KiCad, although it can be installed separately. Nevertheless, several examples of netlist exporters using XSLT are included below.

Struktur Netlist Antara

Contoh berikut ini menunjukkan format berkas Netlist.

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 21:07:51</date>
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamps>4C6E2141</tstamps>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamps>4C6E20BA</tstamps>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamps>4C6E20A6</tstamps>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamps>4C6E2094</tstamps>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamps>4C6E208A</tstamps>
    </comp>
  </components>
  <libparts/>
  <libraries/>
  <nets>
    <net code="1" name="GND">
      <node ref="U1" pin="7"/>
      <node ref="C1" pin="2"/>
      <node ref="U2" pin="7"/>
      <node ref="P1" pin="4"/>
    </net>
    <net code="2" name="VCC">
      <node ref="R1" pin="1"/>
      <node ref="U1" pin="14"/>
      <node ref="U2" pin="4"/>
      <node ref="U2" pin="1"/>
      <node ref="U2" pin="14"/>
      <node ref="P1" pin="1"/>
    </net>
    <net code="3" name="">
      <node ref="U2" pin="6"/>
    </net>
    <net code="4" name="">
      <node ref="U1" pin="2"/>
```

Struktur Berkas Netlist Umum

Berkas Netlist antara terdiri dari lima bagian.

- Bagian *header*.
- Bagian komponen.
- Bagian *lib parts*.
- Bagian pustaka.
- Bagian *net*.

The file content has the delimiter `<export>`

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

Bagian *Header*

The header has the delimiter `<design>`

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

Bagian ini bisa dianggap sebagai bagian komentar.

Bagian Komponen

The component section has the delimiter `<components>`

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

Bagian ini berisi daftar komponen yang ada pada skematik Anda. Setiap komponen dijelaskan seperti ini:

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

Element name	Element description
`libsource`	name of the lib where this component was found.
`part`	component name inside this library.
`sheetpath`	path of the sheet inside the hierarchy: identify the sheet within the full schematic hierarchy.
`tstamps`	timestamp of the component.

Note about time stamps for components

To identify a component in a netlist and therefore on a board, the timestamp reference is used as unique for each component. However KiCad provides an auxiliary way to identify a component which is the corresponding footprint on the board. This allows the re-annotation of components in a schematic project and does not lose the link between the component and its footprint.

Tanda waktu menjadi pengidentifikasi yang bersifat unik untuk setiap komponen atau lembar kerja pada suatu proyek skematik. Namun di hirarki kompleks, satu lembar kerja yang sama bisa digunakan lebih dari satu kali, sehingga lembar kerja ini akan berisi komponen-komponen dengan tanda waktu yang sama.

A given sheet inside a complex hierarchy has an unique identifier: its sheetpath. A given component (inside a complex hierarchy) has a unique identifier: the sheetpath and its timestamp.

Bagian *Libparts*

The libparts section has the delimiter `<libparts>`, and the content of this section is defined in the schematic libraries.

```
<libparts>
<libpart lib="device" part="CP">
  <description>Condensateur polarise</description>
  <footprints>
    <fp>CP*</fp>
    <fp>SM*</fp>
  </footprints>
  <fields>
    <field name="Reference">C</field>
    <field name="Valeur">CP</field>
  </fields>
  <pins>
    <pin num="1" name="1" type="passive"/>
    <pin num="2" name="2" type="passive"/>
  </pins>
</libpart>
</libparts>
```

Element name	Element description
<footprints>	The symbol's footprint filters. Each footprint filter is in a separate <fp> tag.
<fields>	The symbol's fields. Each field's name and value is given in a separate `<field name="fieldname">…</field> tag.
<pins>	The symbol's pins. Each pin is given in a separate <pin num="pinnum" type="pintype"/> tag. Possible pintypes are described below.

Possible electrical pin types are:

Pintype	Description
Input	Usual input pin
Output	Usual output
Bidirectional	Input or Output
Tri-state	Bus input/output
Passive	Usual ends of passive components
Unspecified	Unknown electrical type
Power input	Power input of a component
Power output	Power output like a regulator output
Open collector	Open collector often found in analog comparators
Open emitter	Open emitter sometimes found in logic
Not connected	Must be left open in schematic

Bagian Pustaka

The libraries section has the delimiter `<libraries>`. This section contains the list of schematic libraries used in the project.

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

Bagian *Net*

The nets section has the delimiter `<nets>`. This section describes the connectivity of the schematic by listing all nets and the pins connected to each net.

