Schematic Editor

The KiCad Team

Table of Contents

Introduction to the KiCad Schematic Editor 2
- Description 2
- Configuration Initiale 2
- The Schematic Editor User Interface 4
- Navigating the editing canvas 4
- Raccourcis clavier 5
- Mouse operations and selection 5
- Left toolbar display controls 6

Création et édition de schémas 7
- Introduction 7
- Schematic editing operations 7
- Grids 9
- Snapping 9
- Working with symbols 9
- Reference Designators and Symbol Annotation 15
- Electrical Connections 17
- Graphical items 26
- Schematic Setup 28
- Rescuing cached symbols 28

Schématiques hiérarchiques 30
- Introduction 30
- Adding sheets to a design 30
- Navigating between sheets 32
- Electrical connections between sheets 32
- Hierarchical design examples 34

Inspecting a schematic 37
- Find tool 37
- Net highlighting 38
- Cross-probing from the PCB 38
- Vérification des règles électriques (ERC) 38

Assigning Footprints 45
- Assigning Footprints in Symbol Properties 45
- Assigning Footprints While Placing Symbols 47
- Assigning Footprints with the Footprint Assignment Tool 48

Manuel de référence

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

Copyright

This document is Copyright © 2010-2022 by its contributors as listed below. You may distribute it and/or modify it under the terms of either the GNU General Public License (http://www.gnu.org/licenses/gpl.html), version 3 or later, or the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), version 3.0 or later.

Toutes les marques apparaissant dans ce document appartiennent à leurs propriétaires respectifs.

Contributeurs

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

Traduction

Marc Berlioux <marc.berlioux@gmail.com>, 2015-2016

Retours

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

Introduction to the KiCad Schematic Editor

Description

The KiCad Schematic Editor is a schematic capture software distributed as a part of KiCad and available under 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 drawing, control, layout, library management and access to the PCB design software are carried out within the editor itself.

The KiCad Schematic Editor is intended to cooperate with the KiCad PCB Editor, which is KiCad's printed circuit design software. It can also export netlist files, which lists 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:

- La vérification des règles électriques ou ERC (Electrical Rules Check), pour le contrôle des connexions manquantes ou incorrectes.
- L'exportation de fichiers de tracé en plusieurs formats (Postscript, PDF, HPGL, SVG)
- Bill of Materials generation (via Python or XSLT scripts, which allow many flexible formats).

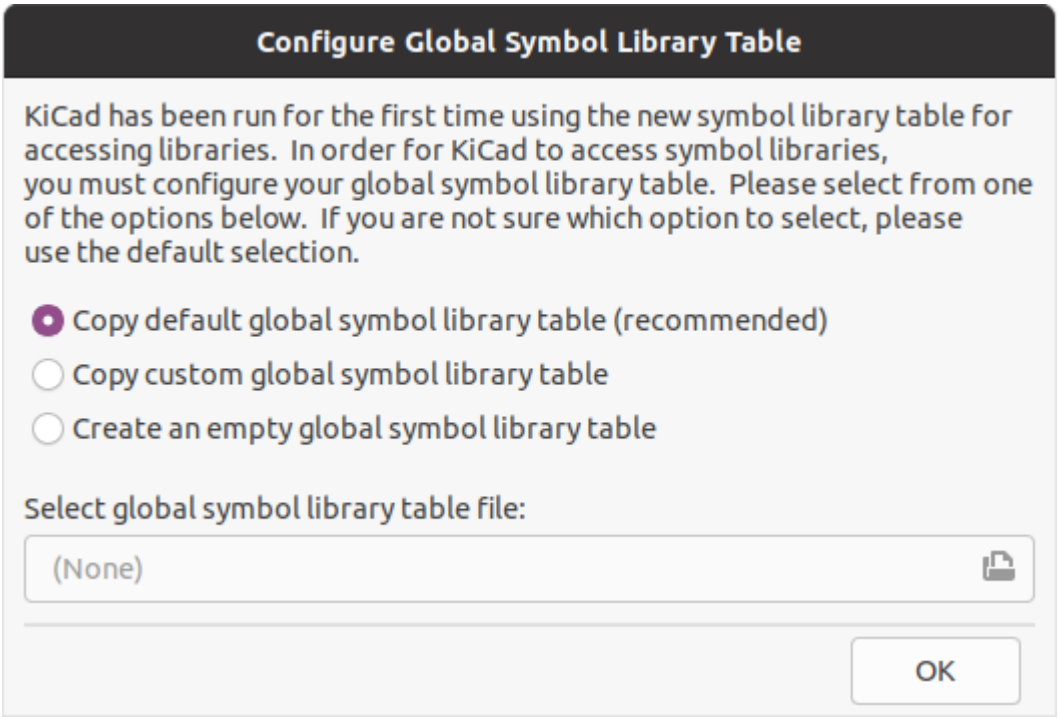
The Schematic Editor supports multi-sheet schematics in several ways:

- 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.

Configuration Initiale

When the Schematic Editor is run for the first time, if the the global symbol library table file `sym-lib-table` is not found in the KiCad configuration folder then KiCad will ask how to create this file:



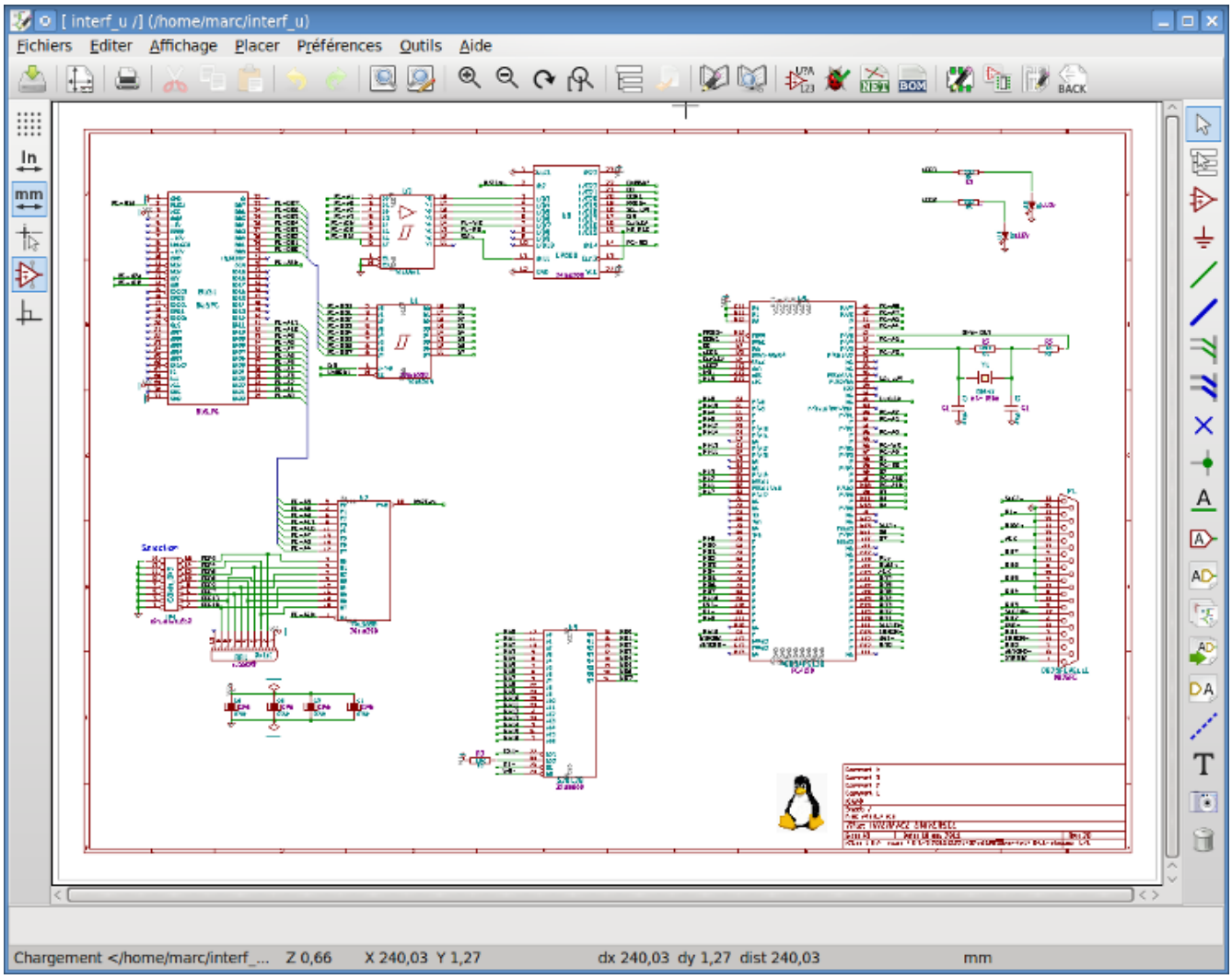
The first option is recommended (**Copy default global symbol library table (recommended)**). The default symbol library table includes all of the standard symbol libraries that are installed as part of KiCad.

If this option is disabled, KiCad was unable to find the default global symbol library table. This probably means you did not install the standard symbol libraries with KiCad, or they are not installed where KiCad expects to find them. On some systems the KiCad libraries are installed as a separate package.

- If you have installed the standard KiCad symbol libraries and want to use them, but the first option is disabled, select the second option and browse to the `sym-lib-table` file in the directory where the KiCad libraries were installed.
- If you already have a custom symbol library table that you would like to use, select the second option and browse to your `sym-lib-table` file.
- If you want to construct a new symbol library table from scratch, select the third option.

Symbol library management is described in more detail later.

The Schematic Editor User Interface



The main Schematic Editor user interface is shown above. The center contains the main editing canvas, which is surrounded by:

- Top toolbars (file management, zoom tools, editing tools)
- Left toolbar (display options)
- Message panel and status bar at bottom
- Right panel (drawing and design tools)

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

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

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

Raccourcis clavier

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\6.0\user.hotkeys`
- Linux: `~/.config/kicad/6.0/user.hotkeys`
- macOS: `~/Library/Preferences/kicad/6.0/user.hotkeys`

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

Mouse operations and selection

Selecting items in the editing canvas is done with the left mouse button. Single-clicking on an object will select it and dragging will perform a box selection. A box selection from left to right will only select items that are fully inside the box. A box 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.

The selection action can be modified by holding modifier keys while clicking or dragging. The following modifier keys apply when clicking to select single items:

Modifier Keys (Windows)	Modifier Keys (Linux)	Modifier Keys (macOS)	Selection Effect
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.
Ctrl	Ctrl	Cmd	Highlight the net of the selected copper item.

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

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

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.

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.

Left toolbar display controls

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

	Turns grid display on/off. **Note:** by default, hiding the grid will disable grid snapping. This behavior can be changed in the Display Options section of Preferences.
in mil mm	Display/entry of coordinates and dimensions in inches, mils, or millimeters.
	Switches between full-screen and small editing cursor (crosshairs).
	Turns invisible pin display on/off.
	Switches between free angle and horizontal/vertical placement of new wires, buses, and graphical lines.

Création et édition de schémas

Introduction

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:

- Valider un ensemble de règles (Vérification des règles électriques (ERC)) pour détecter les erreurs et omissions.
- Automatically generating a bill of materials.
- Générer une netliste pour des logiciels de simulation, comme SPICE.
- Defining a circuit for transferring to PCB layout.

A schematic mainly consists of symbols, wires, labels, junctions, buses and power ports. 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.

	Selection tool (the default tool)
	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 Connect" flag. These flags should be placed on symbol pins which are meant to be left unconnected. "No connect" flags indicate to the Electrical Rule Checker that the pin is intentionally unconnected and not an error.
	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. Local labels connect items located **in the same sheet**. For connections between two different sheets, use global or hierarchical labels.
	Place a global label. All global labels with the same name are connected, even when located on different sheets.
	Place a hierarchical label. 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.
	Place a hierarchical subsheet. You must specify the file name for this subsheet.
	Import a hierarchical pin from a subsheet. This command can be executed only on hierarchical subsheets. It will create hierarchical pins corresponding to hierarchical labels placed in the target subsheet.
	Draw 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 text comment.
	Place a bitmap image.
	Delete clicked items.

Grids

In the Schematic Editor the cursor always moves over a grid. The grid can be customized:

- Size can be changed using the right click menu or using **View → Grid Properties...**.
- Color can be changed in the **Colors** page of the **Preferences** dialog (menu **Preferences → General Options**).
- Visibility can be switched using the left-hand toolbar button.

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.

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!**

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	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**.

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 tool snaps to pins even when grid snapping is disabled. Both grid and pin snapping can be disabled while moving the mouse by using the modifier keys in the table below.

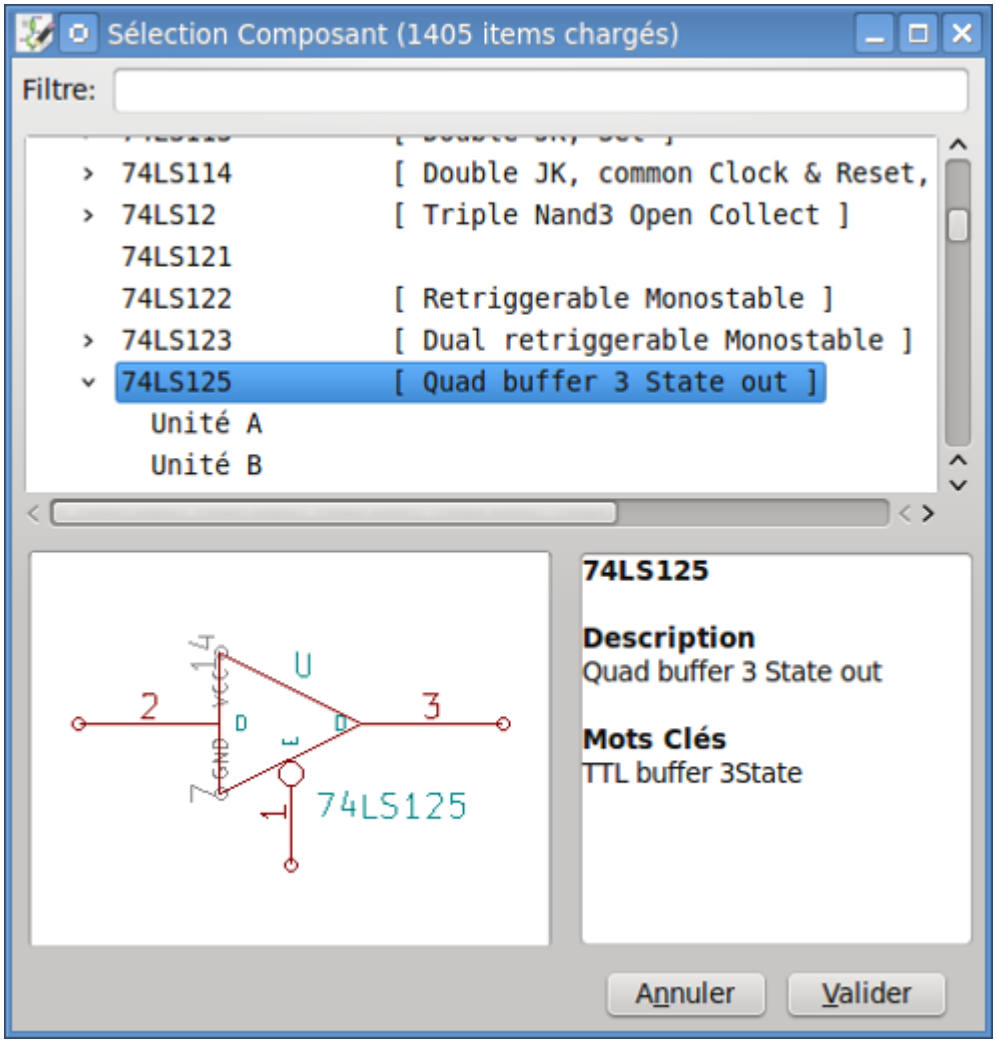
NOTE	On Apple keyboards, use the `Cmd` key instead of `Ctrl`.

Modifier Key	Effect
`Ctrl`	Disable grid snapping.
`Shift`	Disable snapping wires to pins.

Working with symbols

Placing symbols

To load a symbol into your schematic you can use the icon . A dialog box allows you to type the name of the symbol to load.



The Choose Symbols dialog will filter symbols by name, keywords, and description according to what you type into the search field.

Some advanced filters are available:

- **Wildcards:** use the characters `?` and `*` respectively to mean "any single character or no characters" and "any number of any characters, including none".
- **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}

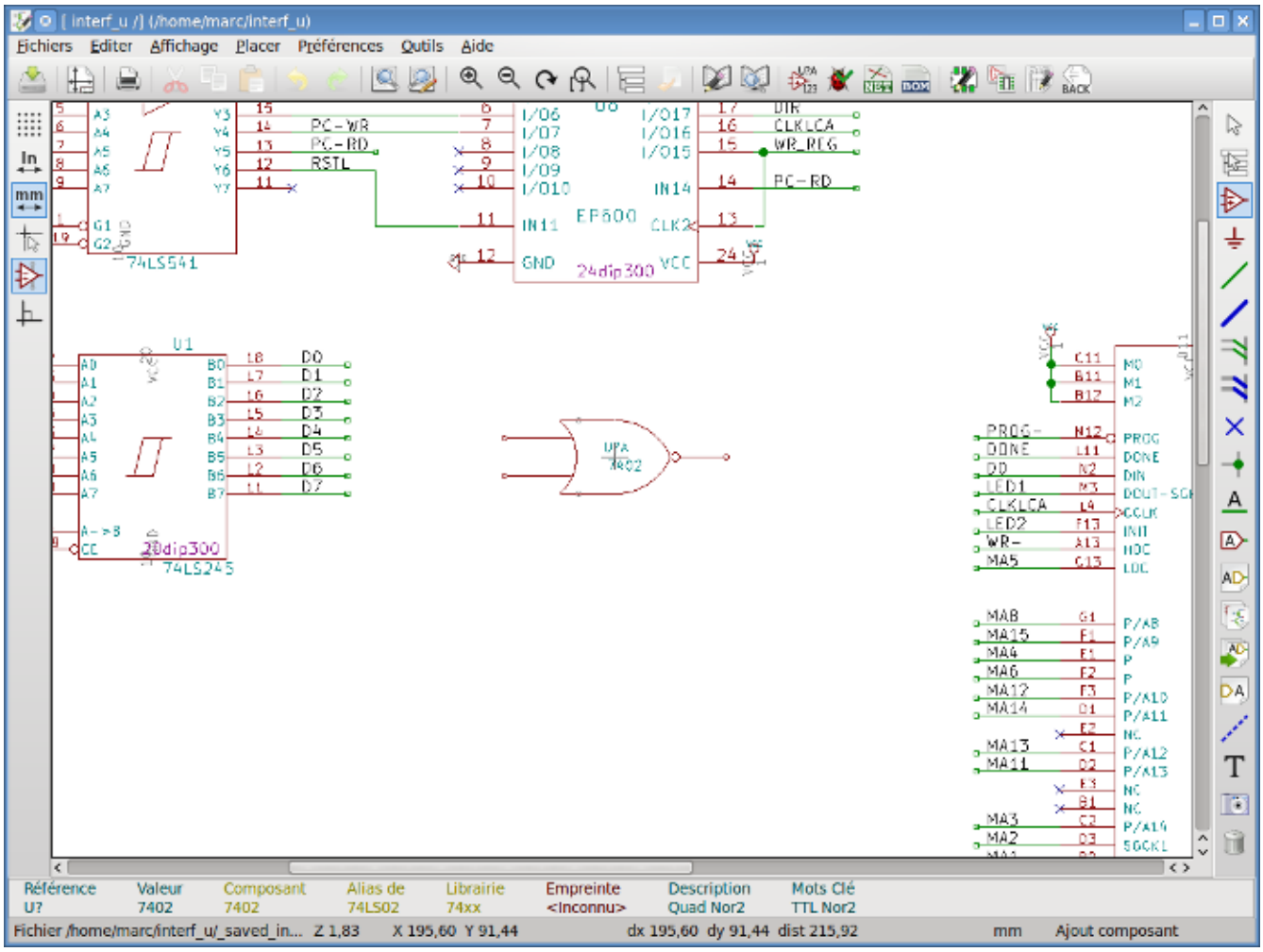
ki	mi	gi	ti
2^{10}	2^{20}	2^{30}	2^{40}

- **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.

Here is a symbol during 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 ports

A power port 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.

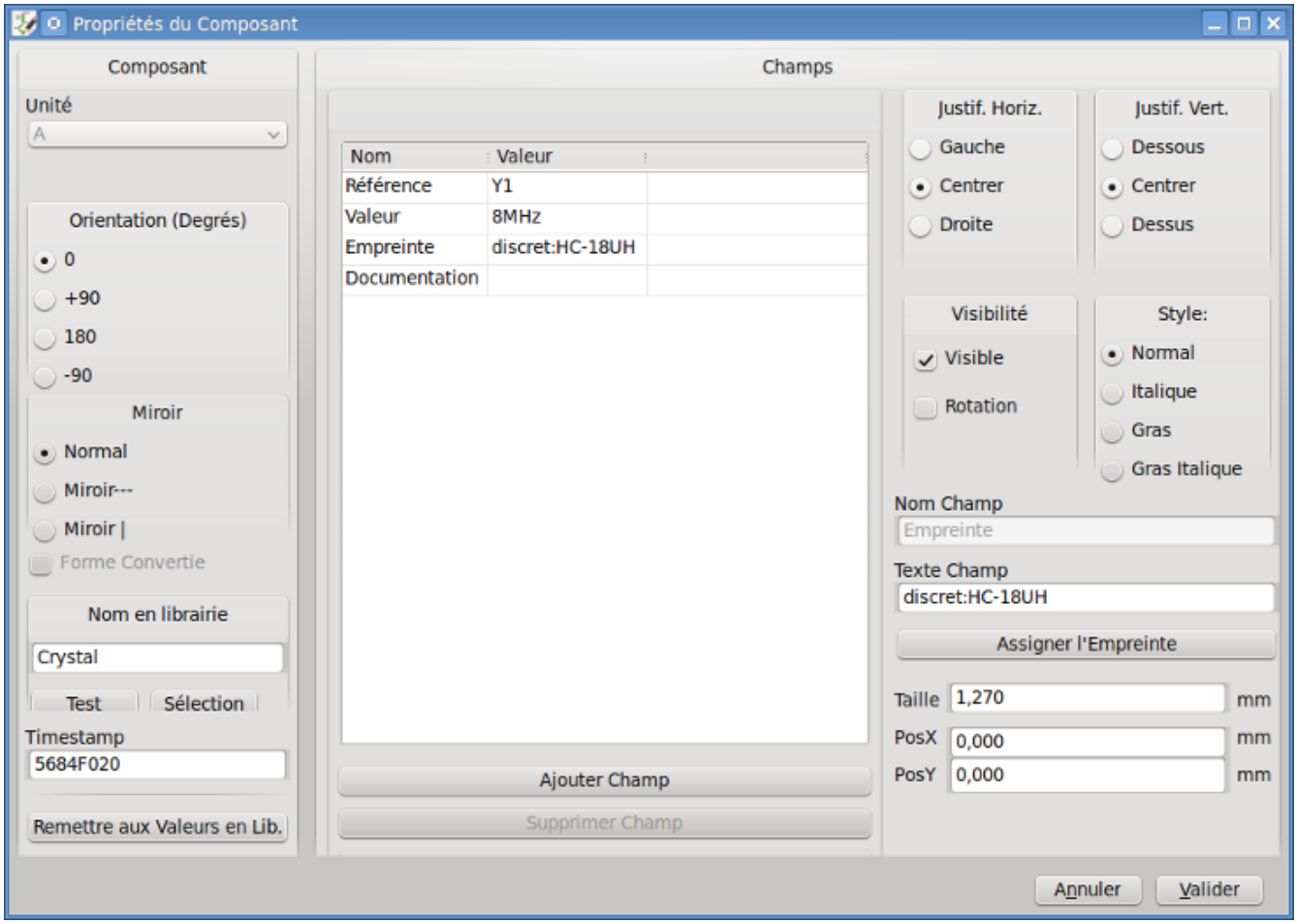
The **Drag** tool moves the symbol without breaking wired connections to its pins, and therefore moves the connected wires as well.

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

A symbol's fields can be edited in the symbol's Properties window. Open the Symbol Properties window for a symbol with the E hotkey or by double-clicking on the symbol.



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.

Each field can be visible or hidden, and displayed horizontally or vertically. 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.

The position and orientation properties of each field may be hidden in this dialog. They can be shown by right-clicking on the column header of the fields table and enabling the "Orientation", "X Position", and/or "Y Position" columns. Other columns can be shown or hidden as desired.

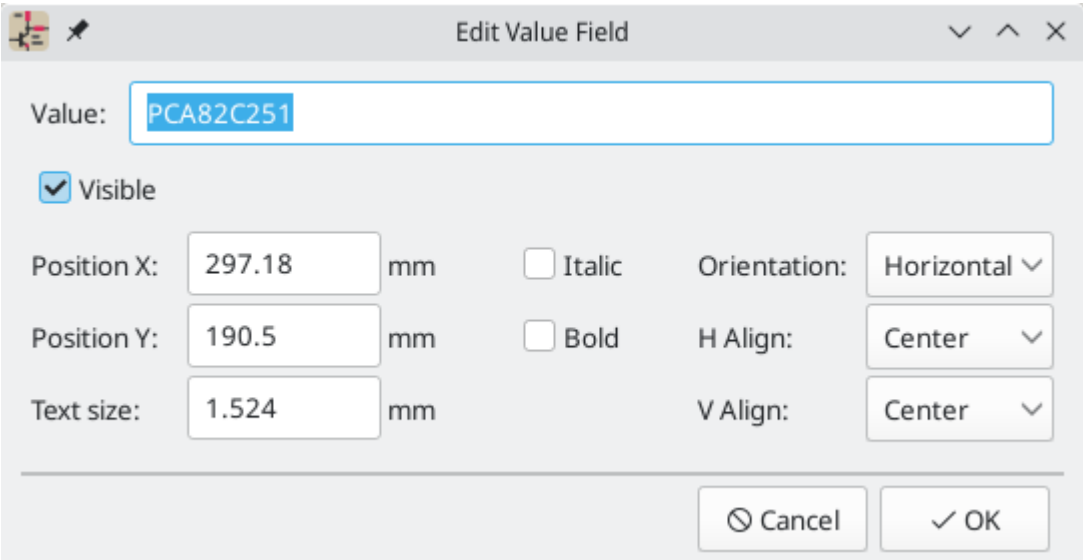
The "Update Symbol from Library..." button is used to update the schematic's copy of the symbol to match the copy in the library. The "Change Symbol..." button is used to swap the current symbol to a different symbol in the library.