```
<nets>
  <net code="1" name="GND">
    <node ref="U1" pin="7"/>
    <node ref="C1" pin="2"/>
    <node ref="U2" pin="7"/>
    <node ref="P1" pin="4"/>
  </net>
  <net code="2" name="VCC">
    <node ref="R1" pin="1"/>
    <node ref="U1" pin="14"/>
    <node ref="U2" pin="4"/>
    <node ref="U2" pin="1"/>
    <node ref="U2" pin="14"/>
    <node ref="P1" pin="1"/>
  </net>
</nets>
```

Sebuah *net* bisa memiliki daftar seperti berikut ini.

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

Element name	Element Description
`net code`	an internal identifier for this net
`name`	the net name
`node`	the pin (identified by `pin`) of a symbol (identified by `ref`) which is connected to the net

Example netlist exporters

Some example netlist exporters using XSLT are included below.

XSLT itself is an XML language very suitable for XML transformations. The `xsltproc` program can be used to read the Intermediate XML netlist input file, apply a style-sheet to transform the input, and save the results in an output file. Use of `xsltproc` requires a style-sheet file using XSLT conventions. The full conversion process is handled by KiCad, after it is configured once to run `xsltproc` in a specific way.

The document that describes XSL Transformations (XSLT) is available here: http://www.w3.org/TR/xslt

NOTE When writing a new netlist exporter, consider using Python or another tool rather than XSLT.

PADS netlist example using XSLT

The following example shows how to create an exporter for the PADS netlist format using `xlstproc`.

The PADS netlist format is comprised of two sections:

- A list of footprints
- A list of nets, together with the pads connected to each net.

Below is an XSL style-sheet which converts the intermediate netlist file to the PADS netlist format.

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

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

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

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

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

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

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

<!-- for each node -->
```

And here is the PADS netlist output file after running `xsltproc`:

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

*END*
```

Perintah untuk menjalankan konversi ini adalah:

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

Cadstar netlist example using XSLT

The following example shows how to create an exporter for the Cadstar netlist format using `xlstproc`.

The Cadstar format is comprised of two sections:

- The footprint list
- The Nets list: grouping pads references by nets

Below is an XSL style-sheet which converts the intermediate netlist file to the Cadstar netlist format.

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

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

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

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

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

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

<!-- for each component -->
<xsl:template match="comp">
    <xsl:text>.ADD_COM </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "value != '' ">
            <xsl:text>"</xsl:text> <xsl:apply-templates select="value"/> <xsl:text>"
</xsl:text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>""</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!-- for each net -->
<xsl:template match="net">
    <!-- nets are output only if there is more than one pin in net -->
```

Berikut adalah berkas keluaran Cadstar.

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


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

.END
```

OrcadPCB2 netlist example using XSLT

This format has only one section which is the footprint list. Each footprint includes a list of its pads with reference to a net.

Below is an XSL style-sheet which converts the intermediate netlist file to the Orcad netlist format.

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

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

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

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

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

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

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

<!--
    Generate id in header like "eeschema (2010-08-17 BZR 2450)-unstable"
-->
<xsl:template match="tool">
    <xsl:apply-templates/>
</xsl:template>

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

<!--
    This template read each component
    (path = /export/components/comp)
    creates lines:
     ( 3EBF7DBD $noname U1 74LS125
      ... pin list ...
      )
    and calls "create_pin_list" template to build the pin list
-->
```

Berikut adalah berkas keluaran OrcadPCB2.

```
( { Eeschema Netlist Version 1.1  29/08/2010 21:07:51
eeschema (2010-08-28 BZR 2458)-unstable}
 ( 4C6E2141 $noname P1 CONN_4
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 /CLOCK_IN )
  (  4 GND )
 )
 ( 4C6E20BA $noname U2 74LS74
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 N-04 )
  (  4 VCC )
  (  5 /SIG_OUT )
  (  6 ? )
  (  7 GND )
  (  14 VCC )
 )
 ( 4C6E20A6 $noname U1 74LS04
  (  1 /CLOCK_IN )
  (  2 N-04 )
  (  7 GND )
  (  14 VCC )
 )
 ( 4C6E2094 $noname C1 CP
  (  1 /CLOCK_IN )
  (  2 GND )
 )
 ( 4C6E208A $noname R1 R
  (  1 VCC )
  (  2 /CLOCK_IN )
 )
)
*
```

Actions reference

Below is a list of every available **action** in the KiCad Schematic Editor: a command that can be assigned to a hotkey.

Schematic Editor

The actions below are available in the Schematic Editor. Hotkeys can be assigned to any of these actions in the **Hotkeys** section of the preferences.

Action	Default Hotkey	Description
Align Items to Grid		
Annotate Schematic…		Fill in schematic symbol reference designators
Annotate Automatically		Toggle automatic annotation of new symbols
Assign Footprints…		Run footprint assignment tool
Clear Net Highlighting	~	Clear any existing net highlighting
Export Drawing to Clipboard		Export drawing of current sheet to clipboard
Edit Library Symbol…	Ctrl + Shift + E	Open the library symbol in the Symbol Editor
Edit Sheet Page Number…		Edit the page number of the current or selected sheet
Edit Symbol Fields…		Bulk-edit fields of all symbols in schematic
Edit Symbol Library Links…		Edit links between schematic and library symbols
Edit with Symbol Editor	Ctrl + E	Open the selected symbol in the Symbol Editor
Export Netlist…		Export file containing netlist in one of several formats
Export Symbols to Library…		Add symbols used in schematic to an existing symbol library (does not remove other symbols from this library)
Export Symbols to New Library…		Create a new symbol library using the symbols used in the schematic (if the library already exists it will be replaced)

Action	Default Hotkey	Description
Generate Bill of Materials…		Generate a bill of materials for the current schematic
Generate Bill of Materials (External)…		Generate a bill of materials for the current schematic using external generator
Generate Legacy Bill of Materials…		Generate a bill of materials for the current schematic (Legacy Generator)
Highlight Net	`	Highlight net under cursor
Highlight Nets		Highlight wires and pins of a net
Import Footprint Assignments…		Import symbol footprint assignments from .cmp file created by board editor
Import Graphics…	Ctrl + Shift + F	Import 2D drawing file
Increment Annotations From…		Increment a subset of reference designators starting at a particular symbol
Line Mode for Wires and Buses		Constrain drawing and dragging to horizontal, vertical, or 45-degree angle motions
Line Mode for Wires and Buses		Draw and drag at any angle
Line Mode for Wires and Buses	Shift + Space	Switch to next line mode
Line Mode for Wires and Buses		Constrain drawing and dragging to horizontal or vertical motions
Mark items excluded from simulation		Draw 'X’s over items which have been excluded from simulation
Next Symbol Unit		Open the next unit of the symbol
Previous Symbol Unit		Open the previous unit of the symbol
Remap Legacy Library Symbols…		Remap library symbol references in legacy schematics to the symbol library table
Repair Schematic		Run various diagnostics and attempt to repair schematic
Rescue Symbols…		Find old symbols in project and rename/rescue them

Action	Default Hotkey	Description
Do not Populate		Set the do not populate attribute
Exclude from Bill of Materials		Set the exclude from bill of materials attribute
Exclude from Board		Set the exclude from board attribute
Exclude from Simulation		Set the exclude from simulation attribute
Show Directive Labels		
Show ERC Errors		Show markers for electrical rules checker errors
Show ERC Exclusions		Show markers for excluded electrical rules checker violations
Show ERC Warnings		Show markers for electrical rules checker warnings
Show Hidden Fields		
Show Hidden Pins		
Net Navigator		Show/hide the net navigator
Show OP Currents		Show operating point current data from simulation
Show OP Voltages		Show operating point voltage data from simulation
Switch to PCB Editor		Open PCB in board editor
Simulator		Show simulation window for running SPICE or IBIS simulations.
Show Pin Alternate Icons		Show indicator icons for pins with alternate modes
Hierarchy Navigator	Ctrl + H	Show/hide the schematic sheet hierarchy navigator
Symbol Checker		Show the symbol checker window
Compare Symbol with Library		Show differences between schematic symbol and its library equivalent
Electrical Rules Checker		Show the electrical rules checker window