"Edit Symbol..." opens the Symbol Editor to edit the copy of the symbol in the schematic. Note that the original symbol in the library will not be modified. The "Edit Library Symbol..." button opens the Symbol Editor to edit the original symbol in the library. In this case, the symbol in the schematic will not be modified until the user clicks the "Update Symbol from Library..." button.

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.

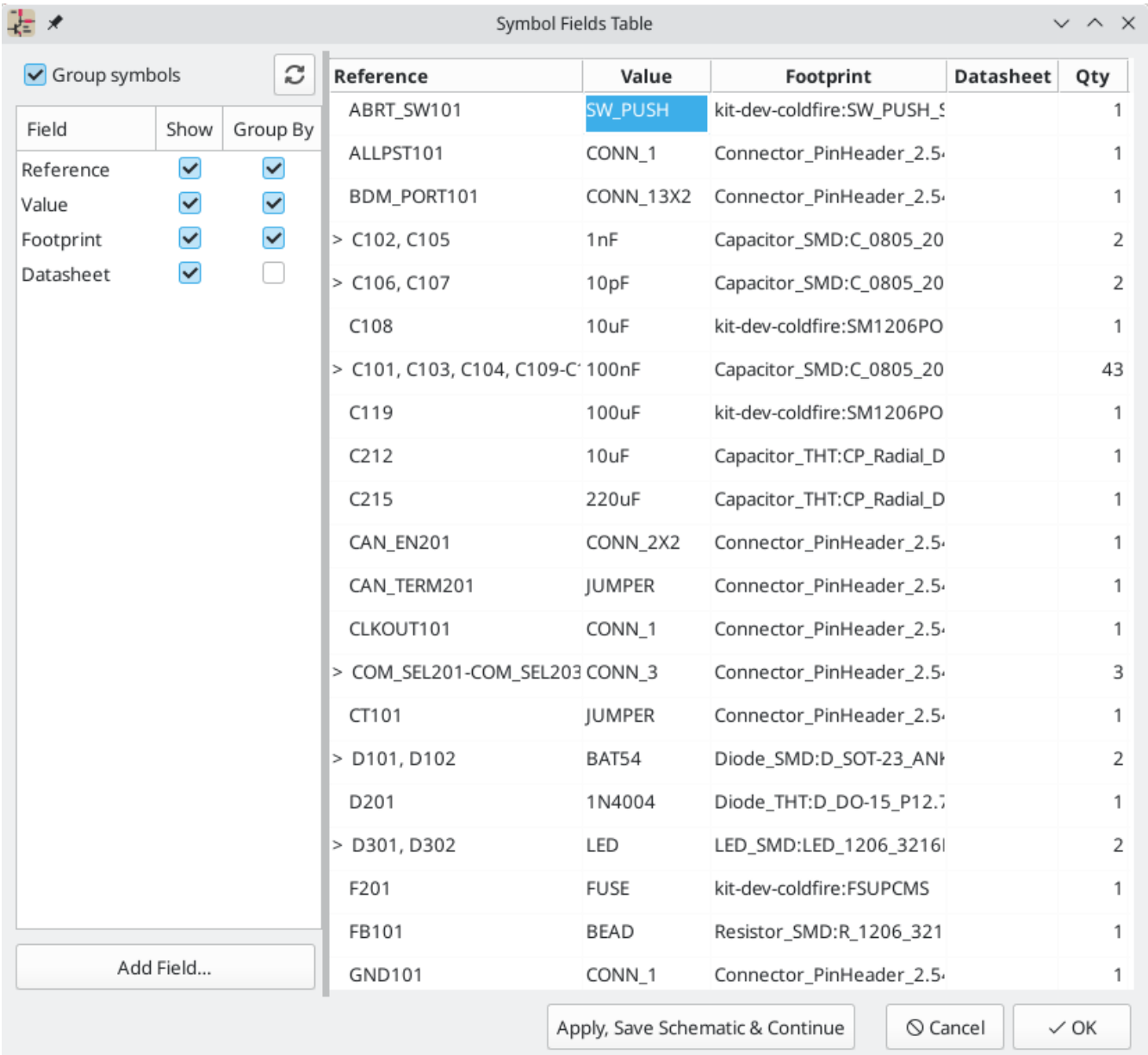


The options in this dialog are the same as those in the full Symbol Properties dialog, but are specific to a single field.

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.

Symbol Fields Table

The Symbol Fields Table allows you to view and modify field values for all symbols in a spreadsheet interface. You can open the Symbol Fields Table with the button.



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

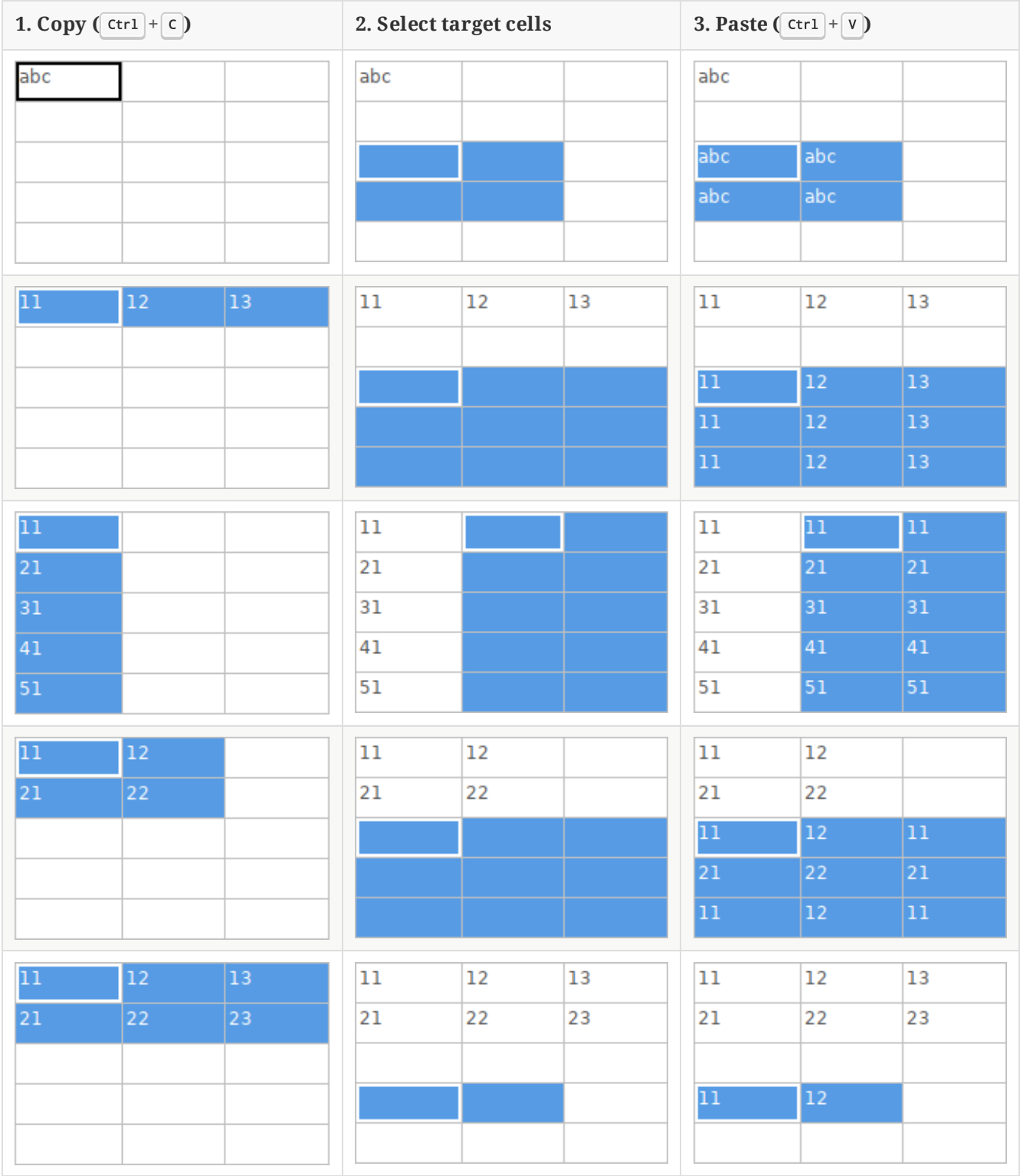
Any symbol field can be shown or hidden using the **Show** checkboxes on the left, or by right-clicking on the header of the table. New symbol fields can be added using the **Add Field...** button.

Similar symbols can optionally be grouped by any symbol field using the **Group By** checkboxes. Grouped symbols are shown in a single row in the table. The grouped row can be expanded to show the individual symbols by clicking the arrow at the left of the row.

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.



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.

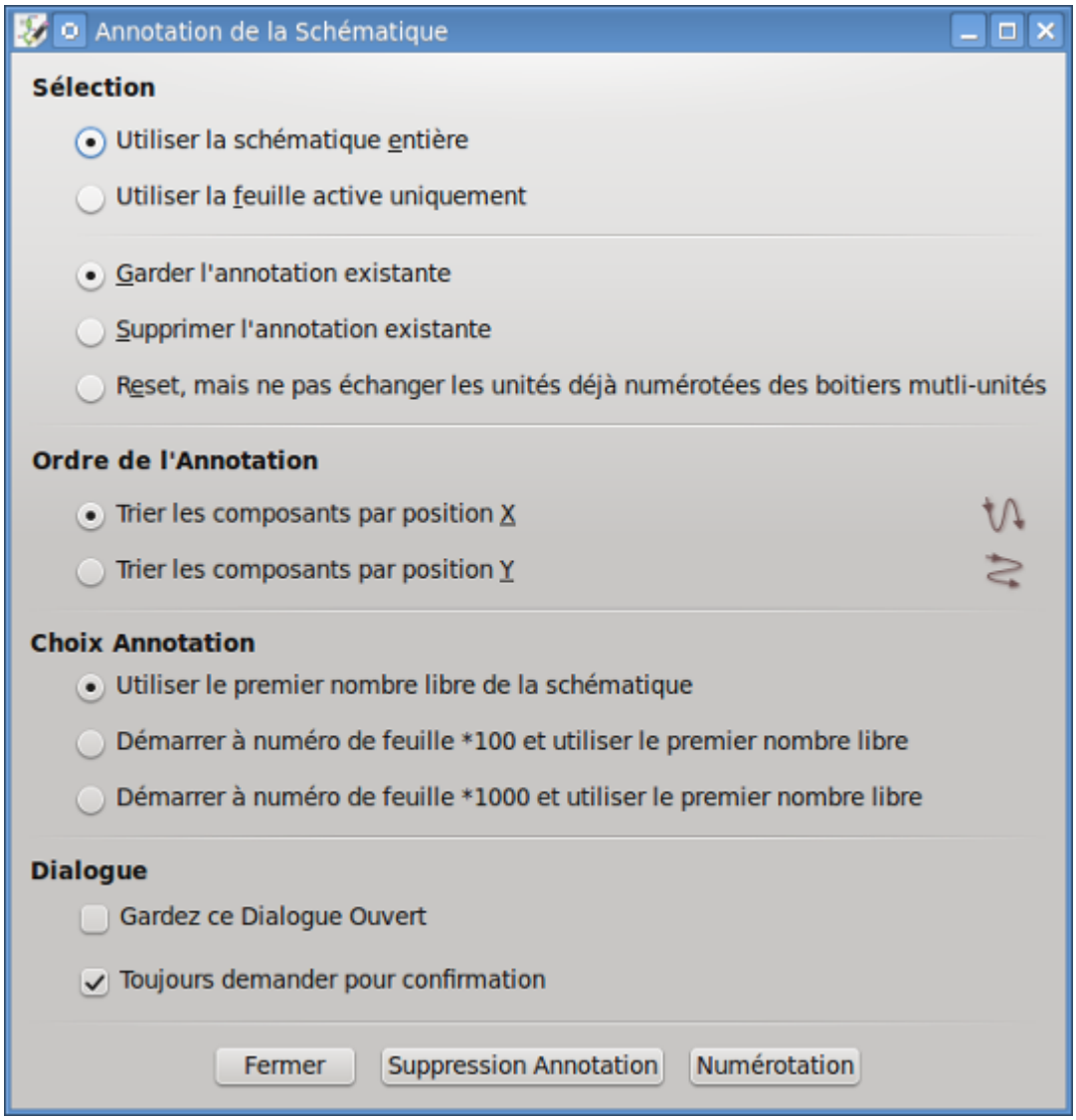
In KiCad, reference designators consist of a letter indicating the type of component (`R` for resistor, `C` for capacitor, `U` for IC, etc.) followed by a number. If the symbol has multiple units then the reference designator will also have a trailing letter indicating the unit. Symbols that don't have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

Reference designators can set manually by editing a symbol's reference designator field, or automatically using the Annotation tool.

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

Outil d'annotation

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



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.

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.

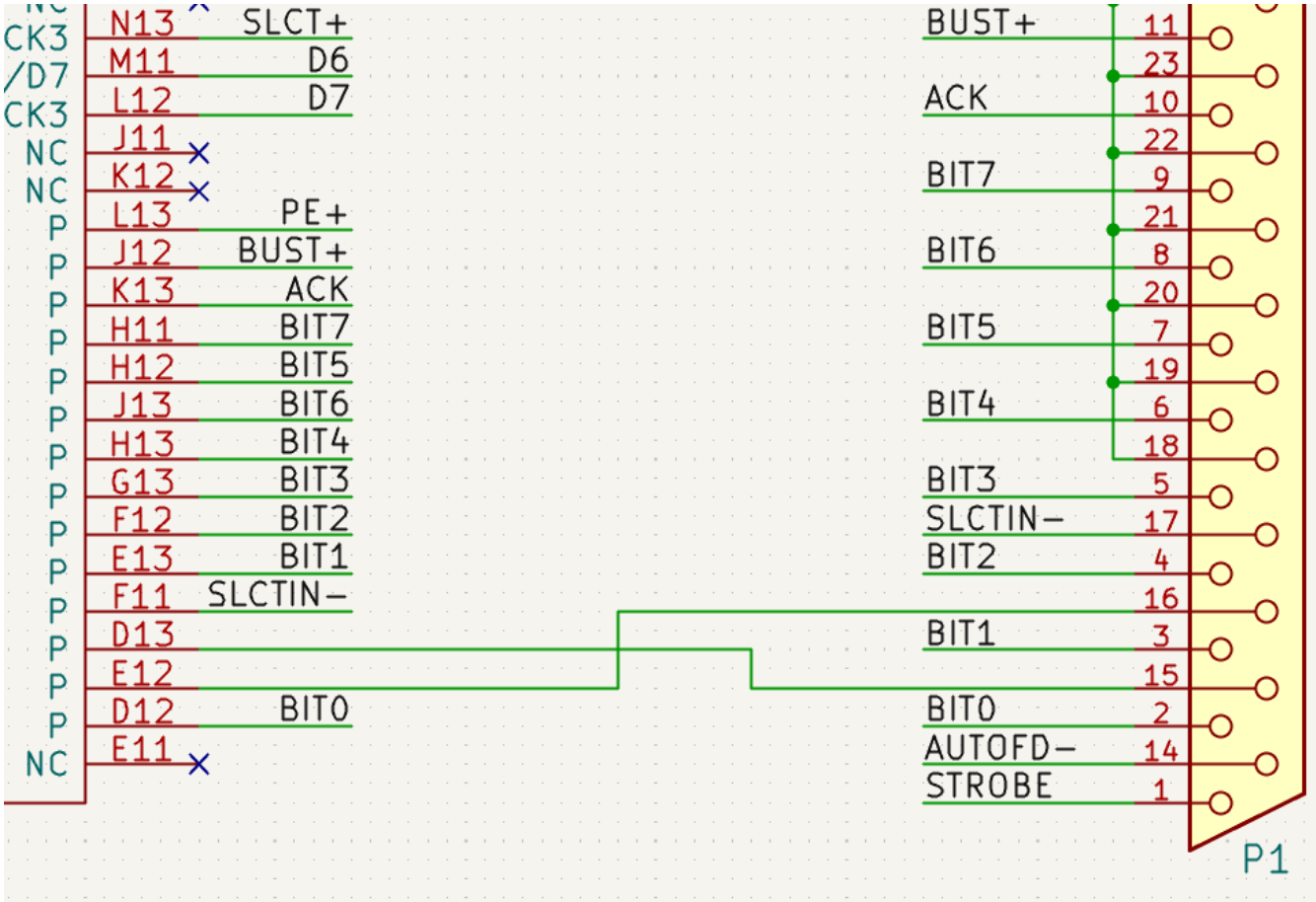
Numbering: Selects the starting point for numbering reference designators. The lowest unused number above the starting point is picked for each reference designator. The starting point can be an arbitrary number (typically zero), or it can be the sheet number multiplied by 100 or 1000 so that each part's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

There are two primary ways to establish connections: wires and labels. Both are shown in the schematic below.



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

This section will also discuss two special types of symbols that can be added with the "Power port" button on the right toolbar:

- **Power ports**: symbols for connecting wires to a power or ground net.
-

PWR_FLAG: a specific symbol for indicating that a net is powered when it is not connected to a power output pin (for example, a power net that is supplied by an off-board connector).

Label Connections

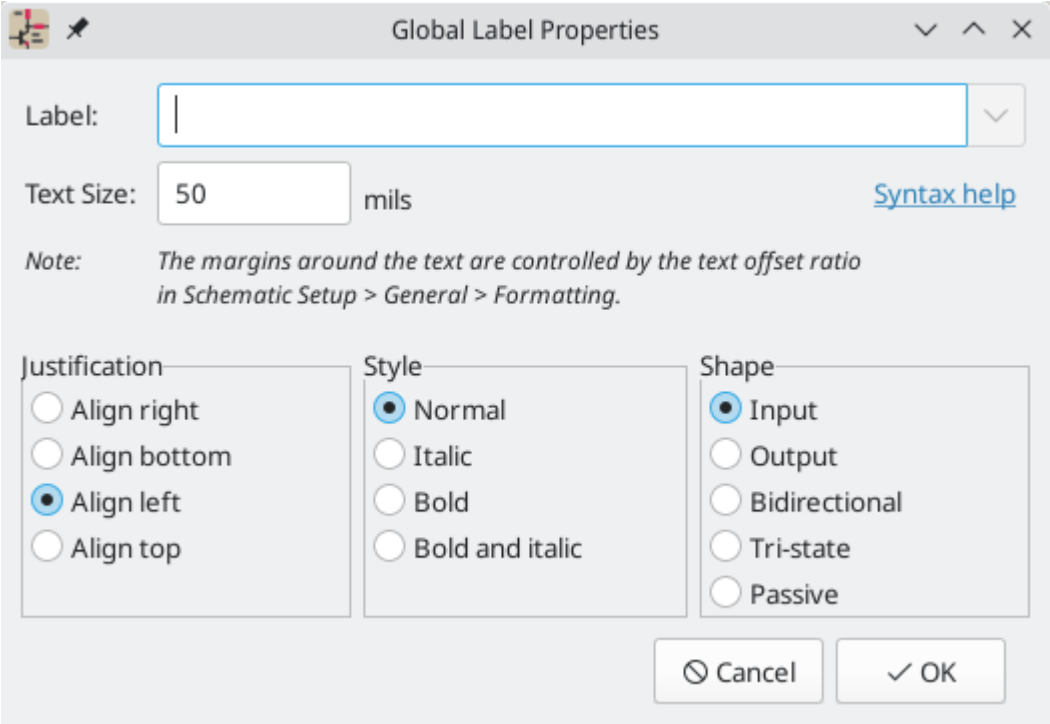
Labels are used to assign net names to wires and pins. Wires with the same net name are considered to be connected. A net can only have one name. If two different labels are placed on the same net, an ERC violation will be generated. Only one of the net names will be used in the netlist. The final net 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 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 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.

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



The **Label** field sets the label's text, which determines the net that the label assigns to its attached wire. Label text supports markup for overbars, subscripts, etc., as well as variable substitution. Use the **Syntax help** link in the dialog for a summary.

Justification sets the position of the label's connection point relative to the label's text. For example, when **Align right** is selected the connection point will be to the right of the text.

Text size and **Style** control the appearance of the label's text. **Shape** controls the shape of the outline around the label; this is purely visual and has no electrical consequence. Local labels do not have an outline, and therefore do not have **Shape** options.

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

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.

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.

Wire Connections

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.

Wires can be moved 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.

If a segment is selected or the cursor is over the middle of a wire, the move/drag action will move the entire segment. If the cursor is over a corner or wire end, the move/drag action will act on one end of the segment.

Wire Junctions

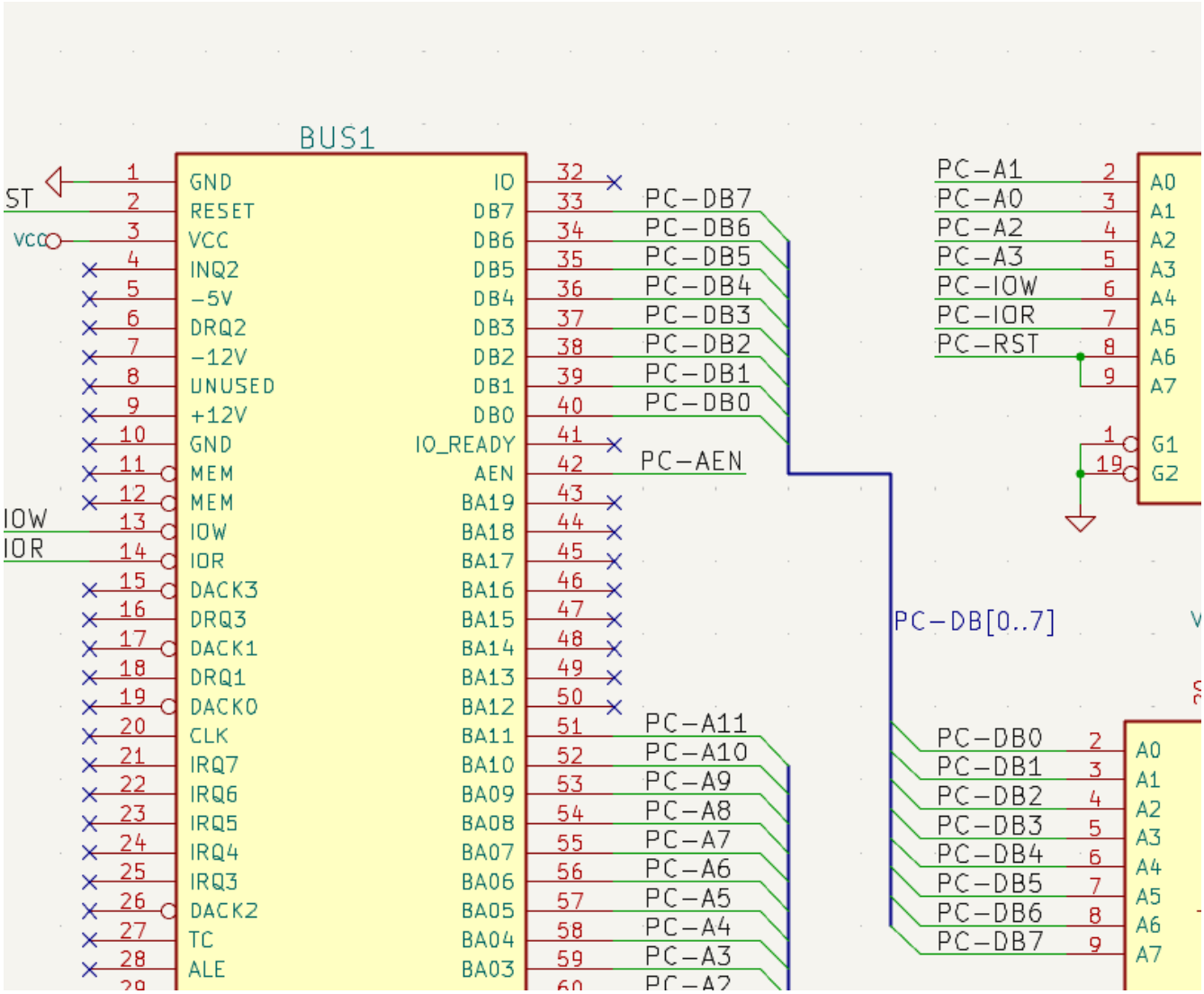
Wires that cross are not implicitly connected. It is necessary to join them by explicitly adding a junction dot if a connection is desired (button in the right toolbar). Junction dots will be automatically added to wires that start or end on top of an existing wire.

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

Bus Connections

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.



Membres d'un 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.

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.

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.

Connexions entre membres de 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.

Dans l'exemple ci-dessus, les connexions sont faites par des labels placés sur les fils connectés aux pins. Les entrées de bus (segments de fil à 45 degrés) sont purement décoratifs, et ne sont pas nécessaires pour établir des connexions logiques.

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.
- Dessinez le fil sous le premier label. Ensuite, utilisez la commande de répétition pour placer les autres fils sous les autres labels.
- Au besoin, placez les entrées de bus de la même façon (placez la première entrée, puis utilisez la commande de répétition).

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

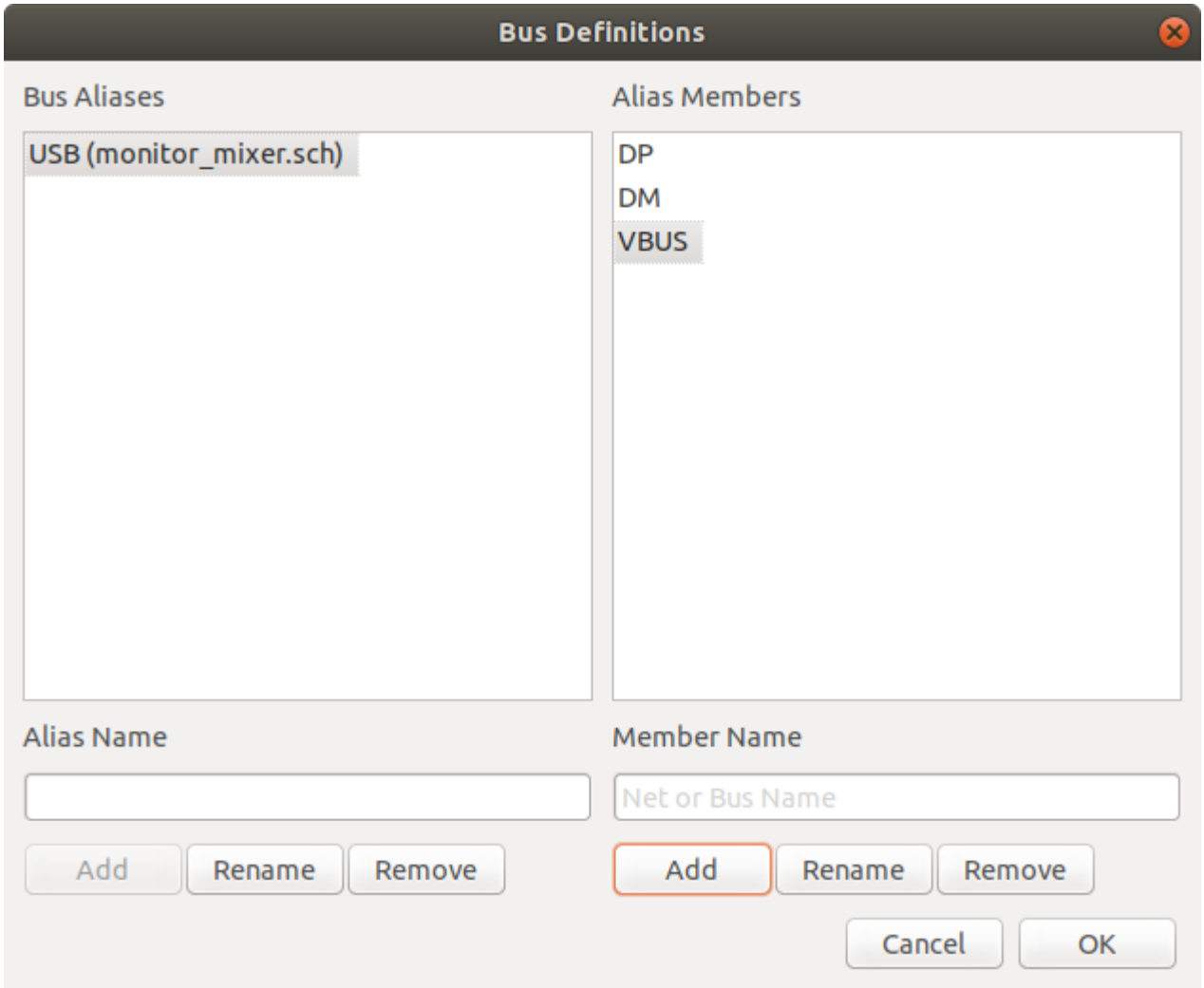
The unfold tool allows you to quickly break out signals from a bus. To unfold a signal, right-click on a bus object (a bus wire, etc) and choose **Unfold from Bus**. Alternatively, use the **Unfold Bus** hotkey (default: `C`) when the cursor is over a bus object. The menu allows you to select which bus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



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`.

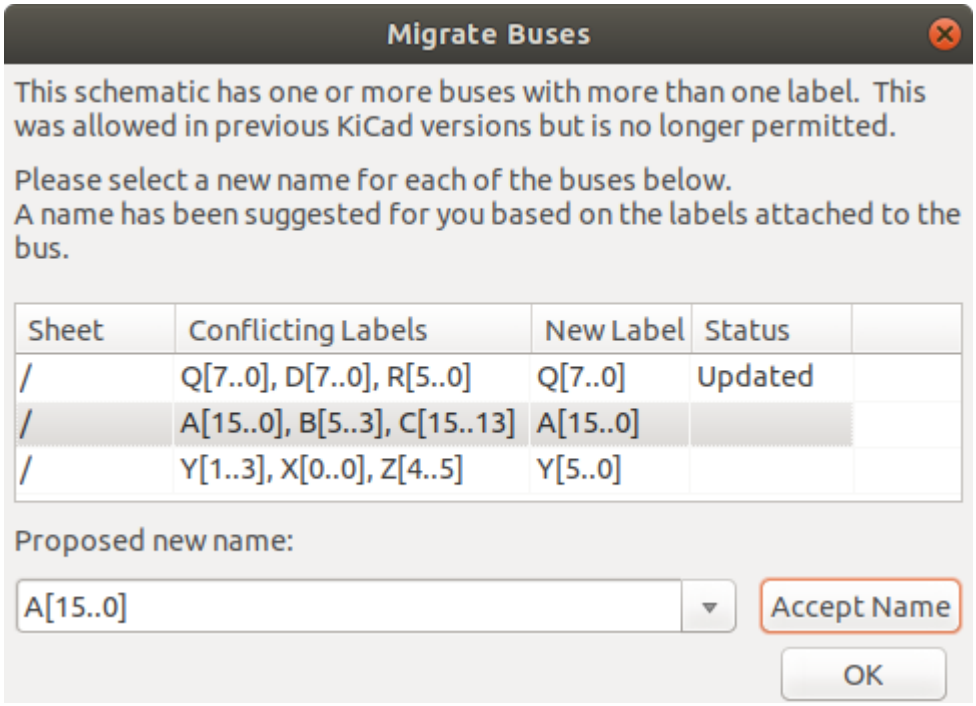
After defining an alias, it can be used in a group bus label by putting the alias name inside the curly braces of the group bus: `{USB}`. This has the same effect as labeling the bus `{DP DM VBUS}`. You can also add a prefix name to the group, such as `USB1{USB}`, which results in nets such as `USB1.DP` as described above. 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. Any aliases created in a given schematic sheet are available to use in any other schematic sheet that is in the same hierarchical design.

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.



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.

Hidden Power Pins

When the power pins of a symbol are visible, they must be connected, as with any other signal. However, symbols such as gates and flip-flops 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.

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.

It may be necessary to join power nets of different names (for example, `GND` in TTL components and `VSS` in MOS components). To accomplish this, add a power port symbol for each net and connect them with a wire.

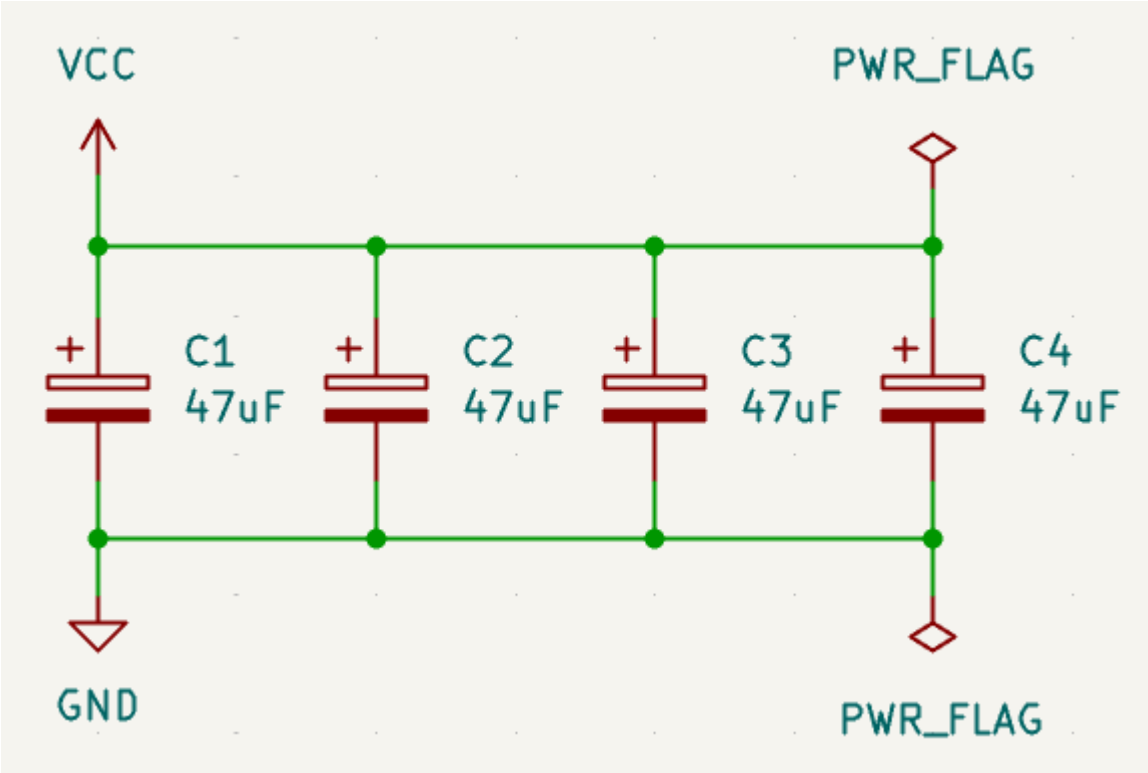
If hidden power pins are used, it is not recommended to use local labels for power connection, as they will not connect to hidden power pins on other sheets.

NOTE	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** outside of power port symbols, and is only supported for compatibility with legacy designs and symbols.

Power Ports

Power ports are symbols that are conventionally used to represent a connection to a power net, such as `VCC` or `GND`. In addition to being a visual indicator that the attached net is a power rail, power ports make global connections: two power ports with the same pin name connect to each other anywhere in the schematic, regardless of sheet.

In the figure below, power port 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 ports are found in the `power` library, but power port symbols can be created in any library. To create a custom power port, make a new symbol with a power input pin that is set to be invisible. Name the pin according to the desired power net. In addition, set the "Define as power symbol" symbol property. As described in the hidden power pins section, invisible power input pins make global connections based on the hidden power pin's name. The process of creating a power port is described in more detail in the Symbol Editor section.

NOTE	The connected net name is determined by the power port's **pin name**, not the name or value of the symbol. This means that power port net names can only be changed in the symbol editor, not in the schematic.

Net name assignment rules

Every net in the schematic is assigned a name, whether that name is specified by the user 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. Labels globaux

Power ports

3. Local labels
4. Labels hiérarchiques
5. Hierarchical 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 in a hierarchy, and has no global label or power port, it will take its name from the highest level of the hierarchy where it has a hierarchical label or local label. As described above, local labels take priority over hierarchical labels.

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

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.

No-connection flag

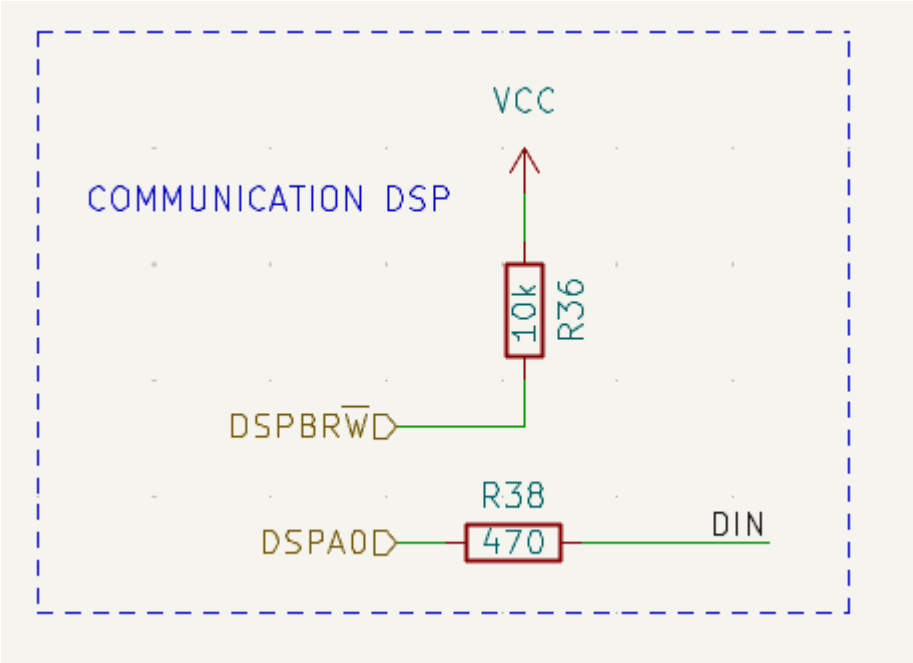
No-connection flags ( ) are used to indicate that a pin is intentionally unconnected. These flags do not have any effect on the schematic's connectivity, but they prevent "unconnected pin" ERC warnings for pins that are intentionally unconnected.

Graphical items

Text and graphic lines

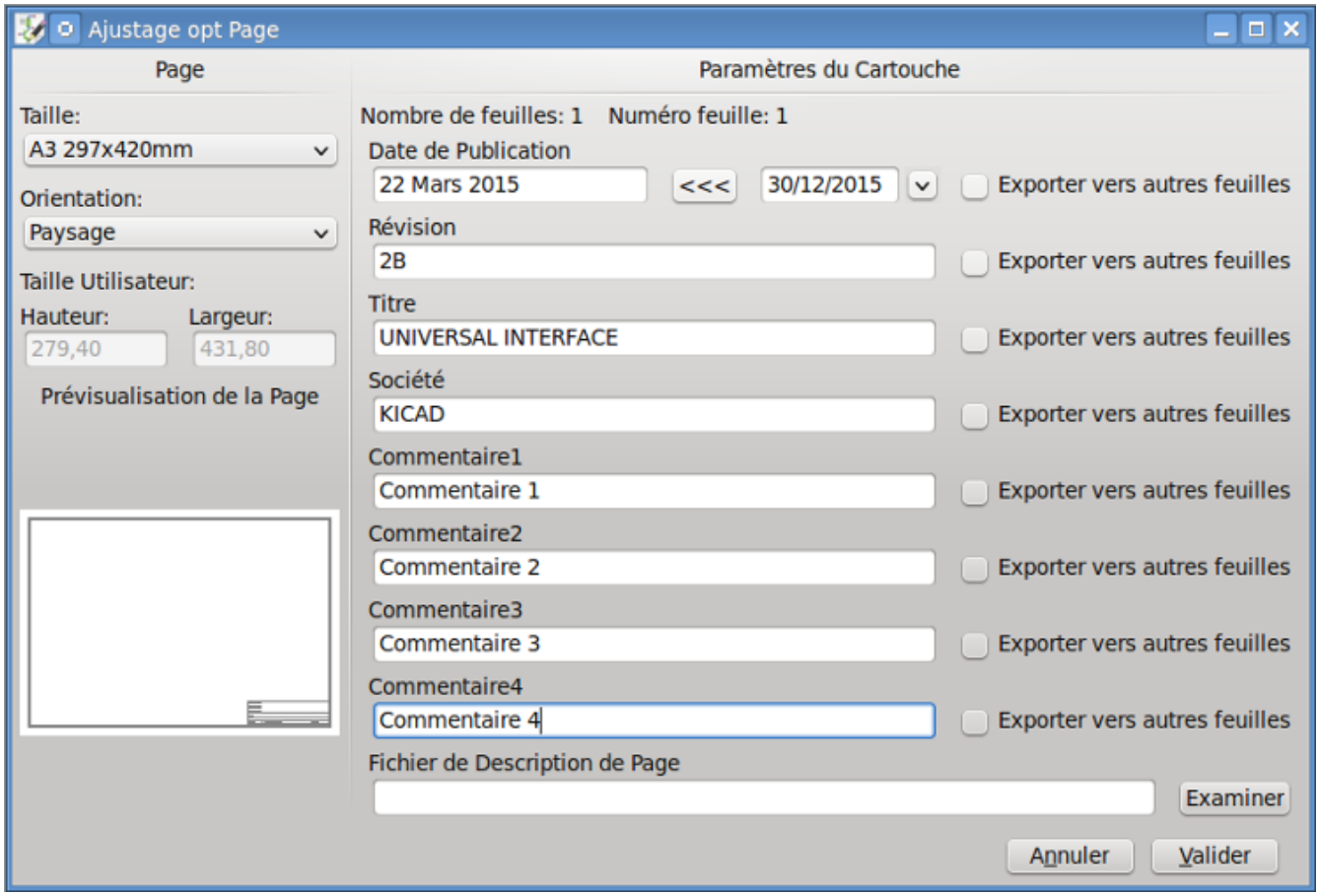
It can be useful to place annotations such as text fields and frames to aid in understanding the schematic. Text fields ( ) and graphic lines ( ) are intended for this use, as opposed to labels and wires, which are connection elements.

The image below shows graphic lines and text in addition to wires, local labels, and hierarchical labels.



Cartouche

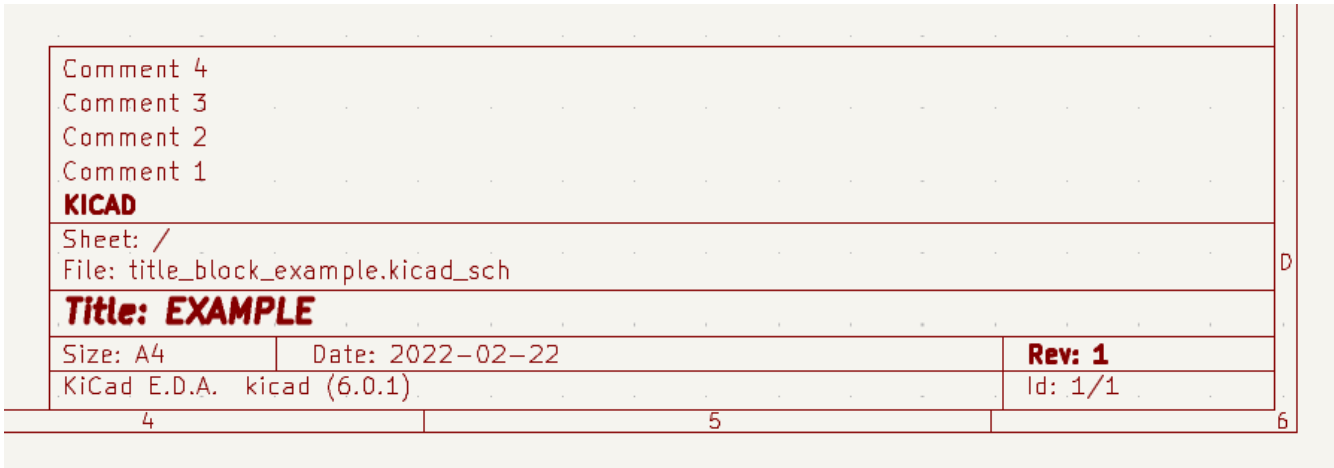
The title block is edited with the Page Settings tool ( ).



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 by "Issue Date", but the date in the schematic will not be automatically updated.

A drawing sheet template file can also be selected.



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

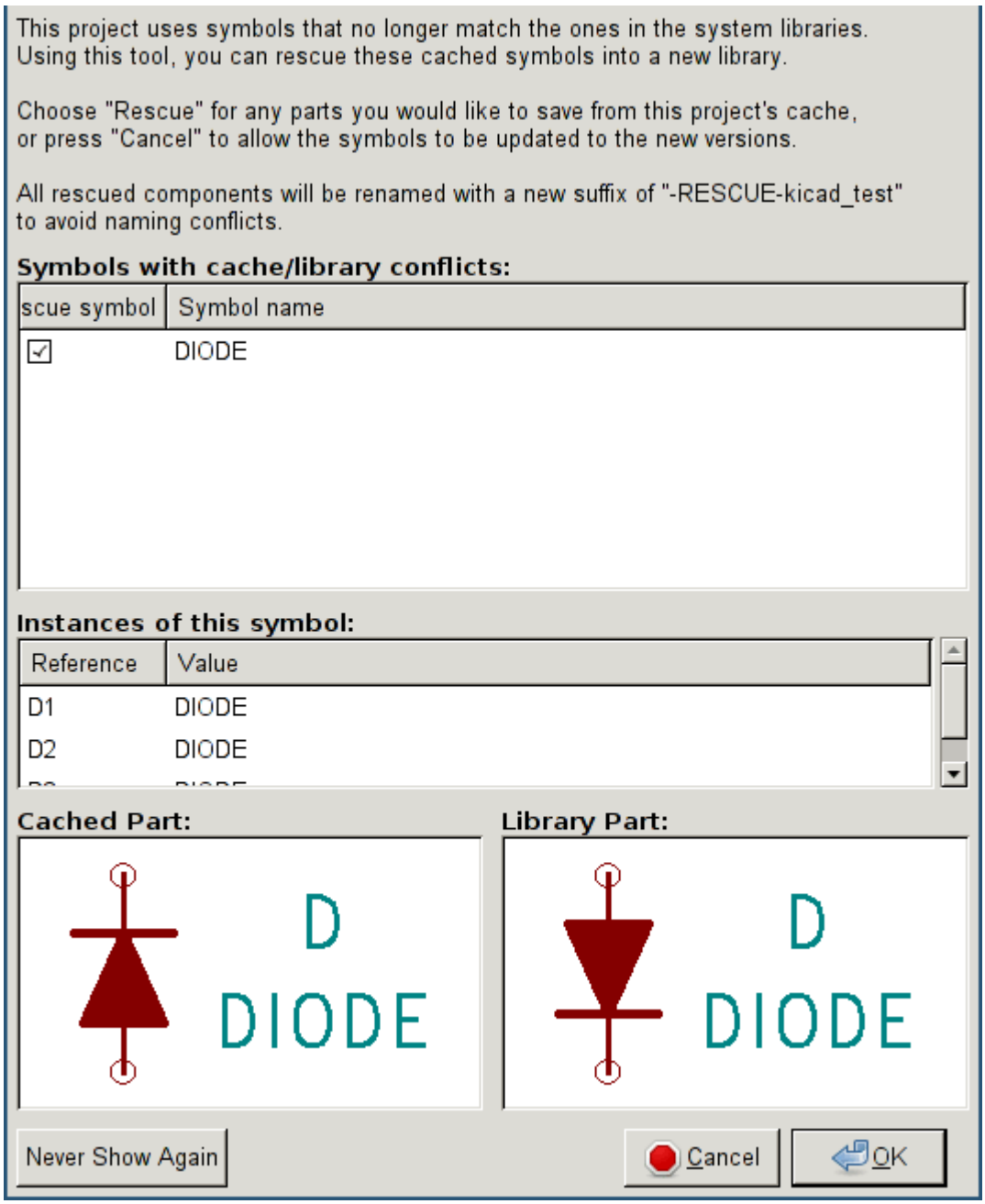
Schematic Setup

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, netclass setup, and schematic text variable setup.

Rescuing cached symbols

By default, KiCad loads symbols from the project libraries according to the set paths and library order. This can cause a problem when loading a very old project: if the symbols in the library have changed or have been removed or the library no longer exists since they were used in the project, the ones in the project would be automatically replaced with the new versions. The new versions might not line up correctly or might be oriented differently leading to a broken schematic.

When a project is saved, a cache library with the contents of the current library symbols is saved along with the schematic. This allows the project to be distributed without the full libraries. If you load a project where symbols are present both in its cache and in the system libraries, KiCad will scan the libraries for conflicts. Any conflicts found will be listed in the following dialog:



You can see in this example that the project originally used a diode with the cathode facing up, but the library now contains one with the cathode facing down. This change would break the schematic! Pressing OK here will cause the symbol cache library to be saved into a special `rescue` library and all the symbols are renamed to avoid naming conflicts.

If you press Cancel, no rescues will be made, so KiCad will load all the new components by default. If you save the schematic at this point, your cache will be overwritten and the old symbols will not be recoverable. If you have saved the schematic, you can still go back and run the rescue function again by selecting "Rescue Cached Components" in the "Tools" menu to call up the rescue dialog again.

If you would prefer not to see this dialog, you can press "Never Show Again". The default will be to do nothing and allow the new components to be loaded. This option can be changed back in the Libraries preferences.

Schématiques hiérarchiques

Introduction

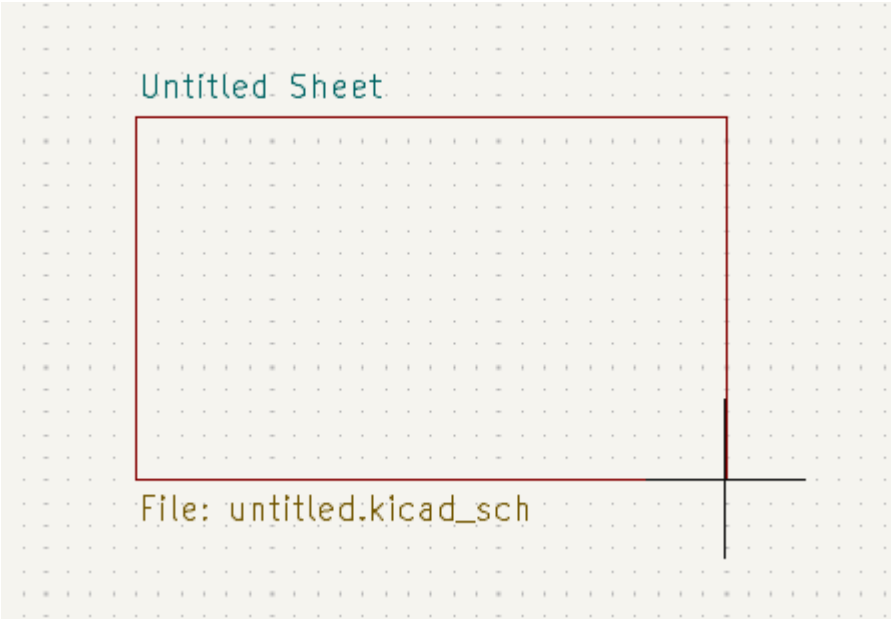
In KiCad, multi-sheet schematics are hierarchical: there is a single root sheet, and additional sheets are created as subsheets of either the root sheet or another subsheet. Sheets can be included in a hierarchy multiple times, if desired.