Action	Default Hotkey	Description
Delete Last Point		Delete the last point added to the current item
Draw Arcs		
Draw Bezier Curve		
Draw Circles		
Draw Rectangles		
Draw Rule Areas		
Draw Hierarchical Sheets	S	
Draw Sheet from Design Block		Copy design block into project as a sheet on current sheet
Draw Sheet from File		Copy sheet into project and draw on current sheet
Draw Tables		
Draw Text Boxes		
Import Sheet		Import sheet into project
Place Wire to Bus Entries	Z	
Place Directive Labels		
Place Design Block	Shift + B	Add selected design block to current sheet
Place Global Labels	Ctrl + L	
Place Hierarchical Labels	H	
Place Images		
Place Junctions	J	
Place Net Labels	L	
Place Next Symbol Unit		Place the next unit of the current symbol that is missing from the schematic
Place No Connect Flags	Q	

Action	Default Hotkey	Description
Sync Sheet Pins		Synchronize sheet pins and hierarchical labels
Sync Sheet Pins		Synchronize sheet pins and hierarchical labels
Draw Buses	B	
Draw Lines	I	
Draw Wires	W	
Switch Segment Posture	/	Switches posture of the current segment.
Undo Last Segment	Back	Walks the current line back one segment.
Unfold from Bus	C	Break a wire out of a bus
Assign Netclass…		Assign a netclass to nets matching a pattern
Autoplace Fields	O	Runs the automatic placement algorithm on the symbol's (or sheet's) fields
Break		Divide into connected segments
Change Symbol…		Assign a different symbol from the library
Change Symbols…		Assign different symbols from the library
Cleanup Sheet Pins		Delete unreferenced sheet pins
Edit Footprint…	F	
Edit Reference Designator…	U	
Edit Text & Graphics Properties…		Edit text and graphics properties globally across schematic
Edit Value…	V	
Mirror Horizontally	X	Flips selected item(s) from left to right
Mirror Vertically	Y	Flips selected item(s) from top to bottom
Pin Table…		Displays pin table for bulk editing of pins
Properties…	E	
Repeat Last Item	Ins	Duplicates the last drawn item

Action	Default Hotkey	Description
Slice		Divide into unconnected segments
Swap	Alt + S	Swap positions of selected items
Symbol Properties…		
Change to Directive Label		Change existing item to a directive label
Change to Global Label		Change existing item to a global label
Change to Hierarchical Label		Change existing item to a hierarchical label
Change to Label		Change existing item to a label
Change to Text		Change existing item to a text comment
Change to Text Box		Change existing item to a text box
De Morgan Conversion		Switch between De Morgan representations
Update Symbol…		Update symbol to include any changes from the library
Update Symbols from Library…		Update symbols to include any changes from the library
Drag	G	Move items while keeping their connections
Move	M	
Select Connection	Ctrl + 4	Select a complete connection
Select Node	Alt + 3	Select a connection item under the cursor
Navigate Back	Alt + Left	Move backward in sheet navigation history
Change Sheet		Change to provided sheet’s contents in the schematic editor
Enter Sheet		Display the selected sheet’s contents in the schematic editor
Navigate Forward	Alt + Right	Move forward in sheet navigation history
Leave Sheet	Alt + Back	Display the parent sheet in the schematic editor
Next Sheet	PgDn	Move to next sheet by number
Previous Sheet	PgUp	Move to previous sheet by number

Action	Default Hotkey	Description
Push Pin Number Size		Copy pin number size to other pins in symbol
Create Corner		
Remove Corner		
Properties…		Edit properies of design block
Delete Design Block		Remove the selected design block from its library
Save Selection as Design Block…		Create a new design block from the current selection
Save Current Sheet as Design Block…		Create a new design block from the current sheet
Design Blocks		Show/hide design blocks library
User-defined Signals…		Add, edit or delete user-defined simulation signals
New Analysis Tab…	Ctrl + N	Create a new tab containing a simulation analysis
Open Workbook…	Ctrl + O	Open a saved set of analysis tabs and settings
Probe Schematic…	P	Add a simulator probe
Run Simulation	R	
Save Workbook	Ctrl + S	Save the current set of analysis tabs and settings
Save Workbook As…	Ctrl + Shift + S	Save the current set of analysis tabs and settings to another location
Show SPICE Netlist		
Edit Analysis Tab…		Edit the current analysis tab’s SPICE command and plot setup
Stop Simulation		
Add Tuned Value…	T	Select a value to be tuned
Export Current Plot as CSV…		
Export Current Plot as PNG…		