Carefully drawing a schematic as a hierarchical design improves schematic legibility and reduces repetitive drawing.

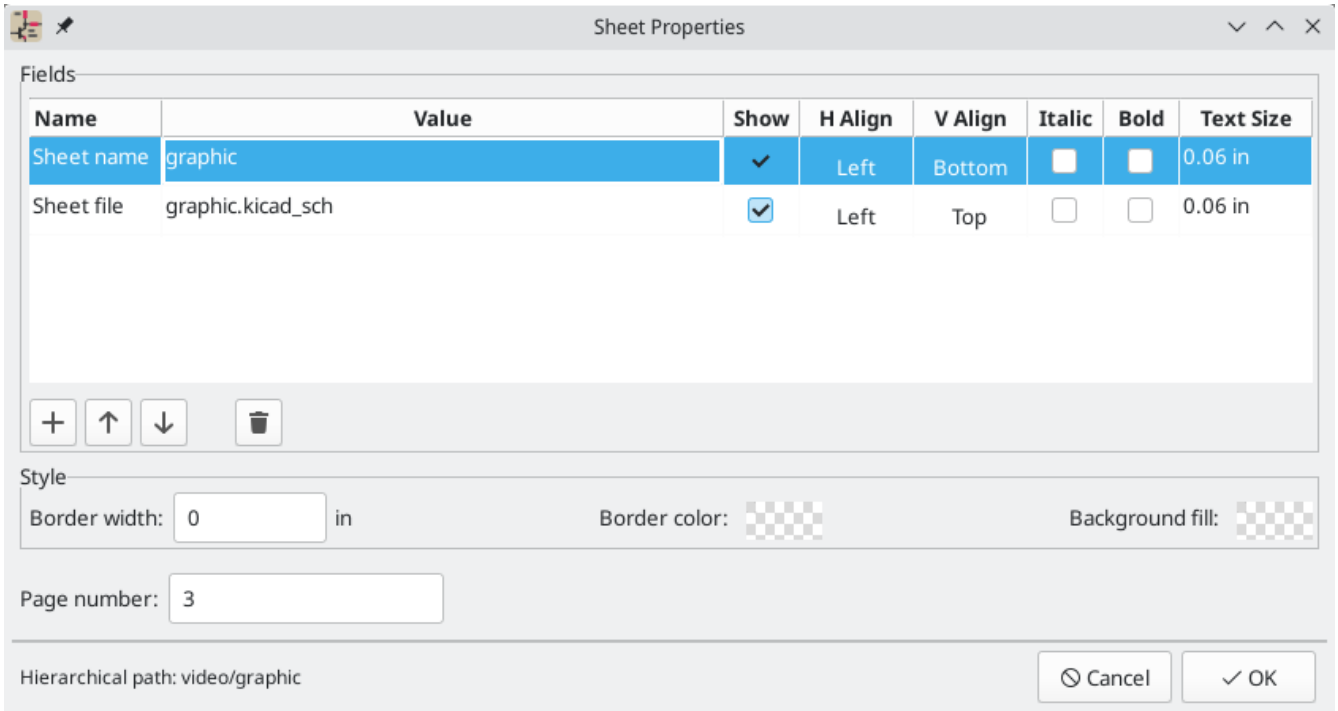
Creating a hierarchical schematic starts from the root sheet. The process is to create a subsheet, then draw the circuit in the subsheet and make the necessary electrical connections between sheets. Connections can be made between nets in a subsheet and nets in the parent sheet using hierarchical pins and labels, or between any two nets in the hierarchy using global labels.

Adding sheets to a design

You can add a 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. 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.



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 filename** 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.

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.

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.

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.

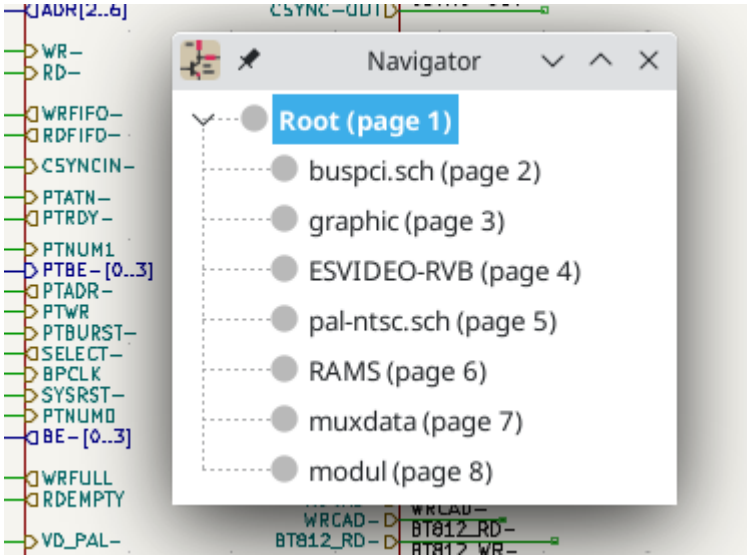
The Sheet Properties dialog can be accessed at any time by selecting a sheet symbol and using the E hotkey, or by right-clicking on a sheet symbol and selecting **Properties...**.

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**.

Alternatively, you can jump to any sheet with the hierarchy navigator. To open the hierarchy navigator, click the button in the top toolbar. Each sheet in the design is displayed as an item in the tree. Clicking a sheet name opens that sheet in the editing canvas.



By default, the hierarchy navigator closes after a new sheet is opened. It can be configured to always remain open by selecting the **Keep hierarchy navigator open** option in the Editing Options section of the Schematic Editor preferences.

Electrical connections between sheets

Label overview

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 button.
- **Global labels** make connections anywhere in a schematic, regardless of sheet. Global labels are added with the button.
- **Hierarchical labels** 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 hierarchical pins as defining the interface for a sheet. Hierarchical labels are added with the button.

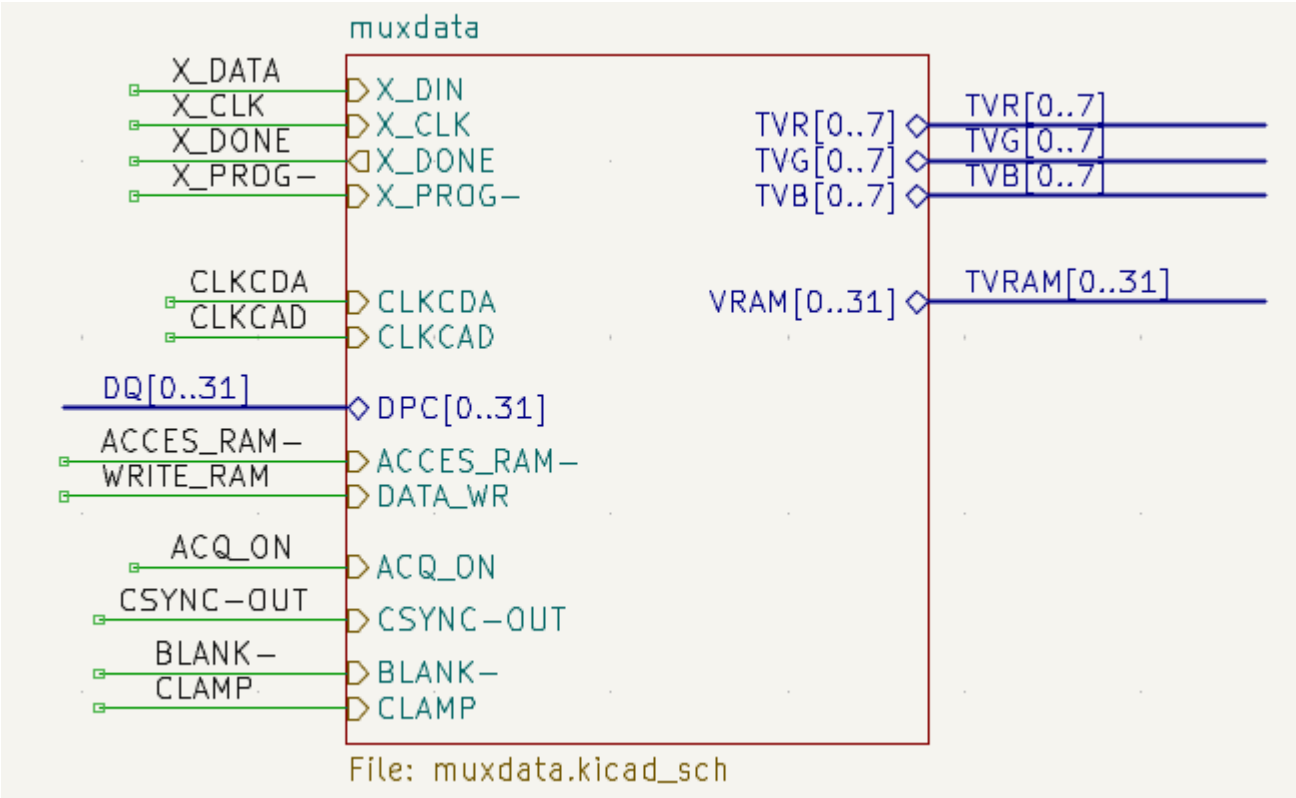
NOTE	Labels that have the same name will connect, regardless of the label type, if they are in the same sheet.

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

Hierarchical sheet pins

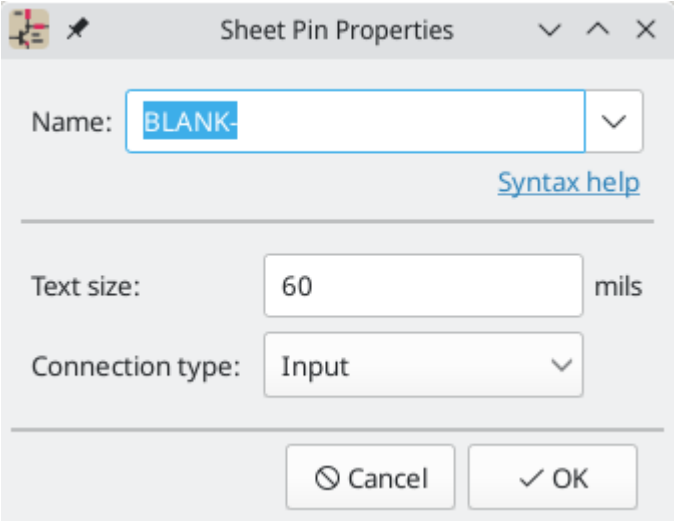
After placing hierarchical labels within the subsheet, matching **hierarchical pins** can be added to the subsheet symbol in the parent sheet. You can then make connections to the hierarchical pins with wires, labels, and buses. Hierarchical pins in a subsheet symbol are connected to the matching hierarchical 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, import the corresponding hierarchical pin into 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 import additional sheet pins until there are no more hierarchical pins to import from the subsheet. Sheet pins can also be imported by selecting **Import 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...**.



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 a pin name is changed the label must change as well.

The **connection type** 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 **text size** can also be changed.

Hierarchical design examples

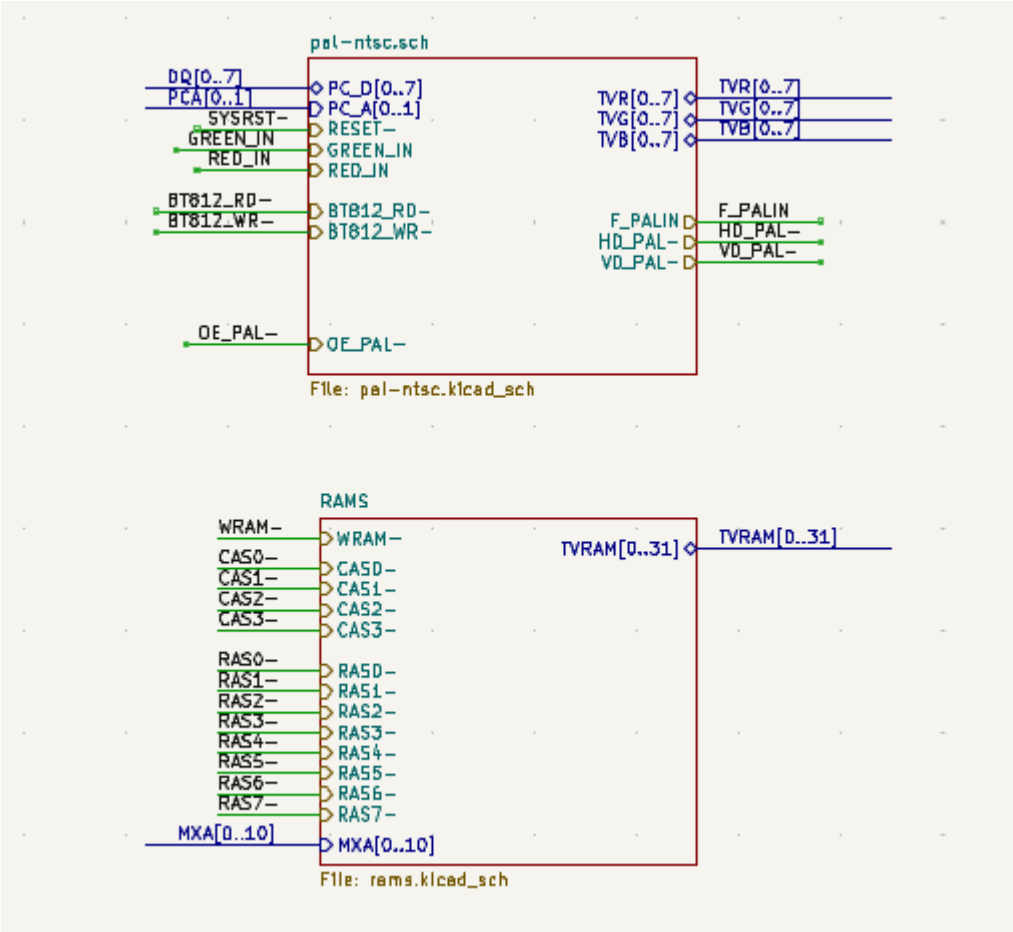
Hierarchical designs can be put into one of several categories:

- **Simple:** each sheet is used only once.
- **Complex:** some sheets are instantiated multiple times.
- **Flat:** a sub-case of a **simple** hierarchy, without connections between subsheets and their parent. Flat hierarchies can be used to represent a non-hierarchical design.

Each hierarchy model can be useful; the most appropriate one depends on the design.

Simple hierarchy

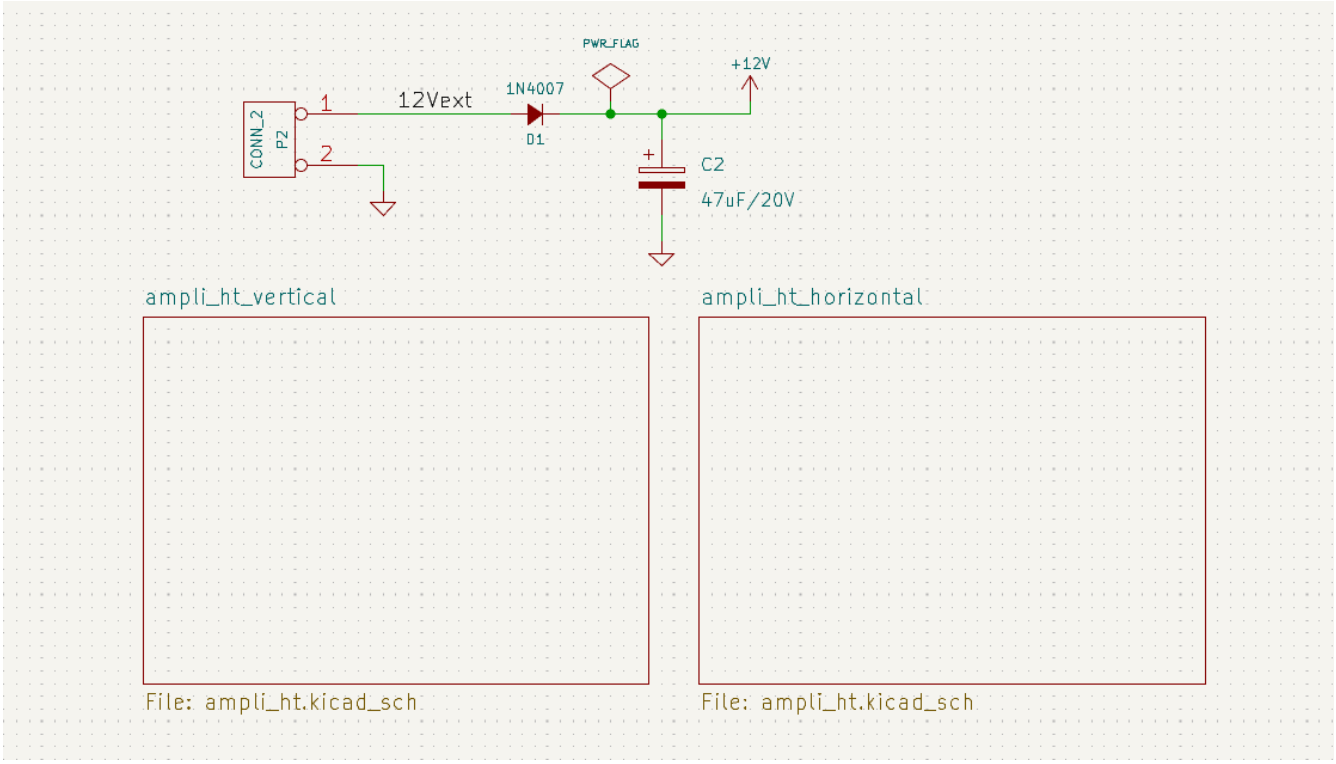
An example of a simple hierarchy is the `video` demo project included with KiCad. The root sheet contains seven unique subsheets, each with hierarchical labels and sheet pins linking the sheets to each other in the root sheet. Two of the subsheet symbols are shown below.



Hiérarchie complexe

The `complex_hierarchy` demo project is an example of a complex hierarchy. The root sheet contains two subsheet symbols, which both refer to the same sheet file (`ampli_ht.kicad_sch`). This allows the design to include two copies of the same amplifier circuit. Although the two sheet symbols refer to the same filename, the sheet names are unique (`ampli_ht_vertical` and `ampli_ht_horizontal`). Inside each subsheet the circuits are identical except for the reference designators, which as always are unique.

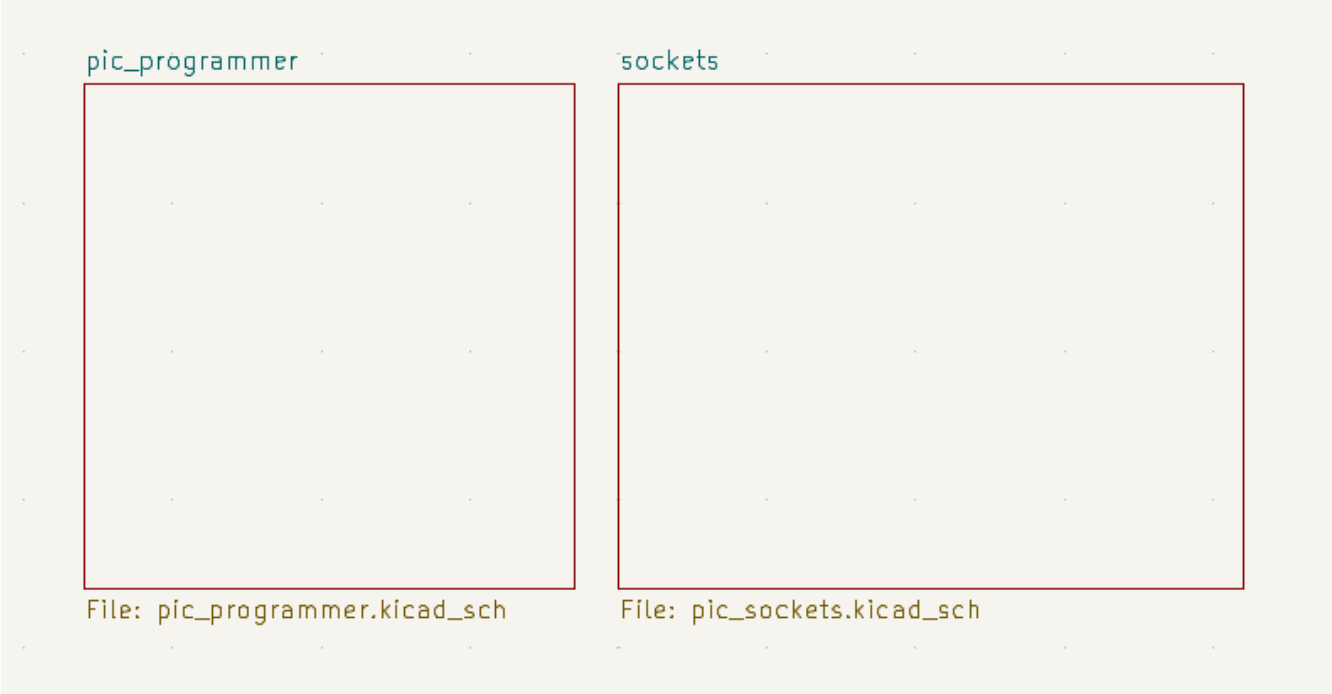
This project contains no sheet pin connections. The only connections between the root sheet and the subsheets are global power connections made with power port symbols. However, sheets in a complex hierarchy could include sheet pin connections if appropriate for the design.



Hiérarchie à plat

The `flat_hierarchy` demo project is an example of a flat hierarchy. The root sheet contains two unique subsheet symbols with no hierarchical sheet pins. The root sheet in this project does nothing except hold the subsheets, and the subsheets are used only as additional pages in the schematic.

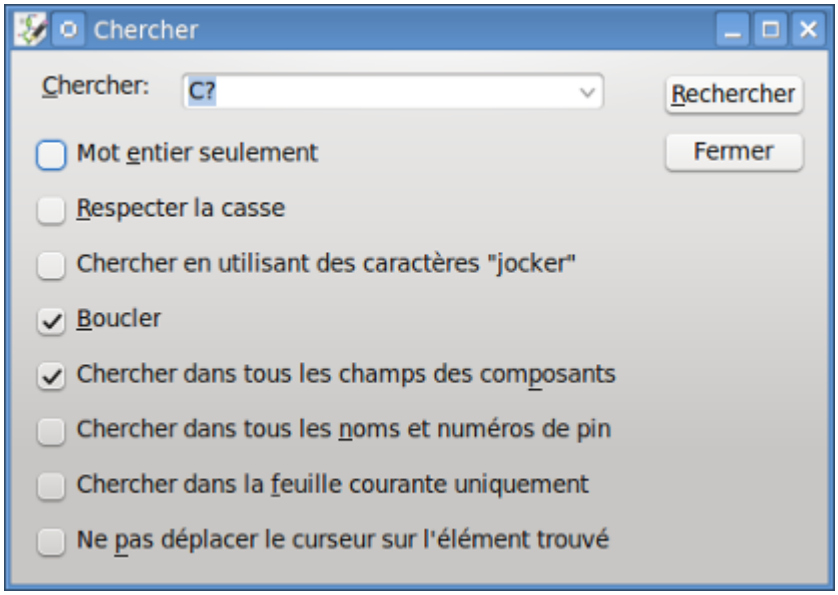
NOTE	This is the simplest way to create multi-page schematics in KiCad.



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.



The Find tool has several options:

Match case: Selects whether the search is case-sensitive.

Words: 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.

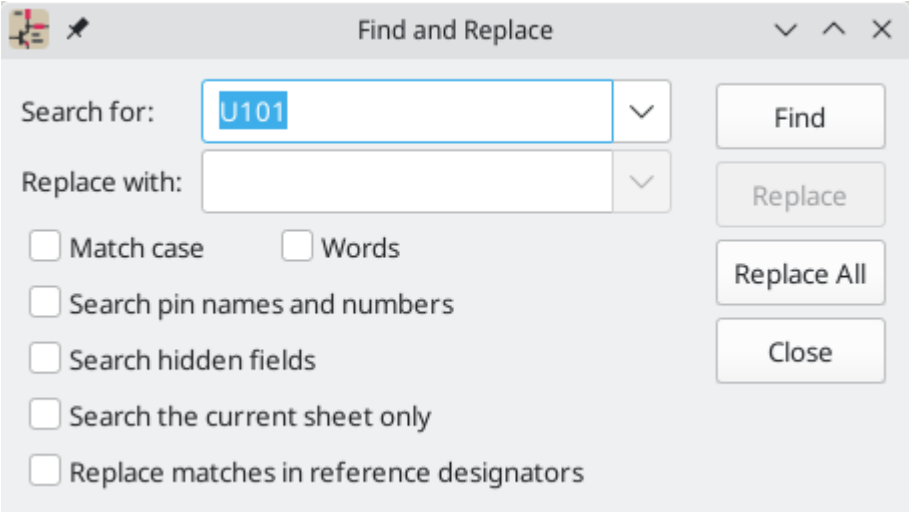
Wildcards: When selected, wildcards can be used in the search terms. `?` matches any single character, and `*` matches any number of characters. Note that when this option is selected, partial matches are not returned: searching for `abc*` will match the string `abcd`, but searching for `abc` will not.

Search pin names and numbers: Selects whether the search should apply to pin names and numbers.

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 or to the entire schematic.

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.



If the **Replace matches in reference designators** option is checked, reference designators will be modified if they contain matching text. Otherwise reference designators will not be affected.

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. If there are no nets or pins under the cursor, any existing highlighting will be cleared. The highlighting can also be cleared by using the Clear Net Highlight action (hotkey ~).

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. This behavior can be disabled in the Display Options section of the Preferences dialog.

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.

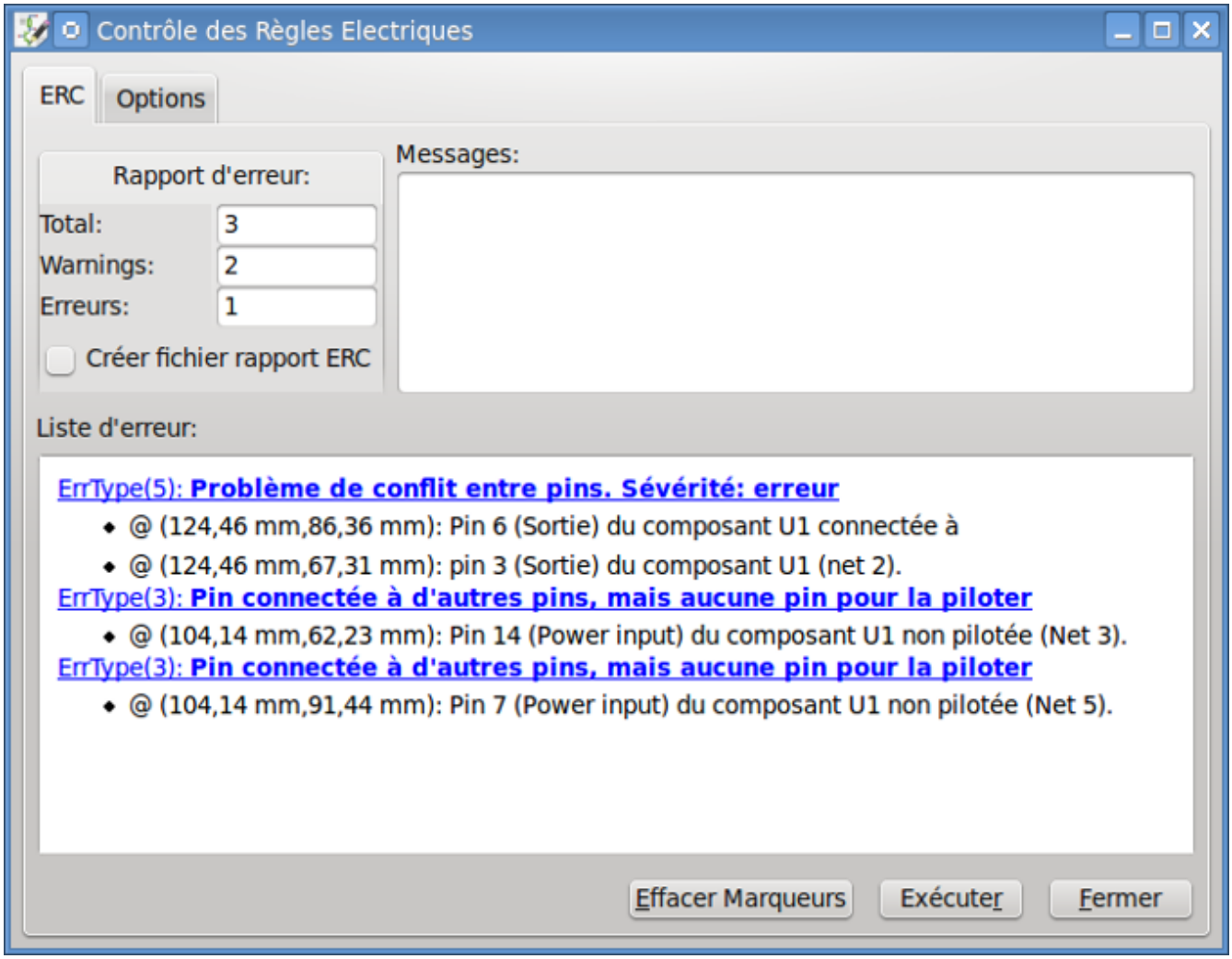
Vérification des règles électriques (ERC)

The Electrical Rules Check (ERC) tool performs an automatic check of your schematic. The ERC checks for any errors in your sheet, such as unconnected pins, unconnected hierarchical symbols, shorted outputs, etc. ERC output is reported as errors or warnings depending on the severity of the issue detected.

Naturally, an automatic check is not infallible, and it is not possible to detect all design errors. Such a check is still very useful, because it allows you to detect many oversights and small errors. 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 library creation.

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

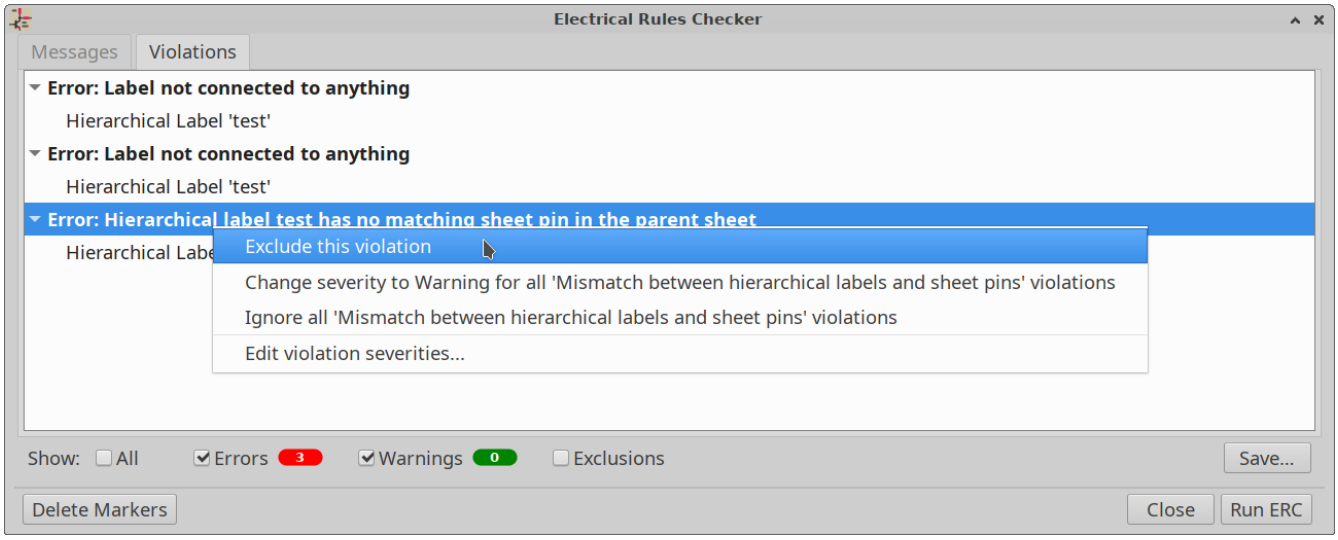


Any warnings or errors are reported in the **Violations** tab, and markers for each violation are placed in the schematic so that they point to the relevant part of the schematic. Warnings are indicated by yellow arrows, and errors have red arrows. Excluded violations are shown as green arrows.

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

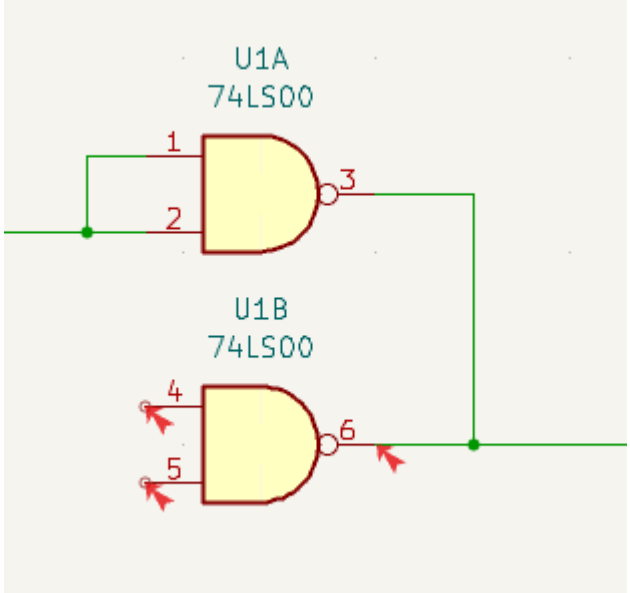
The numbers at the bottom of the window show the number of errors, warnings, and exclusions. Each type of violation can be filtered from the list using the respective checkboxes. Clicking **Delete Markers** will clear all violations until ERC is run again.

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



- **Exclude this violation:** ignores this particular violation, but does not affect any other violations.
- **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.

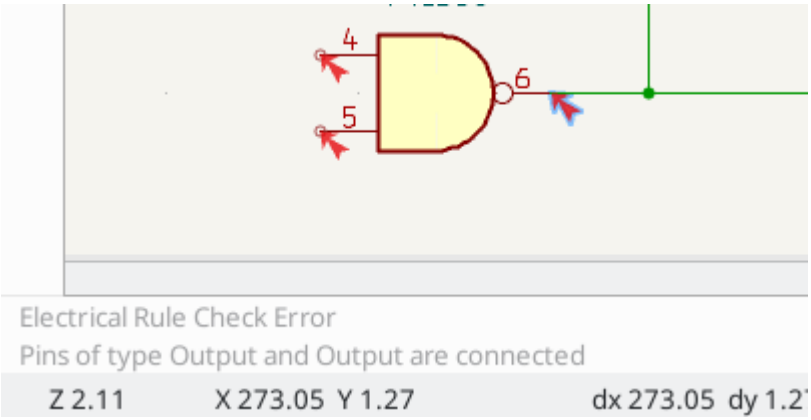
ERC example



Here you can see three errors:

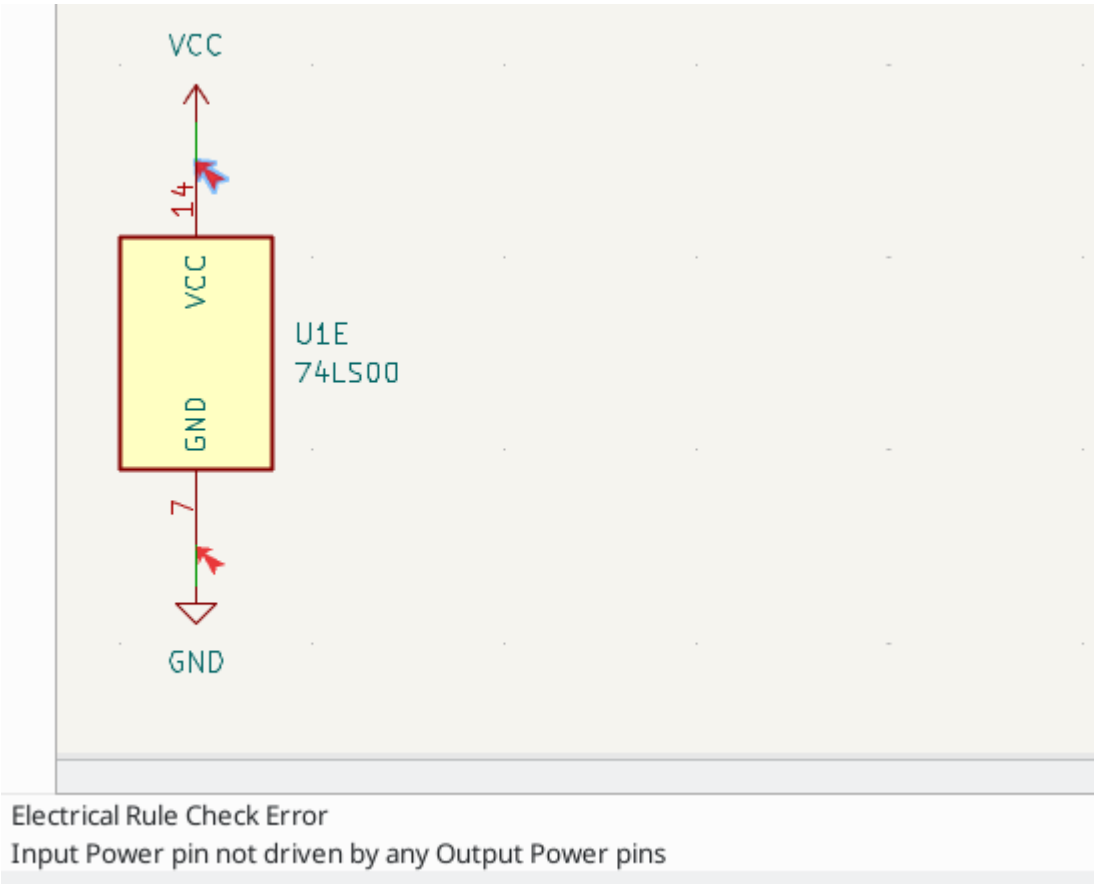
- Two outputs have been connected together (red arrow at right).
- Two inputs have been left unconnected (red arrows at left).

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

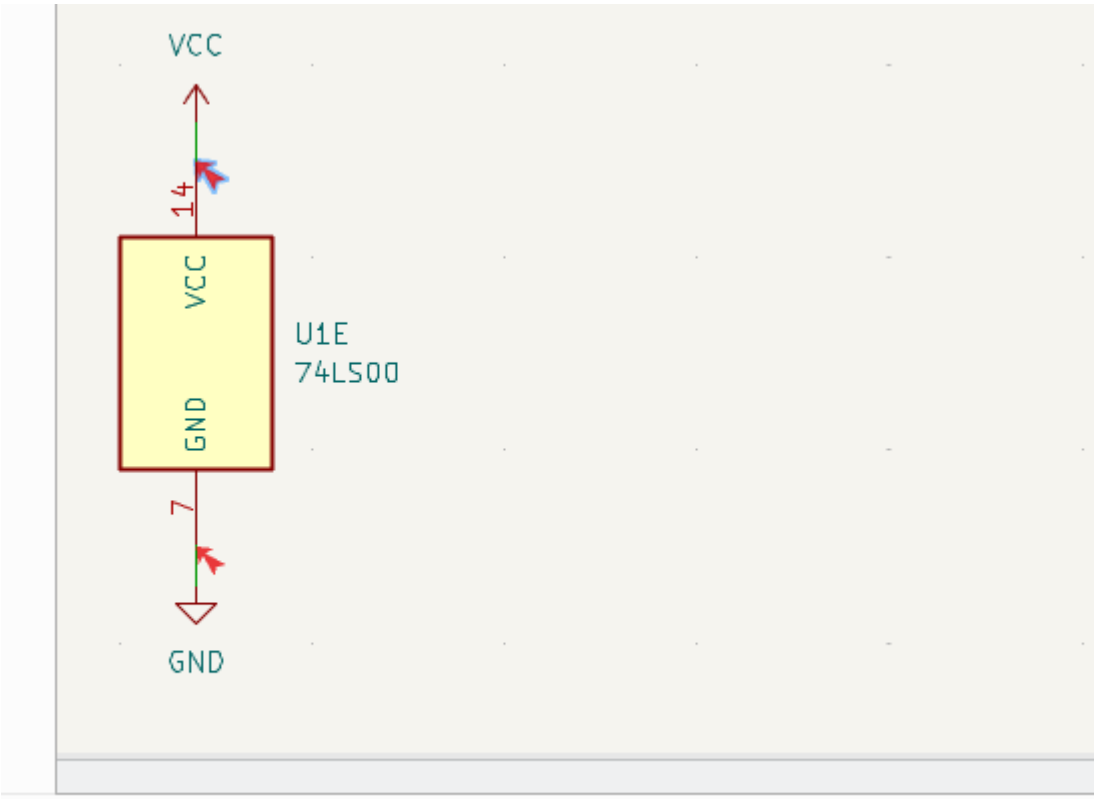


Power pins and power flags

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.



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 power output pin to the net; `PWR_FLAG` is simply a symbol with a single power output pin.



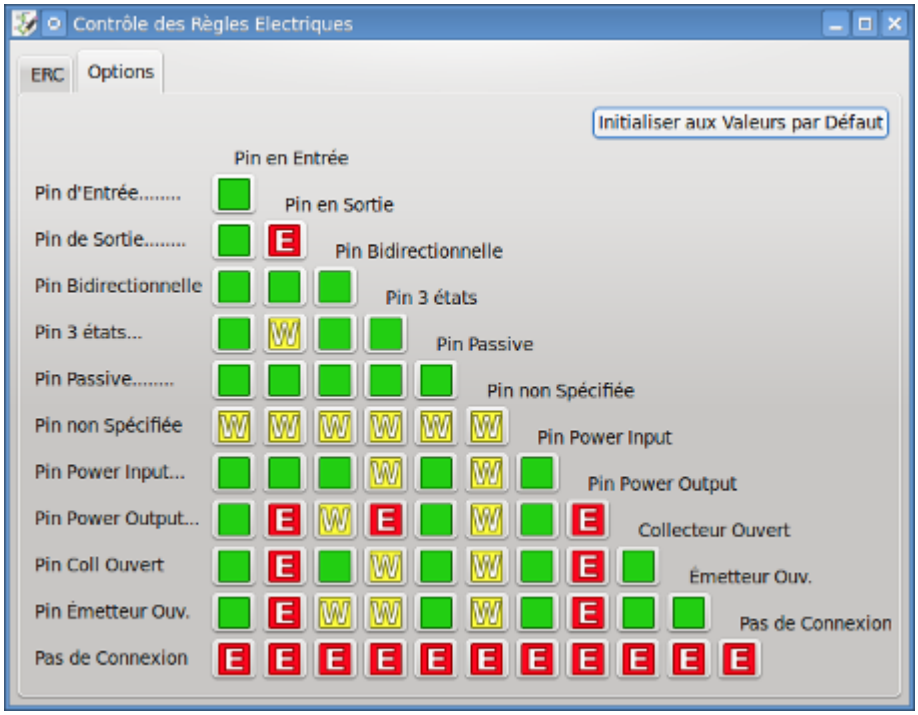
Electrical Rule Check Error
Input Power pin not driven by any Output Power pins

Ground nets often need a `PWR_FLAG` as well, because voltage regulators have outputs declared as power outputs, but their ground pins are typically marked as power inputs. Therefore grounds can appear unconnected to a source unless a `PWR_FLAG` symbol is used.

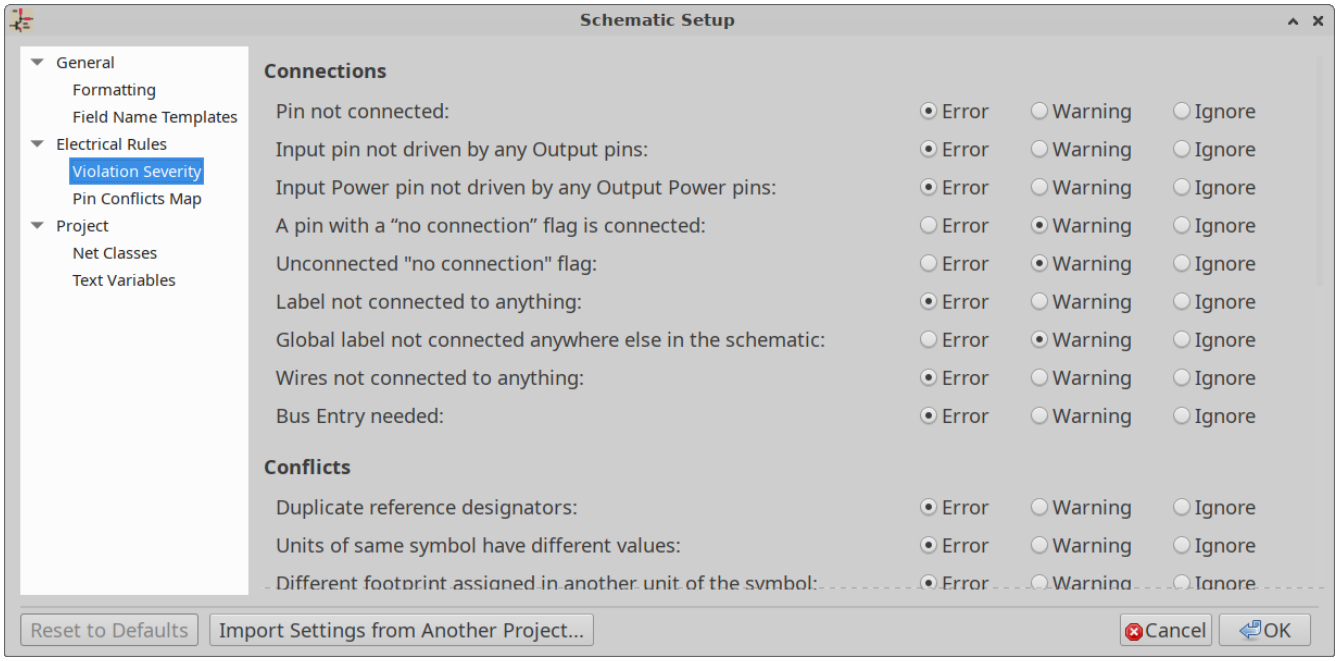
For more information about power pins and power flags, see the `PWR_FLAG` documentation.

ERC Configuration

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.



Les règles sont modifiées en cliquant plusieurs fois sur le bouton carré dans le tableau pour faire défiler les différents choix : normal [vert], avertissement [W jaune], erreur [E rouge].



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

Fichier de rapport d'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`. Here is an example ERC report file.

```
ERC report (Fri 21 Oct 2022 02:07:05 PM EDT, Encoding UTF8)

***** Sheet /
[pin_not_driven]: Input pin not driven by any Output pins
    ; Severity: error
    @(149.86 mm, 60.96 mm): Symbol U1B [74LS00] Pin 4 [, Input, Line]
[pin_not_connected]: Pin not connected
    ; Severity: error
    @(149.86 mm, 60.96 mm): Symbol U1B [74LS00] Pin 4 [, Input, Line]
[pin_not_connected]: Pin not connected
    ; Severity: error
    @(149.86 mm, 66.04 mm): Symbol U1B [74LS00] Pin 5 [, Input, Line]
[pin_to_pin]: Pins of type Output and Output are connected
    ; Severity: error
    @(165.10 mm, 63.50 mm): Symbol U1B [74LS00] Pin 6 [, Output, Inverted]
    @(165.10 mm, 46.99 mm): Symbol U1A [74LS00] Pin 3 [, Output, Inverted]
[pin_not_driven]: Input pin not driven by any Output pins
    ; Severity: error
    @(149.86 mm, 66.04 mm): Symbol U1B [74LS00] Pin 5 [, Input, Line]

 ** ERC messages: 5  Errors 5  Warnings 0
```

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:

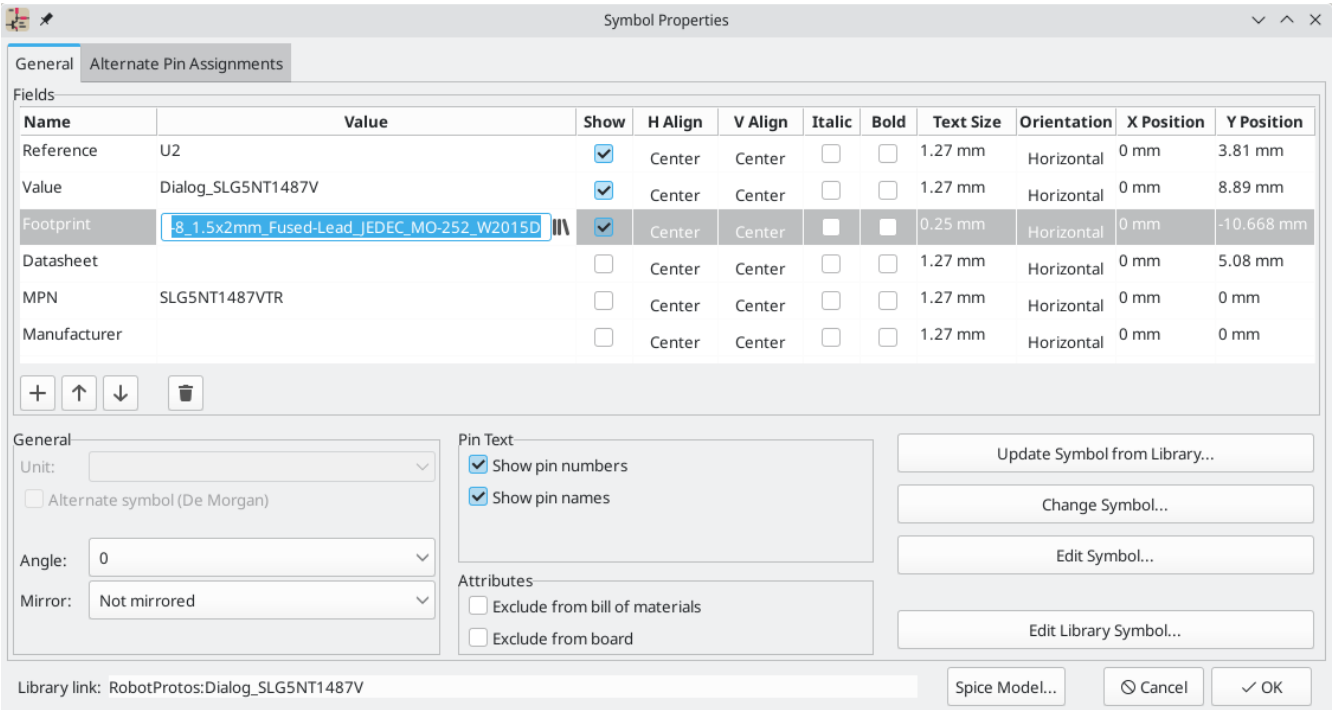
- Symbol Properties
  - Symbol Properties Dialog
  - Symbol Fields Table
- While placing symbols
- Footprint Assignment Tool

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.