Action	Default Hotkey	Description
Dotted Current/Phase		Draw secondary signal trace (current or phase) with a dotted line
Show Legend		
Draw Lines		Draw connected graphic lines
Draw Polygons		
Draw Text Boxes		
Move Symbol Anchor		
Draw Pins	P	
Draw Text		
Add Symbol to Schematic		Add the current symbol to the schematic
Copy		
Cut		
Delete Symbol		Remove the selected symbol from its library
Derive from Existing Symbol…		Create a new symbol, derived from an existing symbol
Duplicate Symbol		
Edit Symbol		Show selected symbol on editor canvas
Export Symbol as SVG…		Create SVG file from the current symbol
Export View as PNG…		Create PNG file from the current view
Import Symbol…		Import a symbol to the current library
New Symbol…	Ctrl + N	Create a new symbol in an existing library
Paste Symbol		

Action	Default Hotkey	Description
Rename Symbol...		
Save Library As...	Ctrl + Shift + S	Save the current library to a new file
Save As...		Save the current symbol to a different library or name
Save Copy As...		Save a copy of the current symbol to a different library or name
Set Unit Display Name...		Set the display name for a particular unit in a multi-unit symbol
Show Pin Electrical Types		Annotate pins with their electrical types
Show Hidden Fields		
Show Hidden Pins		
Show Pin Numbers		Annotate pins with their numbers
Synchronized Pins Mode		Synchronized Pins Mode When enabled propagates all changes (except pin numbers) to other units. Enabled by default for multiunit parts with interchangeable units.
Update Symbol Fields...		Update symbol to match changes made in parent symbol

Common

The actions below are available across KiCad, including in the Schematic Editor. Hotkeys can be assigned to any of these actions in the **Hotkeys** section of the preferences.

Action	Default Hotkey	Description
Refresh Plugins		Reload all python plugins and refresh plugin menus
Exclude Marker		Mark current violation in Checker window as an exclusion
Next Marker		
Previous Marker		

Action	Default Hotkey	Description
Add Library…		Add an existing library folder
Center Justify		Center-justify fields and text items
Pan to Center Selected Objects		
Collapse All		
Click	Return	Performs left mouse button click
Double-click	End	Performs left mouse button double-click
Cursor Down	Down	
Cursor Down Fast	Ctrl + Down	
Cursor Left	Left	
Cursor Left Fast	Ctrl + Left	
Cursor Right	Right	
Cursor Right Fast	Ctrl + Right	
Cursor Up	Up	
Cursor Up Fast	Ctrl + Up	
Grid Origin…		Set the grid origin point
Edit Grids…		Edit grid definitions
Expand All		
Switch to Fast Grid 1	Alt + 1	
Switch to Fast Grid 2	Alt + 2	
Cycle Fast Grid	Alt + 4	
Switch to Next Grid	N	
Switch to Previous Grid	Shift + N	
Reset Grid Origin		
Grid Origin		Place the grid origin point

Action	Default Hotkey	Description
Inactive Layer View Mode (3-state)	H	Cycle inactive layers between normal, dimmed, and hidden
Inches		
Left Justify		Left-justify fields and text items
Focus Library Tree Search Field	Ctrl + L	
Snap to Objects on the Active Layer Only		Enables snapping to objects on the active layer only
Snap to Objects on All Layers		Enables snapping to objects on all visible layers
Toggle Snapping Between Active and All Layers	Shift + S	Toggles between snapping on all visible layers and only the active area
Millimeters		
Mils		
New…	Ctrl + N	Create a new document in the editor
New Library…		Create a new library folder
Open…	Ctrl + O	Open existing document
Open in file explorer…		Open a library file with system file explorer
Edit in a Text Editor…		Open a library file with a text editor
Page Settings…		Settings for paper size and title block info
Pan Down	Shift + Down	
Pan Left	Shift + Left	
Pan Right	Shift + Right	
Pan Up	Shift + Up	
Pin Library		Keep the library at the top of the list
Plot…		