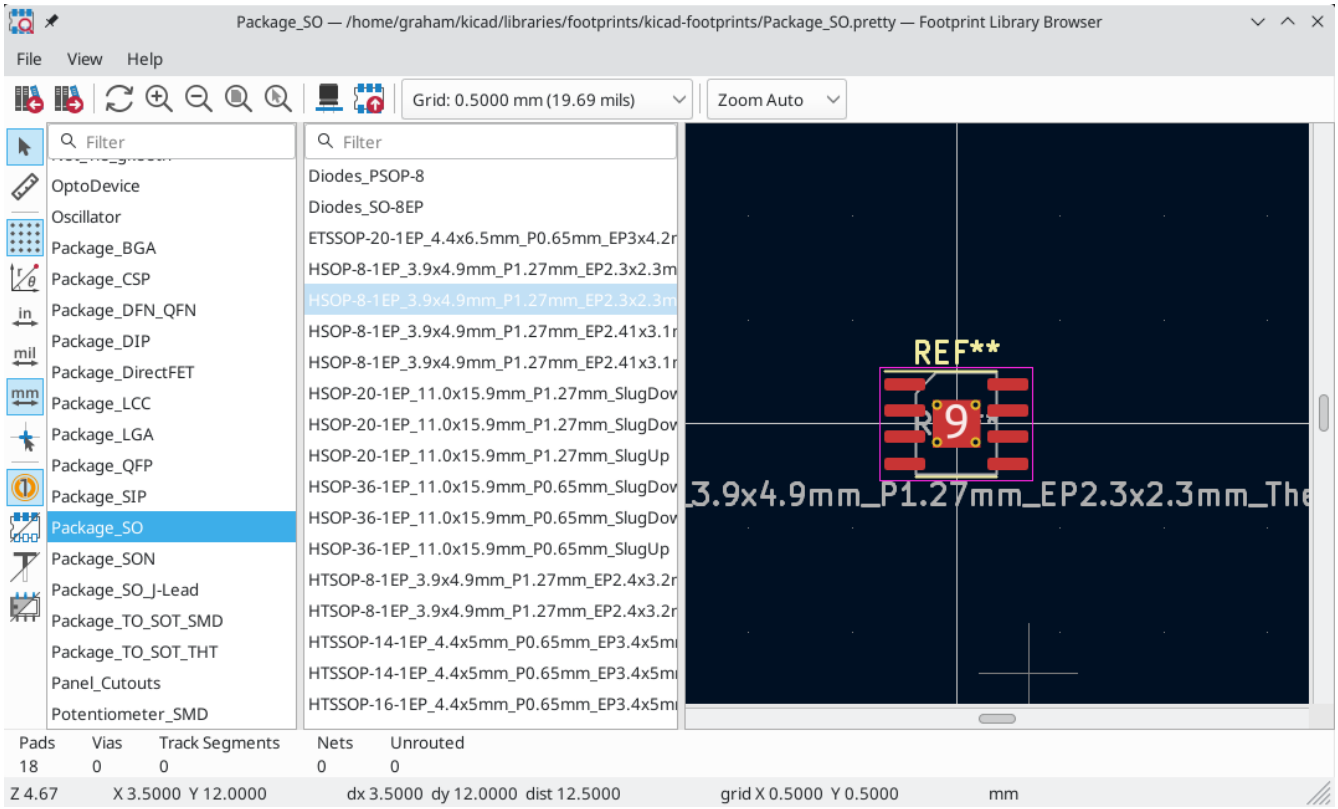
NOTE	The Footprint Library Table needs to be configured before footprints can be assigned. For information on configuring the Footprint Library Table, please see the PCB Editor manual.

Assigning Footprints in Symbol Properties

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



Clicking the button in the `Footprint` field opens the Footprint Library Browser, which shows the available footprints and footprint libraries. Single clicking a footprint name selects the footprint and displays it in the preview pane on the right, while double clicking on a footprint closes the browser and sets the symbol's `Footprint` field to the selected footprint.

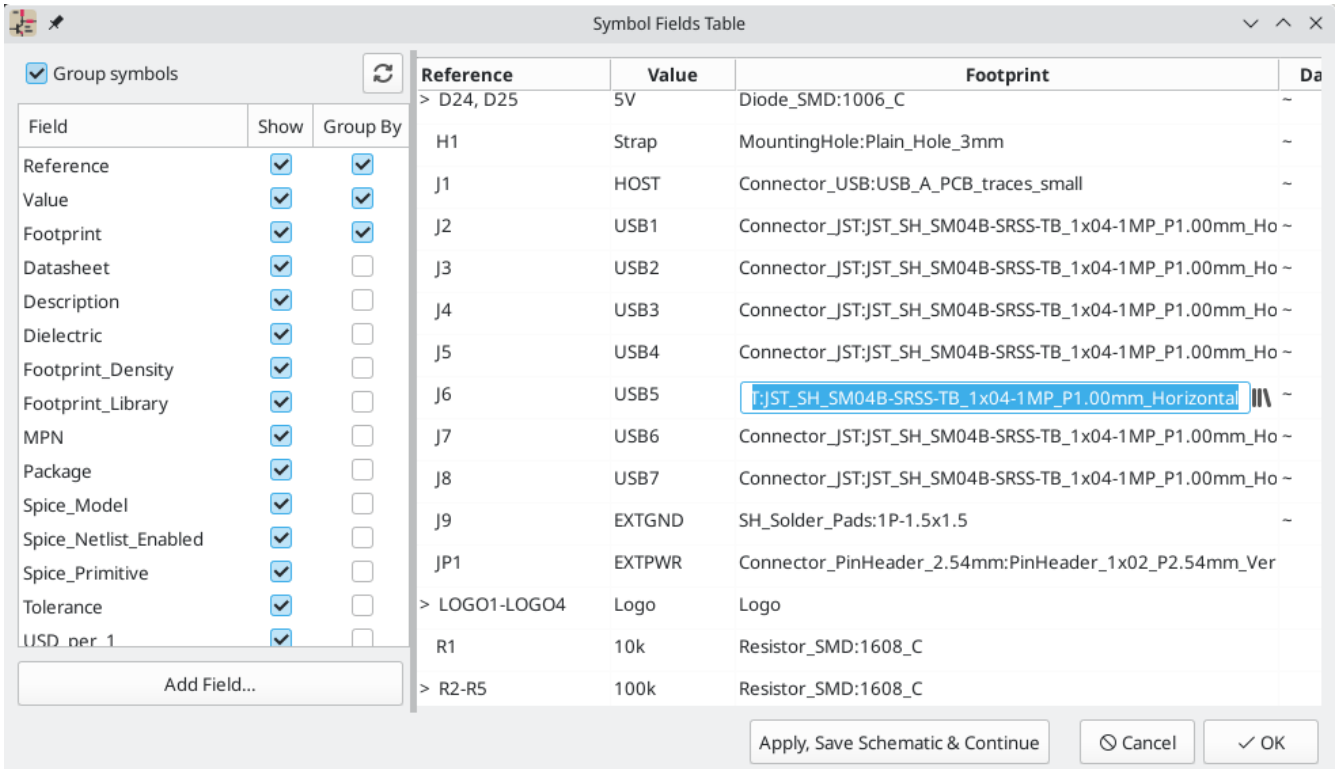


Assigning Footprints with the Symbol Fields Table

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.

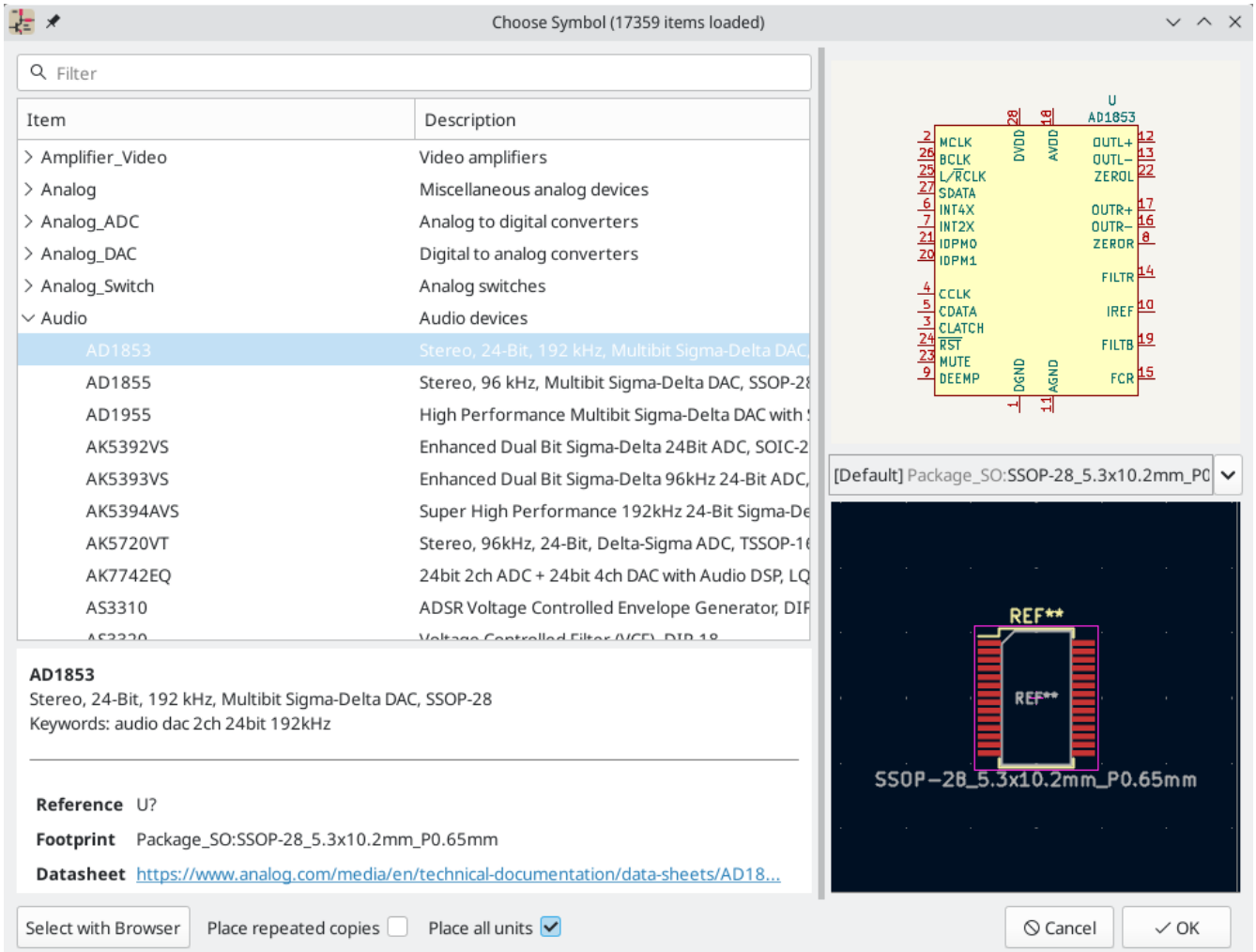


For more information on the Symbol Fields Table, see the section on editing symbol properties.

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. The default footprint is shown in the Add Symbol dialog. 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".



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

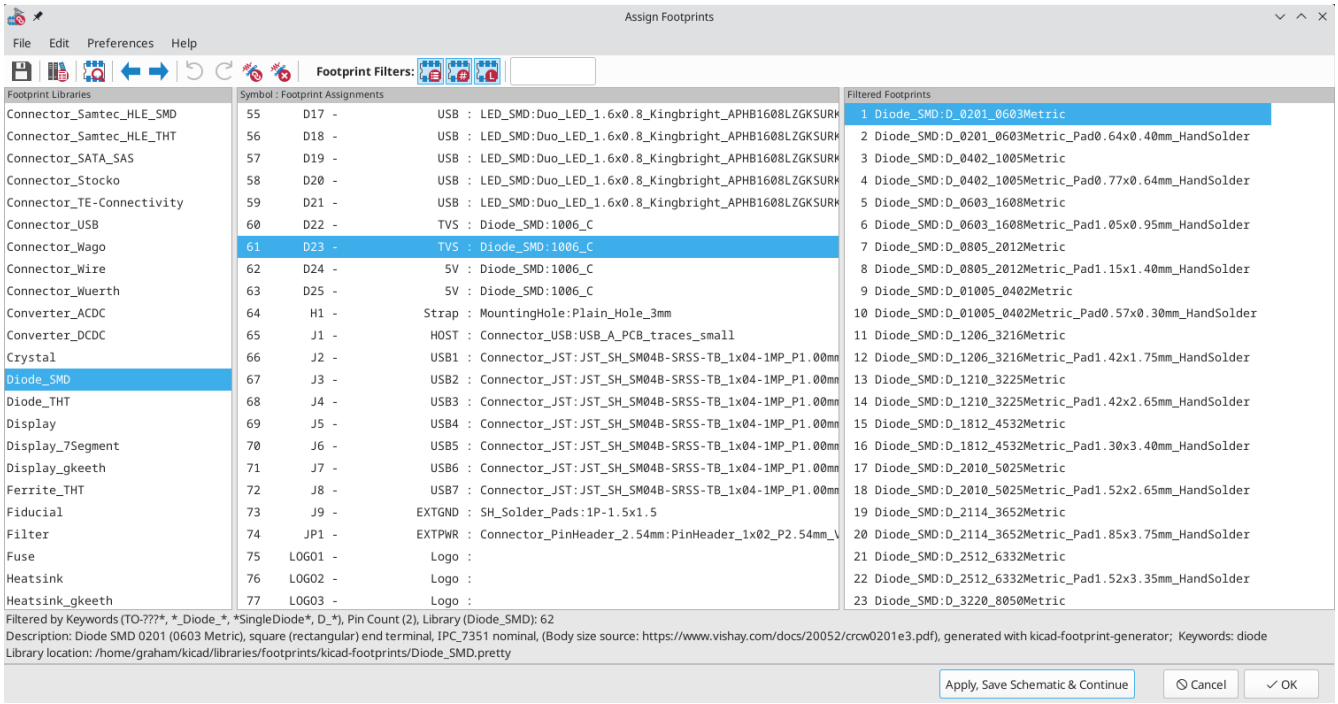
The Footprint Assignment Tool allows you to associate symbols in your schematic to footprints used when laying out the printed circuit board. It provides footprint list filtering, footprint viewing, and 3D component model viewing to help ensure the correct footprint is associated with each component.

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.



- 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.
	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.

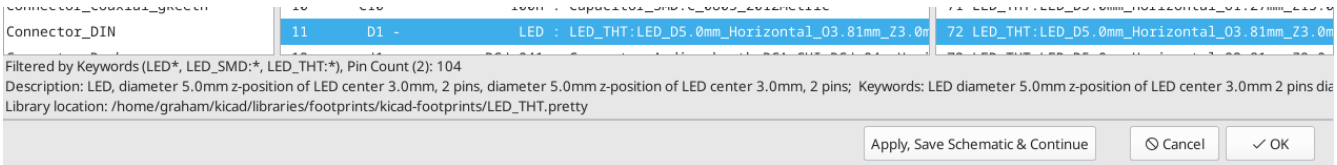
Filtrage de la Liste d'Empreintes

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:



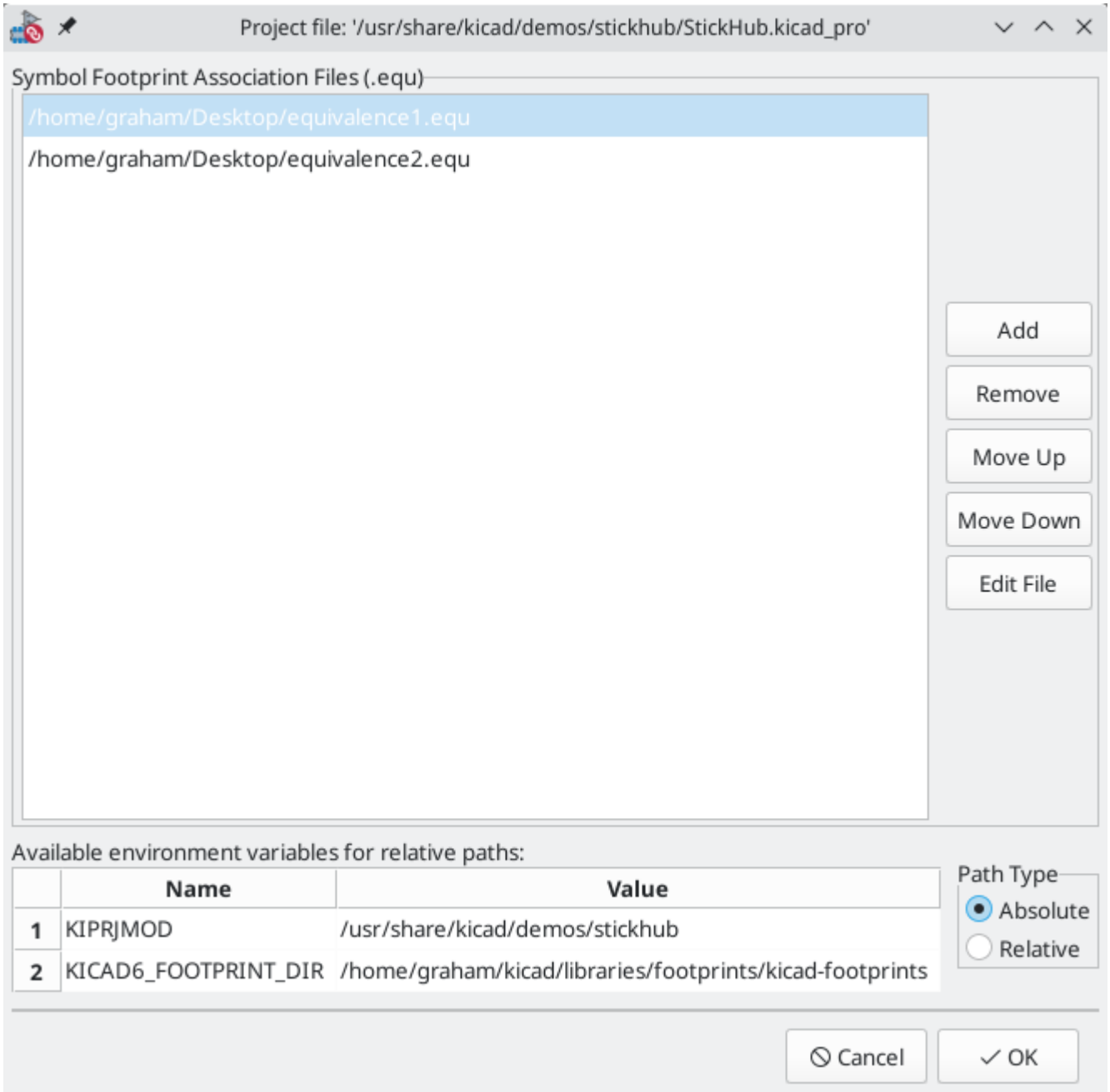
Multiple filters can be used at once to help narrow down the list of possibly appropriate footprints in the footprint pane. The symbols in KiCad's standard library define footprint filters that are designed to be used in combination with the pin count filter.

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.



- 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.

Format des Fichiers d'Équivalences

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'
```

Voici un exemple de fichier d'équivalences :

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

==== Visualiser l'Empreinte Courante

The Footprint Assignment Tool contains a footprint viewer. Clicking the
image:images/icons/icon_footprint_browser_24.png[footprint viewer icon] button in the top
toolbar launches the footprint viewer and shows the selected footprint.

image::images/en/footprint_view.png[scaledwidth="90%", alt="Viewing a footprint"]

The top toolbar contains the following commands:

[width="90%", cols="10%,90%"]
|=========================================================================
|image:images/icons/refresh_24.png[]
|Refresh view
|image:images/icons/zoom_in_24.png[]
|Zoom in

|image:images/icons/zoom_out_24.png[]
|Zoom out

|image:images/icons/zoom_fit_in_page_24.png[]
|Zoom to fit drawing in display area

|image:images/icons/shape_3d_24.png[]
|Show 3D viewer
|=========================================================================

The left toolbar contains the following commands:

[width="90%", cols="10%,90%"]
|=========================================================================
|image:images/icons/cursor_24.png[]
|Use the select tool

|image:images/icons/measurement_24.png[]
|Interactively measure between two points

|image:images/icons/grid_24.png[]
|Display grid dots or lines

|image:images/icons/polar_coord_24.png[]
|Switch between polar and cartesian coordinate systems

|image:images/icons/unit_inch_24.png[]
|Use inches

|image:images/icons/unit_mil_24.png[]
|Display coordinates in mils (1/1000 of an inch)

|image:images/icons/unit_mm_24.png[]
|Display coordinates in millimeters

|image:images/icons/cursor_shape_24.png[]
|Toggle display of full-window crosshairs

|image:images/icons/pad_number_24.png[]
|Toggle between drawing pads in sketch or normal mode
```

(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 (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 "00000000-0000-0000-0000-000057890691")) (comp (ref "U1") (value "AD8051") (fields (field (name "Spice_Lib_File") "ad8051.lib") (field (name "Spice_Model") "AD8051") (field (name "Spice_Netlist_Enabled") "Y") (field (name "Spice_Primitive") "X")) (libsource (lib "sallen_key_schlib") (part "Generic_Opamp") (description "")) (property (name "Spice_Primitive") (value "X")) (property (name "Spice_Model") (value "AD8051")) (property (name "Spice_Lib_File") (value "ad8051.lib")) (property (name "Spice_Netlist_Enabled") (value "Y")) (property (name "Sheetname") (value "")) (property (name "Sheetfile") (value "sallen_key.kicad_sch")) (sheetpath (names "/") (tstamps "/")) (tstamps "00000000-0000-0000-0000-00005788ff9f")) (comp (ref "V1") (value "AC 1") (libsource (lib "sallen_key_schlib") (part "VSOURCE") (description "")) (property (name "Sheetname") (value "")) (property (name "Sheetfile") (value "sallen_key.kicad_sch")) (sheetpath (names "/") (tstamps "/")) (tstamps "00000000-0000-0000-0000-000057336052")) (comp (ref "V2") (value "DC 10") (fields (field (name "Fieldname") "Value") (field (name "Spice_Node_Sequence") "1 2") (field (name "Spice_Primitive") "V")) (libsource (lib "sallen_key_schlib") (part "VSOURCE") (description "")) (property (name "Fieldname") (value "Value")) (property (name "Spice_Primitive") (value "V")) (property (name "Spice_Node_Sequence") (value "1 2")) (property (name "Sheetname") (value "")) (property (name "Sheetfile") (value "sallen_key.kicad_sch")) (sheetpath (names "/") (tstamps "/")) (tstamps "00000000-0000-0000-0000-0000578900ba")) (comp (ref "V3") (value "DC 10") (fields (field (name "Fieldname") "Value") (field (name "Spice_Node_Sequence") "1 2") (field (name "Spice_Primitive") "V")) (libsource (lib "sallen_key_schlib") (part "VSOURCE") (description "")) (property (name "Fieldname") (value "Value")) (property (name "Spice_Primitive") (value "V")) (property (name "Spice_Node_Sequence") (value "1 2")) (property (name "Sheetname") (value "")) (property (name "Sheetfile") (value "sallen_key.kicad_sch")) (sheetpath (names "/") (tstamps "/")) (tstamps "00000000-0000-0000-0000-000057890232"))) (libparts (libpart (lib "sallen_key_schlib") (part "C") (footprints (fp "C?") (fp "C_????*") (fp "C????") (fp "SMD*c") (fp "Capacitor*")) (fields (field (name "Reference") "C") (field (name "Value") "C")) (pins (pin (num "1") (name "") (type "passive")) (pin (num "2") (name "") (type "passive")))) (libpart (lib

*"sallen_key_schlib") (part "Generic_Opamp") (fields (field (name "Reference") "U") (field (name "Value") "Generic_Opamp")) (pins (pin (num "1") (name "") (type "input")) (pin (num "2") (name "-") (type "input")) (pin (num "3") (name "V") (type "power_in")) (pin (num "4") (name "V-") (type "power_in")) (pin (num "5") (name "") (type "output")))) (libpart (lib "sallen_key_schlib") (part "R") (footprints (fp "R**"*) (fp "Resistor_*")) (fields (field (name "Reference") "R") (field (name "Value") "R")) (pins (pin (num "1") (name "") (type "passive")) (pin (num "2") (name "") (type "passive")))) (libpart (lib "sallen_key_schlib") (part "VSOURCE") (fields (field (name "Reference") "V") (field (name "Value") "VSOURCE") (field (name "Fieldname") "Value") (field (name "Spice_Primitive") "V") (field (name "Spice_Node_Sequence") "1 2")) (pins (pin (num "1") (name "") (type "input")) (pin (num "2") (name "") (type "input"))))) (libraries (library (logical "sallen_key_schlib") (uri "/usr/share/kicad/demos/simulation/sallen_key/sallen_key_schlib.kicad_sym"))) (nets (net (code "1") (name "/lowpass") (node (ref "C1") (pin "1") (pintype "passive")) (node (ref "U1") (pin "2") (pinfunction "-") (pintype "input")) (node (ref "U1") (pin "5") (pintype "output"))) (net (code "2") (name "GND") (node (ref "C2") (pin "2") (pintype "passive")) (node (ref "V1") (pin "2") (pintype "input")) (node (ref "V2") (pin "2") (pintype "input")) (node (ref "V3") (pin "1") (pintype "input"))) (net (code "3") (name "Net-(C1-Pad2)") (node (ref "C1") (pin "2") (pintype "passive")) (node (ref "R1") (pin "1") (pintype "passive")) (node (ref "R2") (pin "2") (pintype "passive"))) (net (code "4") (name "Net-(C2-Pad1)") (node (ref "C2") (pin "1") (pintype "passive")) (node (ref "R2") (pin "1") (pintype "passive")) (node (ref "U1") (pin "1") (pinfunction "") (pintype "input"))) (net (code "5") (name "Net-(R1-Pad2)") (node (ref "R1") (pin "2") (pintype "passive")) (node (ref "V1") (pin "1") (pintype "input"))) (net (code "6") (name "VDD") (node (ref "U1") (pin "3") (pinfunction "V") (pintype "power_in")) (node (ref "V2") (pin "1") (pintype "input"))) (net (code "7") (name "VSS") (node (ref "U1") (pin "4") (pinfunction "V-") (pintype "power_in")) (node (ref "V3") (pin "2") (pintype "input")))))*

```
In Spice format, the netlist is as follows:
```

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

```
:experimental:

[[managing-symbol-libraries]]
== Managing Symbol Libraries

Symbol libraries hold collections of symbols used when creating schematics. Each symbol in
a schematic is uniquely identified by a full name that is composed of a library nickname
and a symbol name. An example is `Audio:AD1853`.

=== Symbol Library Table

KiCad uses a table of symbol libraries to map symbol libraries to a library nickname. Kicad
uses a global symbol library table as well as a table specific to each project. To edit
either symbol library table, use **Preferences** -> **Manage Symbol Libraries...**.

image::images/en/options_symbol_lib.png[scaledwidth="80%", alt="sym lib table dlg"]

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. xref:../kicad/kicad.adoc#config-file-
location[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.

==== Configuration Initiale

The first time the KiCad Schematic Editor is run and the global symbol table file `sym-lib-
table` is not found in the KiCad configuration folder, KiCad will guide the user through
setting up a new symbol library table. This process is described <<initial-
configuration,above>>.

==== Managing Table Entries

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

Add a library either by clicking the image:images/icons/small_folder_16.png[Folder icon]
button and selecting a library or clicking the image:images/icons/small_plus_16.png[Plus
icon] 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
image:images/icons/small_trash_16.png[Delete icon] button.

The image:images/icons/small_up_16.png[Up icon] and
image:images/icons/small_down_16.png[Down icon] buttons move the selected library up and
down in the library table. This does not affect the display order of libraries in the
Symbol Library Browser, Symbol Editor, or Add Symbol tool.

Libraries can be made inactive by unchecking the *Active* 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.

A range of libraries can be selected by clicking the first library in the range and then
kbd:[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 <<environment-variable-substitution,environment variable substitution>>.

The appropriate library format must be selected in order for the library to be properly
read. "KiCad" format is used for KiCad version 6 libraries (`.kicad_sym` files), while
"Legacy" format is used for libraries from older versions of KiCad (`.lib` files). Legacy
libraries are read-only, but can be migrated to KiCad format libraries using the **Migrate
Libraries** button (see section <<migrating-libraries,Migrating Legacy Libraries>>).

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.

[[environment-variable-substitution]]
==== Substitution d'une Variable d'Environnement

The symbol library tables support environment variable substitution, which allows you to
define environment variables containing custom paths to where your libraries are stored.
Environment variable substitution is supported by using the syntax `${ENV_VAR_NAME}` in the
symbol library path.

By default, KiCad defines several environment variables:

* `${KIPROJMOD}` points to the current project directory and cannot be modified.
* `${KICAD6_FOOTPRINT_DIR}` points to the default location of KiCad's standard footprint
libraries.
* `${KICAD6_SYMBOL_DIR}` points to the default location of KiCad's standard symbol
libraries.
* `${KICAD6_3DMODEL_DIR}` points to the default location of KiCad's standard 3D model
libraries.
* `${KICAD6_TEMPLATE_DIR}` points to the default location of KiCad's standard template
library.

`${KIPROJMOD}` cannot be redefined, but the other environment variables can be redefined
and new environment variables added in the **Preferences** -> **Configure Paths...**
dialog.

Using environment variables in the symbol library tables allows libraries to be relocated
without breaking the symbol library tables, so long as the environment variables are
updated when the library location changes.

`${KIPROJMOD}` allows libraries to be stored in the project folder without having to use an
absolute path in the project library table. This makes it possible to relocate projects
without breaking their project library tables. One of the most powerful features of the
symbol library table is environment variable substitution. This allows for definition of
custom paths to where symbol libraries are stored in environment variables. Environment
variable substitution is supported by using the syntax `${ENV_VAR_NAME}` in the library
path.

==== Scénarios d'Utilisation
```

```
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.

There are advantages and disadvantages to each method. Defining all libraries in the global
table means they will always be available when needed. The disadvantage of this is that
load time will increase.

Defining all symbol libraries on a project specific basis means that you only have the
libraries required for the project which decreases symbol library load times. The
disadvantage is that you always have to remember to add each symbol library that you need
for every project.

One usage pattern would be to define commonly used libraries globally and the libraries
only required for the project in the project specific library table. There is no
restriction on how to define libraries.

[[migrating-libraries]]
==== Migrating Legacy Libraries

Legacy libraries (`.lib` files) are read-only, but they can be migrated to KiCad version 6
libraries (`.kicad_sym`). KiCad version 6 libraries cannot be viewed or edited by KiCad
versions older than 6.0.0.

Legacy libraries can be converted to KiCad 6 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 kbd:[Ctrl]-clicking or kbd:[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:

- the original libraries used in the schematic are still available and unchanged from when
the symbol was added to the schematic.

- all rescue operations were performed when detected to create a rescue library or keep the
existing rescue library up to date.

- the integrity of the project symbol cache library has not been corrupted.

[WARNING]
====
The remapping will make a back up of all the files that are changed during remapping in the
rescue-backup folder in the project folder. Always make a back up of your project before
remapping just in case something goes wrong.
```

```
====

[WARNING]
====
The rescue operation is performed even if it has been disabled to ensure the correct
symbols are available for remapping. Do not cancel this operation or the remapping will
fail to correctly remap schematics symbols. Any broken symbol links will have to be fixed
manually.
====

[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.
====

:experimental:

[[symbol-editor]]
== Symbol Editor

[[general-information-about-symbol-libraries]]
=== General Information About Symbol Libraries

A symbol is a schematic element which contains a graphical representation, electrical
connections, and text fields describing the symbol. Symbols used in a schematic are stored
in symbol libraries. KiCad provides a symbol editing tool that allows you to create
libraries, add, delete or transfer symbols between libraries, export symbols to files, and
import symbols from files. The symbol editing tool provides a simple way to manage symbols
and symbol libraries.

[[symbol-library-overview]]
=== Symbol Library Overview

A symbol library is composed of one or more symbols. Generally the symbols are logically
grouped by function, type, and/or manufacturer.

A symbol is composed of:

* Graphical items (lines, circles, arcs, text, etc.) that determine how symbol looks in a
schematic.
* Pins which 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.
* Des champs : référence, valeur, empreintes correspondantes pour le dessin du circuit
imprimé, etc...

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.

Proper symbol designing requires:

* Defining if the symbol is made up of one or more units.
* Defining if the symbol has an alternate body style (also known as a De Morgan
representation).
* De dessiner sa représentation symbolique, au moyen de lignes, rectangles, cercles,
polygones, et de texte.
* D'ajouter des pins, en définissant leurs éléments graphiques, leurs noms, leurs numéros
et leurs propriétés électriques (entrées, sorties, trois-états, alimentations, etc..).
* Determining if the symbol should be derived from another symbol with the same graphical
design and pin definition.
* D'ajouter des champs supplémentaires, comme le nom de l'empreinte utilisée par le
logiciel de dessin du circuit imprimé, et de définir leur visibilité.
* Documenting the symbol by adding a description string and links to data sheets, etc.
* De le sauvegarder dans la librairie désirée.

[[symbol-library-editor-overview]]
=== Symbol Library Editor Overview

The symbol library editor main window is shown below. It consists of three tool bars for
quick access to common features and a symbol viewing/editing area. Not all commands are
available on the tool bars but can be accessed using the menus.

image::images/libedit_main_window.png[alt="Symbol Editor main window", scaledwidth="95%"]

[[main-toolbar]]
==== Barre d'outils principale

The main tool bar is located at the top of the main window. It consists of the undo/redo
commands, zoom commands, symbol properties dialogs, and unit/representation management
controls.

image::images/toolbar_libedit.png[alt="Symbol Editor toolbar", scaledwidth="95%"]

[width="100%", cols="20%,80%"]
|=========================================================================
|image:images/icons/new_component_24.png[New symbol icon]
|Create a new symbol in the selected library.

|image:images/icons/save_24.png[Save icon]
|Save the currently selected library. All modified symbols in the library will
be saved.

|image:images/icons/undo_24.png[Undo icon]
```

```
|Undo last edit.

|image:images/icons/redo_24.png[Redo icon]
|Redo last undo.

|image:images/icons/refresh_24.png[Refresh icon]|Refresh display.

|image:images/icons/zoom_in_24.png[Zoom in icon]|Zoom in.

|image:images/icons/zoom_out_24.png[Zoom out icon]|Zoom out.

|image:images/icons/zoom_fit_in_page_24.png[Zoom to fit page icon]|Zoom to fit symbol in
display.

|image:images/icons/zoom_area_24.png[Zoom to selection icon]|Zoom to fit selection.

|image:images/icons/rotate_ccw_24.png[Rotate counterclockwise icon]|Rotate counter-
clockwise.

|image:images/icons/rotate_cw_24.png[Rotate clockwise icon]|Rotate clockwise.

|image:images/icons/mirror_h_24.png[Mirror horizontally icon]|Mirror horizontally.

|image:images/icons/mirror_v_24.png[Mirror vertically icon]|Mirror vertically.

|image:images/icons/part_properties_24.png[Symbol properties icon]
|Edit the current symbol properties.

|image:images/icons/pin_table_24.png[Pin table icon]
|Edit the symbol's pins in a tablular interface.

|image:images/icons/datasheet_24.png[Datasheet icon]
|Open the symbol's datasheet. The button will be disabled if no datasheet is
defined for the current symbol.

|image:images/icons/erc_24.png[ERC icon]
|Test the current symbol for design errors.

|image:images/icons/morgan1_24.png[Normal body style icon]
|Select the normal body style. The button is disabled if the current
symbol does not have an alternate body style.

|image:images/icons/morgan2_24.png[Alternate body style icon]
|Select the alternate body style. The button is disabled if the current
symbol does not have an alternate body style.

|image:images/toolbar_libedit_part.png[alt="Unit dropdown",width="80%"]
|Select the unit to display. The drop down control will be disabled if
the current symbol is not derived from a symbol with multiple units.

|image:images/icons/pin2pin_24.png[Synchronized pin edit mode icon]
|Enable synchronized pins 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.

|image:images/icons/add_symbol_to_schematic_24.png[Add symbol to schematic icon]
Insert current symbol into schematic.

|=========================================================================

[[element-toolbar]]
==== Barre d'outils des éléments

The vertical toolbar located on the right hand side of the main window allows you to place
all of the elements required to design a symbol.

[width="100%", cols="10%,90%"]
|=========================================================================
|image:images/icons/cursor_24.png[Cursor icon]
|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.

|image:images/icons/pin_24.png[Pin icon]
|Pin tool. Left-click to add a new pin.

|image:images/icons/text_24.png[Text icon]
|Graphical text tool. Left-click to add a new graphical text item.

|image:images/icons/add_rectangle_24.png[Add rectangle icon]
|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.

|image:images/icons/add_circle_24.png[Add circle icon]
|Circle tool. Left-click to begin drawing a new graphical circle from
the center. Left-click again to define the radius of the circle.

|image:images/icons/add_arc_24.png[Add arc icon]
|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.

|image:images/icons/add_line_24.png[Add line icon]
|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.

|image:images/icons/anchor_24.png[Anchor icon]
|Anchor tool. Left-click to set the anchor position of the symbol.
```

```
|image:images/icons/delete_cursor_24.png[Delete icon]
|Delete tool. Left-click to delete an object from the current symbol.
|=========================================================================

[[options-toolbar]]
==== Barre d'outils des options

The vertical tool bar located on the left hand side of the main window allows you to set
some of the editor drawing options.

[width="100%", cols="10%,90%"]
|=========================================================================
|image:images/icons/grid_24.png[Grid icon]
|Toggle grid visibility on and off.

|image:images/icons/unit_inch_24.png[Inch unit icon]
|Set units to inches.

|image:images/icons/unit_mil_24.png[Millimeter unit icon]
|Set units to mils (0.001 inch).

|image:images/icons/unit_mm_24.png[Millimeter unit icon]
|Set units to millimeters.

|image:images/icons/cursor_shape_24.png[Cursor shape icon]
|Toggle full screen cursor on and off.

|image:images/icons/pin_show_etype_24.png[Show pintype icon]
|Toggle display of pin electrical types.

|image:images/icons/search_tree_24.png[Symbol tree icon]
|Toggle display of libraries and symbols.
|=========================================================================

[[library-selection-and-maintenance]]
=== Sélection et gestion des librairies

The selection of the current library is possible via the
image:images/icons/search_tree_24.png[Symbol tree icon] icon which shows you all available
libraries and allows you to select one. When a symbol is loaded or saved, it will be put in
this library. The library name of a symbol is the contents of its `Value` field.

[[select-and-save-a-symbol]]
==== Select and Save a Symbol

[[symbol-selection]]
===== Symbol Selection

Clicking the image:images/icons/search_tree_24.png[Symbol tree icon] icon on the left tool
bar toggles the treeview of libraries and symbols. Clicking on a symbol opens that 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.

[[save-a-symbol]]
===== Save a 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
image:images/icons/save_24.png[Save icon] icon. The modifications will be written to the
existing symbol.

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. A new name can be set for the
symbol.

To create a new file containing only the current symbol, click **File** -> **Export** ->
**Symbol...**. This file will be a standard library file which will contain only one
symbol.

[[creating-library-symbols]]
=== Creating Library Symbols

[[create-a-new-symbol]]
==== Create a New Symbol

A new symbol can be created by clicking the image:images/icons/new_component_24.png[New
symbol icon] icon. You will be asked for a number of symbol properties.

* A symbol name (this name is used as the default value for the `Value` field in the
schematic editor)
* 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 is made of 4 units per package).
* 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 Port" dialog
in the Schematic editor, their `Value` fields are not editable in the schematic, they
cannot be assigned a footprint and they are not added to the PCB, and they are not included
in the 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, Symbol Properties
window>>.

image::images/eeschema_new_symbol_properties.png[alt="New symbol properties",
scaledwidth="50%"]

A new symbol will be created using the properties above and will appear in the editor as
shown below.

image::images/eeschema_libedit_new.png[alt="Newly created symbol", scaledwidth="95%"]

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
image:images/icons/anchor_24.png[Anchor icon] icon and clicking on the new desired anchor
position.

[[create-a-symbol-from-another-symbol]]
==== Create a Symbol from Another Symbol

Often, the symbol that you want to make is similar to one already in a symbol library. In
this case it is easy to load and modify an existing symbol.

* Load the symbol which will be used as a starting point.
* Save a new copy of the symbol using **File** -> **Save As...**. The Save As dialog will
prompt for a name for the new symbol and the library to save it in.
* Edit the new symbol as required.
* Save the modified symbol.

[[symbol-properties]]
==== Symbol Properties

Symbol properties are set when the symbol is created but they can be modified at any point.
To change the symbol properties, click on the
image:images/icons/part_properties_24.png[Symbol properties icon] icon to show the dialog
below.

image::images/eeschema_properties_for_symbol.png[alt="Symbol Properties",
scaledwidth="60%"]

It is important to correctly set the number of units per package and the alternate symbolic
representation, if enabled, because when pins are edited or created the corresponding pins
for each unit will be affected. If you change the number of units per package after pin
creation and editing, there will be additional work to specify the pins and graphics for
the new unit. Nevertheless, it is possible to modify these properties at any time.
```

```
The graphic options "Show pin number" and "Show pin name" define the visibility of the pin
number and pin name text. The option "Place pin names inside" defines 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.

The example below shows a symbol with the "Place pin name inside" option unchecked. Notice
the position of the names and pin numbers.

image::images/eeschema_uncheck_pin_name_inside.png[alt="Place pin name inside unchecked",
scaledwidth="95%"]

[[symbol-name-description-and-keywords]]
===== Symbol Name, Description, and Keywords

The symbol's name is the same as the `Value` field. When the symbol name is changed the
value also changes, and vice versa. The symbol's name in the library also changes
accordingly.

The symbol description should contain a brief description of the component, such as the
component function, distinguishing features, and package options. The keywords should
contain additional terms related to the component. Keywords are used primarily to assist in
searching for the symbol.

image::images/eeschema_add_symbol_search_description.png[alt="Searching for a symbol in the
add a symbol dialog", scaledwidth="65%"]

A symbol's name, description, and keywords are all used when searching for symbols in the
Symbol Editor and Add a Symbol dialog. The description and keywords are displayed in the
Symbol Library Browser and Add a Symbol dialog.

[[footprint-filters]]
===== 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`.

image::images/eeschema_libedit_footprint.png[alt="Footprint filters", scaledwidth="70%"]

[[symbols-with-alternate-symbolic-representation]]
==== Symbols with Alternate Symbolic Representation
```

```
If the symbol has an alternate body style defined, one body style must be selected for
editing at a time. To edit the normal representation, click the
image:images/icons/morgan1_24.png[Normal representation icon] icon.

To edit the alternate representation, click on the
image:images/icons/morgan2_24.png[Alternate representation icon] icon. Use the
image:images/toolbar_libedit_alias.png[images/toolbar_libedit_part.png] dropdown shown
below to select the unit you wish to edit.

image::images/eeschema_libedit_select_unit.png[alt="Selecting a symbol unit",
scaledwidth="80%"]

[[graphical-elements]]
=== Éléments graphiques

Graphical elements create the visual representation of a symbol and contain no electrical
connection information. Graphical elements are created with the following tools:

* Lignes et polygones définis par des points d'origine et des points de fin.
* Rectangles définis par leurs deux coins opposés sur la diagonale.
* Cercles définis par leur centre et leur rayon.
* Arcs de cercles définis par leur centre et leurs points de départ et de fin. Un arc peut
aller de 0 à 180°.

The vertical toolbar on the right hand side of the main window allows you to place all of
the graphical elements required to design the representation of a symbol.

[[graphical-element-membership]]
==== Appartenance des éléments graphiques

Chaque élément graphique, (ligne, arc, cercle, etc...), peut être défini comme commun à
toutes les unités et/ou représentations, ou spécifique à une unité donnée et/ou une
représentation. Les options des éléments sont accessibles rapidement par le menu contextuel
: clic droit sur l'élément à modifier. Ci-dessous, le menu contextuel pour un élément de
type ligne.

image::images/eeschema_libedit_context_menu.png[alt="Graphic line context menu",
scaledwidth="80%"]

Vous pouvez aussi double-cliquer sur un élément et modifier ses propriétés. Ci-dessous, la
fenêtre des propriétés pour un élément de type polygone.

image::images/eeschema_libedit_polyline_properties.png[alt="Graphic line properties",
scaledwidth="50%"]

Les propriétés d'un élément graphique sont :

* "Line width" defines the width of the element's line in the current drawing units.
* "Fill Style" determines if the shape defined by the graphical element is to be drawn
unfilled, background filled, or foreground filled.
* "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 (De Morgan)" 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.

[[graphical-text-elements]]
==== Éléments Graphiques Textes

The image:images/icons/text_24.png[Text icon] icon allows for the creation of graphical
text. Graphical text is automatically oriented to be readable, even when the symbol is
mirrored. Please note that graphical text items are not the same as symbol fields.

[[multiple-units-per-symbol-and-alternate-body-styles]]
=== Multiple Units per Symbol and Alternate Body Styles

Symbols can have up to two body styles (a standard symbol and an alternate symbol often
referred to as a "De Morgan equivalent") and/or have more than one unit per package (logic
gates for example). Some symbols can have more than one unit per package each with
different symbols and pin configurations.

Consider for instance a relay with two switches, which can be designed as a symbol with
three different units: a coil, switch 1, and switch 2. Designing a symbol with multiple
units per package and/or alternate body styles is very flexible. A pin or a body symbol
item can be common to all units or specific to a given unit or they can be common to both
symbolic representation so are specific to a given symbol representation.

By default, pins are specific to a unit and body style. When a pin is common to all units
or all body styles, it only needs to be created once. This is also the case for the body
style graphic shapes and text, which may be common to each unit, but typically are specific
to each body style).

[[example-of-a-symbol-with-multiple-noninterchangeable-units]]
==== Example of a Symbol With Multiple Noninterchangeable Units

For an example of a symbol with multiple units that are not interchangeable, consider a
relay with 3 units per package: a coil, switch 1, and switch 2.

The three units are not all the same, so "All units are interchangeable" should be
deselected in the Symbol Properties dialog. Alternatively, this option could have been
specified when the symbol was initially created.

image::images/eeschema_libedit_not_interchangeable.png[alt="Uncheck all units are
interchangeable", scaledwidth="60%"]

===== Unit A

image::images/eeschema_libedit_unit1.png[alt="Relay unit A", scaledwidth="45%"]

===== Unit B

image::images/eeschema_libedit_unit2.png[alt="Relay unit B", scaledwidth="45%"]
```

```
===== Unit C

image::images/eeschema_libedit_unit3.png[alt="Relay unit C", scaledwidth="45%"]

Unit A does not have the same symbol and pin layout as Units B and C, so the units are not
interchangeable.

NOTE: "Synchronized Pins Edit Mode" can be enabled by clicking the
image:images/icons/pin2pin_24.png[Synchronized pins edit mode icon] 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.

[[graphical-symbolic-elements]]
===== Éléments graphiques symboliques

Shown below are properties for a graphic body element. In the relay example above, the
three units have different symbolic representations. Therefore, each unit was created
separately and the graphical body elements have the "Common to all units in symbol" setting
disabled.

image::images/eeschema_libedit_disable_common.png[alt="Disable common to all units in
symbol", scaledwidth="70%"]

[[pin-creation-and-editing]]
=== Création et édition de pins

You can click on the image:images/icons/pin_24.png[Pin icon] icon to create and insert a
pin. The editing of all pin properties is done by double-clicking on the pin or right-
clicking on the pin to open the pin context menu. Pins must be created carefully, because
any error will have consequences on the PCB design. Any pin already placed can be edited,
deleted, and/or moved.

[[pin-overview]]
==== Généralités sur les pins

A pin is defined by its graphical representation, its name and its number. The pin's name
and number can contain letters, numbers, and symbols, but not spaces. For the Electrical
Rules Check (ERC) tool to be useful, the pin's electrical type (input, output, tri-
state...) must also be defined correctly. If this type is not defined properly, the
schematic ERC check results may be invalid.

Notes importantes :

* Symbol pins are matched to footprint pads by number. The pin number in the symbol must
match the corresponding pad number in the footprint.
* Do not use spaces in pin names and numbers. Spaces will be automatically replaced with
underscores (`_`).
* To define a pin name with an inverted signal (overline) use the `~` (tilde) character
followed by the text to invert in braces. For example `~{FO}O` would
```

```
display [overline]#FO# O.
* If the pin name is empty, the pin is considered unnamed.
* Pin names can be repeated in a symbol.
* Pin numbers must be unique in a symbol.

[[pin-properties]]
==== Propriétés des pins

image::images/eeschema_libedit_pin_properties.png[alt="Pin properties", scaledwidth="95%"]

La fenêtre des propriétés des pins vous permet de modifier toutes les caractéristiques
d'une pin. Cette fenêtre apparaît automatiquement à la création de la pin ou quand vous
double-cliquez sur une pin existante. Vous pouvez modifier :

* The pin name and text size.
* The pin number and text size.
* The pin length.
* The pin electrical type and graphical style.
* Son appartenance aux unités et aux représentations alternatives.
* Pin visibility.
* <<alternate-pin-definitions,Alternate pin definitions>>.

[[pin-graphic-styles]]
==== Pin Graphic Styles

Shown in the figure below are the different pin graphic styles. The choice of graphic style
does not have any influence on the pin's electrical type.

image::images/eeschema_libedit_pin_properties_style.png[alt="Pin graphic styles",
scaledwidth="95%"]

[[pin-electrical-types]]
==== Types électriques des pins

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.

[width="100%", cols="25%,75%"]
|=========================================================================
| Pin Type | Description
| Input | A pin which is exclusively an input.
| Output | A pin which is exclusively an output.
| Bidirectional | A pin that can be either an input or an output, such as a
microcontroller data bus pin.
| Tri-state | A three state output pin (high, low, or high impedance)
| Passive | A passive symbol pin: resistors, connectors, etc.
| Free | A pin that can be freely connected to any other pin without electrical
concerns.
| Unspecified | A pin for which the ERC check does not matter.
| Power input | A symbol's power pin. As a special case, power input pins that
```

```
are marked invisible are automatically connected to the net with the same name.
See the <<creating-power-ports, Power Ports section>> for more information.
| Power output | A pin that provides power to other pins, such as a regulator
output.
| Open collector | An open collector logic output.
| Open emitter | An open emitter logic output.
| Unconnected | A pin that should not be connected to anything.
|======================================================================

[[pushing-pin-properties-to-other-pins]]
==== 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.

image::images/eeschema_libedit_pin_context_menu.png[alt="Pin context menu",
scaledwidth="60%"]

[[defining-pins-for-multiple-units-and-alternate-symbolic-representations]]
==== Définitions de pins pour unités multiples et représentations alternatives

Symbols with multiple units and/or graphical representations are particularly problematic
when creating and editing pins. The majority of 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
image:images/icons/pin2pin_24.png[Synchronized pin edit mode icon] 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 clicking the
image:images/icons/morgan2_24.png[Alternate representation icon] button on the tool bar. To
edit the pin number for each unit, select the appropriate unit using the
image:images/toolbar_libedit_alias.png[images/toolbar_libedit_alias.png] drop down control.
```

```
[[pin-table]]
==== Pin Table

Another way to edit pins is to use the Pin Table, which is accessible via the
image:images/icons/pin_table_24.png[Pin table icon] icon. 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.

Any pin property can be edited by clicking on the appropriate cell. Pins can be added and
removed with the image:images/icons/small_plus_16.png[Plus icon] and
image:images/icons/small_trash_16.png[Trash icon] 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.

The screenshot below shows the pin table for a quad opamp.

image::images/eeschema_libedit_pin_table.png[alt="Pin table", scaledwidth="95%"]

[[alternate-pin-definitions]]
==== Alternate Pin Definitions

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

Alternate pin definitions 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.

image::images/eeschema_libedit_alternate_pin_definitions.png[alt="Alternate pin
definitions", scaledwidth="60%"]

Alternate pin definitions are selected in the Schematic Editor once the symbol has been
placed in the schematic. The alternate pin is assigned in the Alternate Pin Assignments tab
of the Symbol Properties dialog. Alternate definitions are selectable in the dropdown in
the Alternate Assignment column.

image::images/eeschema_alternate_pin_assignment_selection.png[alt="Selecting an alternate
pin definition", scaledwidth="60%"]

[[symbol-fields]]
=== Symbol Fields

All library symbols are defined with four default fields. The reference designator, value,
footprint assignment, and datasheet link fields are created whenever a symbol is created or
copied. Only the reference designator and value fields are required.
```

```
Symbols defined in libraries are typically defined with only these four 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
applied to all of the symbols in the schematic.

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.

[[editing-symbol-fields]]
==== Editing Symbol Fields

To edit an existing symbol field, right-click on the field text to show the field context
menu shown below.

image::images/eeschema_libedit_field_context_menu.png[alt="Symbol field context menu",
scaledwidth="35%"]

To add new fields, delete optional fields, or edit existing fields, use the
image:images/icons/part_properties_24.png[Component properties icon] icon on the main tool
bar to open the <<symbol-properties,Symbol Properties dialog>>.

Fields are text information associated a the symbol. Do not confuse them with text in the
graphic representation of a symbol.

Notes importantes :

* Modifying the `Value` field changes the name of the symbol. The symbol's name in the
library will change when the symbol is saved.
* The Symbol Properties dialog must be used to edit a field that is empty or has the
invisible attribute enabled because such fields cannot be clicked on.
* The footprint is defined as an absolute footprint using the `LIBNAME:FOOTPRINTNAME`
format where `LIBNAME` is the name of the footprint library defined 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`.

[[creating-power-ports]]
=== Creating Power Port Symbols

Power ports, or power symbols, are symbols that are used to label a wire as part of a power
net, like `VCC` or `GND`. The behavior of power ports is described in the <<power-
ports,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 Port" dialog for convenient
access.
```

```
Below is an example of a `GND` power symbol.

image::images/eeschema_libedit_power_symbol.png[alt="Editing a power symbol",
scaledwidth="95%"]

Power port symbols consist of a pin of type "Power input" that is marked invisible. They
must also have the "Define as power symbol" property checked. <<hidden-power-pins,Invisible
power input pins>> have a special property of making implicit global connections based on
the pin name.

NOTE: If the power symbol has the "Define as power symbol" property checked, the power
input pin does not need to be marked invisible. However, the convention is to make these
pins invisible anyway.

image::images/eeschema_libedit_power_symbol_pin.png[alt="Power symbol pin",
scaledwidth="60%"]

Pour créer un symbole d'alimentation, utilisez les étapes suivantes :

* Add a pin of type "Power input", with "Visible" unchecked, and the pin named according to
the desired net. Make the pin number `1`, the length `0`, and set the graphic style to
"Line". The pin name establishes the connection to the net; in this case the pin will
automatically connect to the net `GND`. The pin number, length, and line style do not
matter electrically.
* Place the pin on the symbol anchor.
* Use the shape tools to draw the symbol graphics.
* Set the symbol value. The symbol value does not matter electrically, but it is displayed
in the schematic. To eliminate confusion, it should match the pin name (which determines
the connected net name).
* Check the "Define as power symbol" box in Symbol Properties window. This makes the symbol
appear in the "Add Power Port" dialog, makes the `Value` field read-only in the schematic,
prevents the symbol from being assigned a footprint, and excludes the symbol from the
board, BOM, and netlists.
* 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 port 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 port'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 port symbol is to use another symbol as a starting
point, <<creating-a-symbol-from-another-symbol,as described earlier>>.

NOTE: The connected net name is determined by the power port's *pin name*, not
      the name or value of the symbol. When modifying an existing power port
      symbol, make sure to rename the pin so that the new symbol connects to the
      appropriate power net. This means that power port net names can only be
      changed in the symbol editor, not in the schematic.
```

```
:experimental:

[[viewlib]]
== Symbol Library Browser

=== Introduction

The Symbol Library Browser allows you to quickly examine the content of symbol libraries.
The Symbol Library Viewer can be accessed by clicking
image:images/icons/library_browser_24.png[Library viewer icon] icon on the main toolbar,
**View** -> **Symbol Library Browser...**, or clicking **Select With Browser** in the
"Choose Symbol" window.

image::images/eeschema_viewlib_choose.png[alt="eeschema_viewlib_choose_png",
scaledwidth="60%"]

[[viewlib---main-screen]]
=== Viewlib - fenêtre principale

image::images/eeschema_viewlib_select_library.png[alt="eeschema_viewlib_select_library_png"
, scaledwidth="95%"]

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

image::images/eeschema_viewlib_select_component.png[alt="eeschema_viewlib_select_component_
png", scaledwidth="95%"]

[[viewlib-top-toolbar]]
=== Symbol Library Browser Top Toolbar

The top tool bar in Symbol Library Brower is shown below.

image::images/toolbar_viewlib.png[alt="images/toolbar_viewlib.png", scaledwidth="95%"]

The available commands are:

[width="100%", cols="20%,80%"]
|=========================================================================
|image:images/icons/library_browser_24.png[Symbol selection icon]
|Selection of the symbol which can be also selected in the displayed
list.

|image:images/icons/lib_previous_24.png[Previous symbol icon]
|Display previous symbol.

|image:images/icons/lib_next_24.png[Next symbol icon]
|Display next symbol.

|image:images/icons/refresh_24.png[] image:images/icons/zoom_in_24.png[]
image:images/icons/zoom_out_24.png[] image:images/icons/zoom_fit_in_page_24.png[]
```

```
|Zoom tools.

|image:images/icons/morgan1_24.png[] image:images/icons/morgan2_24.png[]
|Selection of the representation (normal or alternate) if an alternate
representation exists.

|image:images/toolbar_viewlib_part.png[alt="images/toolbar_viewlib_part.png",width="70%"]
|Selection of the unit for symbols that contain multiple units.

|image:images/icons/datasheet_24.png[icons/datasheet_png]
|If they exist, display the associated documents.

|image:images/icons/add_symbol_to_schematic_24.png[Add symbol to schematic icon]
|Close the browser and place the selected symbol in the schematic.
|==========================================================================

:experimental:

[[simulator]]
== Simulator ==

KiCad provides an embedded electrical circuit simulator using
http://ngspice.sourceforge.net[ngspice] as the simulation engine.

When working with the simulator, you may find the official _pspice_ library useful. It
contains common symbols used for simulation like voltage/current sources or transistors
with pins numbered to match the ngspice node order specification.

There are also a few demo projects to illustrate the simulator capabilities. You will find
them in _demos/simulation_ directory.

=== Assigning models

Before a simulation is launched, components need to have Spice model assigned.

Each component can have only one model assigned, even if component consists of multiple
units. In such case, the first unit should have the model specified.

[[sim-passive-models]] Passive components with reference matching a device type in Spice
notation (_R*_ for resistors, _C*_ for capacitors, _L*_ for inductors) will have models
assigned implicitly and use the value field to determine their properties.

[NOTE]
Keep in mind that in Spice notation 'M' stands for milli and 'Meg' corresponds to mega. If
you prefer to use 'M' to indicate mega prefix, you may request doing so in the <<sim-
settings, simulation settings dialog>>.

Spice model information is stored as text in symbol fields, therefore you may either define
it in symbol editor or schematics editor. Open symbol properties dialog and click on _Edit
Spice Model_ button to open Spice Model Editor
```

```
dialog.

Spice Model Editor dialog has three tabs corresponding to different model types. There are
two options common to all model types:

[width="90%", cols="30%a,70%a"]
|====
|Disable symbol for simulation
|When checked the component is excluded from simulation.
|Alternate node sequence
|Allows one to override symbol pin to model node mapping.
To define a different mapping, specify pin numbers in order expected by the model.

'Example:' +
____
`* connections:` +
`* 1: non-inverting input` +
`* 2: inverting input` +
`* 3: positive power supply` +
`* 4: negative power supply` +
`* 5: output` +
`.subckt tl071 1 2 3 4 5`
____

image::images/opamp_symbol.png[alt="Generic operational amplifier symbol"]

To match the symbol pins to the Spice model nodes shown above, one needs to use an
alternate node sequence option with value: "1{nbsp}3{nbsp}5{nbsp}2{nbsp}4". It is a list of
pin numbers corresponding to the Spice model nodes order.
|====

==== Passive

_Passive_ tab allows the user to assign a passive device model (resistor, capacitor or
inductor) to a component. It is a rarely used option, as normally passive components have
models assigned <<sim-passive-models,implicitly>>, unless component reference does not
match the actual device type.

[NOTE]
Explicitly defined passive device models have priority over the ones assigned implicitly.
It means that once a passive device model is assigned, the reference and value fields are
not taken into account during simulation. It may lead to a confusing situation when
assigned model value does not match the one displayed on a schematic sheet.

image::images/sim_model_passive.png[alt="Passive device model editor tab"]

[width="90%", cols="30%a,70%a"]
|====
|Type
|Selects the device type (resistor, capacitor or inductor).
|Value
|Defines the device property (resistance, capacitance or inductance). The value
```

```
may use common Spice unit prefixes (as listed below the text input field) and
should use point as the decimal separator. Note that Spice does not correctly
interpret prefixes intertwined in the value (e.g. 1k5).
|====

==== Model

_Model_ tab is used to assign a semiconductor or a complex model defined in an external
library file. Spice model libraries are often offered by device manufacturers.

The main text widget displays the selected library file contents. It is a common practice
to put models description inside library files, including the node order.

image::images/sim_model_subckt.png[alt="Semiconductor device model editor tab"]

[width="90%", cols="30%a,70%a"]
|====================
|File
|Path to a Spice library file. This file is going to be used by the simulator,
as it is added using _.include_ directive.
|Model
|Selected device model. When a file is selected, the list is filled with available
models to choose from.
|Type
|Selects model type (subcircuit, BJT, MOSFET or diode). Normally it is set
automatically when a model is selected.
|====================

==== Source

_Source_ tab is used to assign a power or signal source model. There are two sections:
_DC/AC analysis_ and _Transient analysis_. Each defines source parameters for the
corresponding simulation type.

_Source type_ option applies to all simulation types.

image::images/sim_model_source.png[alt="Source model editor tab"]

Refer to the http://ngspice.sourceforge.net/docs/ngspice-27-manual.pdf[ngspice
documentation], chapter 4 (Voltage and Current Sources) for more details about sources.

[[sim-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. This functionality is
limited to Spice directives starting with a dot (e.g. `.tran 10n 1m`), it is not possible
to place additional components using text fields.
```

```
=== Simulation

To launch a simulation, open _Spice Simulator_ dialog by selecting menu _Tools->Simulator_
in the schematics editor window.

image::images/sim_main_dialog.png[alt="Main simulation dialog"]

The dialog is divided into several sections:

* <<sim-toolbar,Toolbar>>
* <<sim-plot-panel,Plot panel>>
* <<sim-output-console,Output console>>
* <<sim-signals-list,Signals list>>
* <<sim-cursors-list,Cursors list>>
* <<sim-tune-panel,Tune panel>>

==== Menu

[[sim-menu-file]]
===== File
[width="90%", cols="30%,70%"]
|====
|New Plot | Create a new tab in the plot panel.
|Open Workbook | Open a list of plotted signals.
|Save Workbook | Save a list of plotted signals.
|Save as image | Export the active plot to a .png file.
|Save as .csv file | Export the active plot raw data points to a .csv file.
|Exit Simulation | Close the dialog.
|====

[[sim-menu-simulation]]
===== Simulation
[width="90%", cols="30%,70%"]
|====
|Run Simulation | Perform a simulation using the current settings.
|Add signals... | Open a dialog to select signals to be plotted.
|Probe from schematics | Start the schematics <<sim-probe-tool,Probe>> tool.
|Tune component value | Start the <<sim-tuner-tool,Tuner>> tool.
|Show SPICE Netlist... | Open a dialog showing the generated netlist for the
simulated circuit.
|Settings... | Open the <<sim-settings,simulation settings dialog>>.
|====

[[sim-menu-view]]
===== View
[width="90%", cols="30%,70%"]
|====
|Zoom In | Zoom in the active plot.
|Zoom Out | Zoom out the active plot.
|Fit on Screen | Adjust the zoom setting to display all plots.
|Show grid | Toggle grid visibility.
|Show legend | Toggle plot legend visibility.
|====
```

```
[[sim-toolbar]]
==== Toolbar
image::images/sim_main_toolbar.png[alt="Simulation dialog top toolbar"]
The top toolbar provides access to the most frequently performed actions.

[width="90%", cols="30%,70%"]
|====
|Run/Stop Simulation | Start or stop the simulation.
|Add Signals | Open a dialog to select signals to be plotted.
|Probe | Start the schematics <<sim-probe-tool,Probe>> tool.
|Tune | Start the <<sim-tuner-tool,Tuner>> tool.
|Settings | Open the <<sim-settings,simulation settings dialog>>.
|====

[[sim-plot-panel]]
==== Plot panel
Visualizes the simulation results as plots. One can have multiple plots opened in separate
tabs, but only the active one is updated when a simulation is executed. This way it is
possible to compare simulation results for different runs.

Plots might be customized by toggling grid and legend visibility using <<sim-menu-
view,View>> menu. When a legend is visible, it can be dragged to change its position.

Plot panel interaction:

* scroll mouse wheel to zoom in/out
* right click to open a context menu to adjust the view
* draw a selection rectangle to zoom in the selected area
* drag a cursor to change its coordinates

[[sim-output-console]]
==== Output console
Output console displays messages from the simulator. It is advised to check the console
output to verify there are no errors or warnings.

[[sim-signals-list]]
==== Signals list
Shows the list of signals displayed in the active plot.

Signals list interaction:

* right click to open a context menu to hide signal or toggle cursor
* double click to hide signal

[[sim-cursors-list]]
==== Cursors list
Shows the list of cursors and their coordinates. Each signal may have one cursor displayed.
Cursors visibility is set using the <<sim-signals-list,Signals>> list.

[[sim-tune-panel]]
==== Tune panel
```

```
Displays components picked with the <<sim-tuner-tool,Tuner>> tool. Tune panel allows the
user to quickly modify component values and observe their influence on the simulation
results - every time a component value is changed, the simulation is rerun and plots are
updated.

For each component there a few controls associated:

* The top text field sets the maximum component value.
* The middle text field sets the actual component value.
* The bottom text field sets the minimum component value.
* Slider allows the user to modify the component value in a smooth way.
* _Save_ button modifies component value on the schematics to the one selected with the
slider.
* _X_ button removes component from the Tune panel and restores its original value.

The three text fields recognize Spice unit prefixes.

[[sim-tuner-tool]]
==== Tuner tool
Tuner tool lets the user pick components for tuning.

To select a component for tuning, click on one in the schematics editor when the tool is
active. Selected components will appear in the <<sim-tune-panel,Tune>> panel. Only passive
components might be tuned.

[[sim-probe-tool]]
==== Probe tool
Probe tool provides an user-friendly way of selecting signals for plotting.

To add a signal to plot, click on a corresponding wire in the schematics editor when the
tool is active.

[[sim-settings]]
==== Simulation settings

image::images/sim_settings.png[alt="Simulation settings dialog"]

Simulation settings dialog lets the user set the simulation type and parameters. There are
four tabs:

* AC
* DC Transfer
* Transient
* Custom

The first three tabs provide forms where simulation parameters might be specified. The last
tab allows the user to type in custom Spice directives to set up a simulation. You can find
more information about simulation types and parameters in the
http://ngspice.sourceforge.net/docs/ngspice-27-manual.pdf[ngspice documentation], chapter
1.2.

An alternative way to configure a simulation is to type <<sim-directives,Spice directives>>
into text fields on schematics. Any text field directives related
```

```
to simulation type are overridden by the settings selected in the dialog. It means that
once you start using the simulation dialog, the dialog overriddes the schematics directives
until the simulator is reopened.

There are two options common to all simulation types:
[width="90%", cols="30%,70%"]
|====
|Adjust passive symbol values | Replace passive symbol values to convert common
component values notation to Spice notation.
|Add full path for .include library directives | Prepend Spice model library
file names with full path. Normally full path is required by ngspice to access
a library file.
|====

:experimental:

[[advanced]]
== Advanced Topics

[[color-settings]]
[[configuration-and-customization]]
=== Configuration and Customization

NOTE: TODO: write this section

[[text-variables]]
=== Text variables

NOTE: TODO: write this section

////
[[preferences-common]]
===== Common Preferences
NOTE: TODO: write this section

image::images/en/options_common.png[alt="Common settings",scaledwidth="70%"]

[[preferences-mouse]]
===== Mouse and Touchpad
[width="100%",cols="40%,60%",]
|============
|Center and warp cursor on zoom | If checked, the pointed location is warped
to the screen center when zooming in/out.

|Use touchpad to pan |When enabled, view is panned using scroll wheels (or
touchpad gestures) and to zoom one needs to hold kbd:[Ctrl]. Otherwise scroll
wheels zoom in/out and kbd:[Ctrl]/kbd:[Shift] are the panning modifiers.

|Pan while moving object |If checked, automatically pans the window
if the cursor leaves the window during drawing or moving.
|============

[[preferences-controls]]
===== Hotkeys
```

```
Redefine hotkeys.

image::images/en/options_hotkeys.png[alt="Hotkeys settings",scaledwidth="70%"]

Select a new hotkey by double clicking an
action or right click on an action to show a popup menu:

[width="100%",cols="40%,60%",]
|============
|Edit | Define a new hotkey for the action (same as double click).
|Undo Changes | Reverts the recent hotkey changes for the action.
|Clear Assigned Hotkey |
|Restore Default | Sets the action hotkey to its default value.
|============

[[preferences-display]]
===== Display Options

image::images/en/options_display.png[alt="Display options",scaledwidth="70%"]

[width="100%",cols="40%,60%",]
|=====================================================================
|Grid Size| Grid size selection.

It is *recommended* to work with normal grid (0.050 inches or 1,27 mm). Smaller
grids are used for component building.

|Bus thickness |Pen size used to draw buses.

|Line thickness |Pen size used to draw objects that do not have a
specified pen size.

|Part ID notation |Style of suffix that is used to denote symbol units (U1A,
U1.A, U1-1, etc.)

|Icon scale| Adjust toolbar icons size.

|Show Grid | Grid visibility setting.

|Restrict buses and wires to H and V orientation| If checked, buses and
wires are drawn only with vertical or horizontal lines.
Otherwise buses and wires can be placed at any orientation.

|Show hidden pins: |Display invisible (or __hidden__) pins, typically
power pins.

|Show page limits |If checked, shows the page boundaries on screen.

|Footprint previews in symbol chooser| Displays a footprint preview frame and
footprint selector when placing a new symbol.

*Note:* it may cause problems or delays, use at your own risk.
|=====================================================================
```

```
[[preferences-editing]]
===== Editing Options

image::images/en/options_editing.png[alt="Editing settings",scaledwidth="70%"]

[width="100%",cols="40%,60%",]
|=========================================================================
|Measurement units |Select the display and the cursor coordinate units
(inches or millimeters).

|Horizontal pitch of repeated items |
Increment on X axis during element duplication (default: 0)
(after placing an item like a symbol, label or wire,
a duplication is made by the kbd:[Insert] key)

|Vertical pitch of repeated items| Increment on Y axis during
element duplication (default: 0.100 inches or 2,54 mm).

|Increment of repeated labels |Increment of label value during duplication of texts ending
in a number, such as bus members (usual value 1 or -1).

|Default text size |Text size used when creating new text items or labels.

|Auto-save time interval |Time in minutes between saving backups.

|Automatically place symbol fields | If checked, symbol fields (e.g. value and
reference) in newly placed symbols might be moved to avoid collisions with
other items.

|Allow field autoplace to change justification | Extension of 'Automatically
place symbol fields' option. Enable text justification adjustment for symbol fields when
placing
a new part.

|Always align autoplaced fields to the 50 mil grid |Extension of 'Automatically
place symbol fields' option. If checked, fields are autoplaced using 50 mils
grid, otherwise they are placed freely.
|=========================================================================

[[preferences-colors]]
===== Colors

Color scheme for various graphic elements. Click on any of the color swatches to
select a new color for a particular element.

image::images/en/options_color.png[alt="Color settings",scaledwidth="95%"]

[[preferences-default-fields]]
===== Default Fields

Define additional custom fields and corresponding values that will appear in
newly placed symbols.
```

```
image::images/en/options_default_fields.png[alt="Default Fields
settings",scaledwidth="70%"]
////

[[custom-netlist-and-bom-formats]]
=== 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,netlist export
section>>. BOM output is described in the <<bom-export,BOM export section>>.

The following section describes how to create an exporter for a new output format.

[[adding-new-netlist-generators]]
==== Adding new netlist generators

New netlist generators are added to the **Export Netlist** dialog by clicking the **Add
Generator...** button.

image::images/eeschema_netlist_dialog_add_plugin.png[alt="Custom Netlist Generator",
scaledwidth="40%"]

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 <<generator-command-line-
format,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 <<bom-
export,Generate BOM tool>>.

image::images/fr/dialog_bom.png[alt="Fenêtre du générateur de BOM", scaledwidth="60%"]

Additional scripts can be added to the list of BOM generator scripts by clicking the
image:images/icons/small_plus_16.png[Plus icon] button. Scripts can be removed by clicking
the image:images/icons/small_trash_16.png[Delete icon] button. The
image:images/icons/small_edit_16.png[Edit icon] 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]]
==== 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.

[options="header", width="100%", cols="10%,30%,60%"]
|===
|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.
|===

[[intermediate-netlist-file]]
==== Fichier intermédiaire de Netliste

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 <<intermediate-
netlist-structure,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.

[[intermediate-netlist-structure]]
==== Structure du fichier de netliste intermédiaire

L'exemple qui suit donne une idée du format du fichier de netliste intermédiaire.