Action	Default Hotkey	Description
Revert		Throw away changes
Right Justify		Right-justify fields and text items
Save	Ctrl + S	Save changes
Save All		Save all changes
Save As...	Ctrl + Shift + S	Save current document to another location
Save a Copy...		Save a copy of the current document to another location
Select Columns...		
3D Viewer	Alt + 3	Show 3D viewer window
Show Context Menu		Perform the right-mouse-button action
Show Datasheet	D	Open the datasheet in a browser
Footprint Library Browser		
Footprint Editor		Create, delete and edit board footprints
Library Tree		
Switch to Project Manager		Show project window
Properties		Show/hide the properties manager
Symbol Library Browser		
Symbol Editor		Create, delete and edit schematic symbols
Draw Bounding Boxes		
Always Show Crosshairs	Ctrl + Shift + X	Display crosshairs even when not drawing objects
Full-Window Crosshairs		Switch display of full-window crosshairs
Show Grid		Display background grid in the edit window
Grid Overrides	Ctrl + Shift + G	Enables item-specific grids that override the current grid

Action	Default Hotkey	Description
Update PCB from Schematic...	F8	Update PCB with changes made to schematic
Update Schematic from PCB...		Update schematic with changes made to PCB
Center on Cursor	F4	
Zoom to Objects	Ctrl + Home	
Zoom to Fit	Home	
Zoom to Selected Objects		
Zoom In at Cursor	F1	
Zoom In		
Zoom In Horizontally		Zoom in horizontally the plot area
Zoom In Vertically		Zoom in vertically the plot area
Zoom Out at Cursor	F2	
Zoom Out		
Zoom Out Horizontally		Zoom out horizontally the plot area
Zoom Out Vertically		Zoom out vertically the plot area
Refresh	F5	
Zoom to Selection	Ctrl + F5	
Embedded Files		Manage embedded files
Extract File		Extract an embedded file
Remove File		Remove an embedded file
Cancel		Cancel current tool
Copy	Ctrl + C	Copy selected item(s) to clipboard
Copy as Text	Ctrl + Shift + C	Copy selected item(s) to clipboard as text

Action	Default Hotkey	Description
Find and Replace	Ctrl + Alt + F	
Find Next	F3	
Find Next Marker	Ctrl + Shift + F3	
Find Previous	Shift + F3	
Finish	End	Finish current tool
Measure Tool	Ctrl + Shift + M	Interactively measure distance between points
Paste	Ctrl + V	Paste item(s) from clipboard
Paste Special...		Paste item(s) from clipboard with options
Redo	Ctrl + Y	
Replace All		
Replace and Find Next		
Search	Ctrl + G	Show/hide the search panel
Select All	Ctrl + A	Select all items on screen
Undo	Ctrl + Z	
Unselect All	Ctrl + Shift + A	Unselect all items on screen
Select Row(s)		Select complete row(s) containing the current selected cell(s)
Select Column(s)		Select complete column(s) containing the current selected cell(s)
Select Table		Select parent table of selected cell(s)
Select item(s)		
About KiCad		
Configure Paths...		Edit path configuration environment variables
Donate		Open "Donate to KiCad" in a web browser
Get Involved		Open "Contribute to KiCad" in a web browser
Getting Started with KiCad		Open “Getting Started in KiCad” guide for beginners

Action	Default Hotkey	Description
List Hotkeys…	Ctrl + F1	Displays current hotkeys table and corresponding commands
Preferences…	Ctrl + ,	Show preferences for all open tools
Report Bug		Report a problem with KiCad
Manage Design Block Libraries…		Edit the global and project design block library lists
Manage Footprint Libraries…		Edit the global and project footprint library lists
Manage Symbol Libraries…		Edit the global and project symbol library lists
Add Column After		Insert a new table column after the selected cell(s)
Add Column Before		Insert a new table column before the selected cell(s)
Add Row Above		Insert a new table row above the selected cell(s)
Add Row Below		Insert a new table row below the selected cell(s)
Delete Column(s)		Delete columns containing the currently selected cell(s)
Delete Row(s)		Delete rows containing the currently selected cell(s)
Merge Cells		Turn selected table cells into a single cell
Unmerge Cells		Turn merged table cells back into separate cells.