```xml
<?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"/>
      <node ref="U2" pin="3"/>
    </net>
    <net code="5" name="/SIG_OUT">
      <node ref="P1" pin="2"/>
      <node ref="U2" pin="5"/>
      <node ref="U2" pin="2"/>
    </net>
    <net code="6" name="/CLOCK_IN">
      <node ref="R1" pin="2"/>
      <node ref="C1" pin="1"/>
```

```
<node ref="U1" pin="1"/>
      <node ref="P1" pin="3"/>
    </net>
  </nets>
</export>
```

Structure générale

Le fichier de netliste intermédiaire contient cinq sections.

- La section Entête.
- The components section.
- La section Composants en librairie.
- La section Librairies.
- La section Équipotentielles

The file content has the delimiter `<export>`

```
 ...  ```" data-lang="xml  ...  ```">[[the-header-section]]
===== Section Entête (Header)

The header has the delimiter `<design>`

```xml <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> ```

Cette section peut être considérée comme une section de commentaires.

[[the-components-section]]
===== Section Composants

The component section has the delimiter `<components>`

```xml <components> <comp ref="P1"> <value>CONN_4</value> <libsource lib="conn"
part="CONN_4"/> <sheetpath names="/" tstamps="/"/> <tstamps>4C6E2141</tstamps> </comp>
</components> ``` This section contains the list of components in your schematic. Each
component is described like this:

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

[width="100%", cols="37%,63%"]
|=========================================================================
|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]]
====== 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.

Un horodatage (timestamp) est un identifiant unique pour chaque composant, ou chaque
feuille d'un projet schématique. Cependant, dans des hiérarchies complexes, la même feuille
étant utilisée plus d'une fois, cette feuille contiendra des composants avec le même
horodatage.

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.

[[the-libparts-section]]
===== Section Composants en librairie (libparts)

The libparts section has the delimiter `<libparts>`, and the content of this section is
defined in the schematic libraries.

```xml
```

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

Section Librairies

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>
```

Section Équipotentielles (nets)

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>
```

Une entrée net peut contenir :

```
<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 -->
```

La ligne de commande utilisée pour effectuer cette conversion :

```
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 -->
```

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
<?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  "
"> <!--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>

<!--
```
```

({ 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))) *

```
:experimental:

[[eeschema-actions-reference]]
== Actions reference
Below is a list of every available *action* in the KiCad Schematic Editor: a command that
can be assigned to a hotkey.

////
Note to translators: you do not need to translate this table by hand.

It is generated from KiCad using the Dump Hotkeys button that is shown in the hotkeys
editor if you
add the line `HotkeysDumper=1` to your advanced config file (`kicad_advanced` file in the
config
directory)
////

=== Schematic Editor

// NOTE: this text between the section header and the table is *required* or
// asciidoctor-web-pdf will not insert page breaks in the table correctly and
// the PDF will be truncated.
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.

[width="100%", options="header", cols="20%,15%,65%"]
|===
| Action | Default Hotkey | Description
| Align Elements to Grid
  |
  |
| Annotate Schematic...
  |
  | Fill in schematic symbol reference designators
| Assign Footprints...
  |
  | Run footprint assignment tool
| Clear Net Highlighting
  | kbd:[~]
  | Clear any existing net highlighting
| Export Drawing to Clipboard
  |
  | Export drawing of current sheet to clipboard
| Edit Library Symbol...
  | kbd:[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
  | kbd:[Ctrl+E]
  | Open the selected symbol in the Symbol Editor
| Highlight on PCB
  |
```