Schematic Editor

The KiCad Team

Table of Contents

Reference manual

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

Contributors

Jean-Pierre Charras, Fabrizio Tappero, Graham Keeth

Feedback

Please direct any bug reports, suggestions or new versions to here:

- About KiCad documentation: https://gitlab.com/kicad/services/kicad-doc/issues
- About KiCad software: https://gitlab.com/kicad/code/kicad/issues

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:

- Electrical rules check (ERC) for the automatic control of incorrect and missing connections
- Export of plot files in many formats (Postscript, PDF, HPGL, and SVG)
- Bill of Materials generation (via Python or XSLT scripts, which allow many flexible formats).

Technical overview

The Schematic Editor is limited only by the available memory. There is thus no real limitation to the number of components, component pins, connections or sheets. In the case of multi-sheet schematics, the representation is hierarchical.

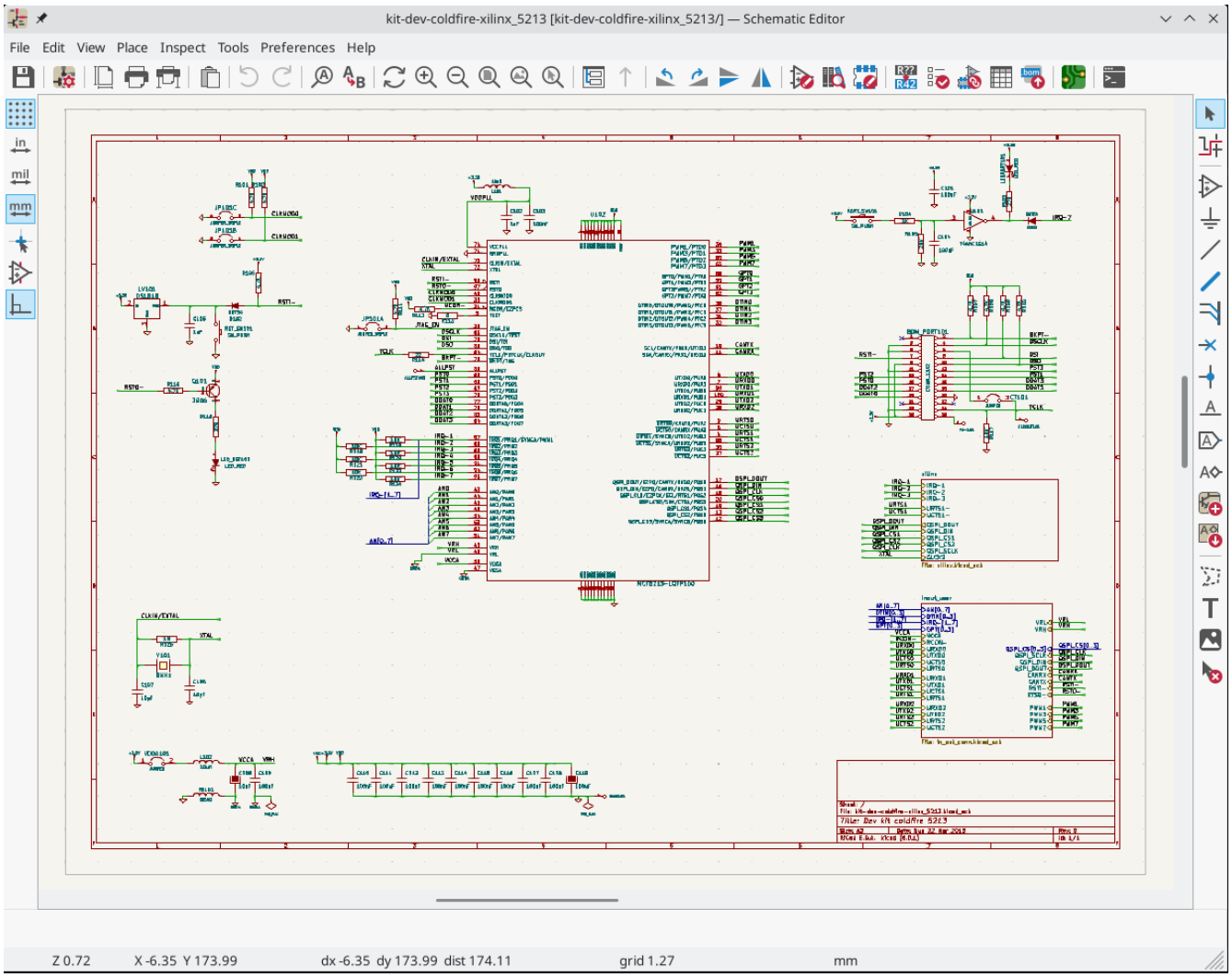
The Schematic Editor can use multi-sheet schematics in a few ways:

- Simple hierarchies (each schematic is used only once).
- Complex hierarchies (some schematics are used more than once with multiple instances).
- Flat hierarchies (schematics are not explicitly connected in a master diagram).

Generic Schematic Editor commands

Commands can be executed by:

- Clicking on the menu bar (top of screen).
- Clicking on the icons on top of the screen (general commands).
- Clicking on the icons on the right side of the screen (particular commands or "tools").
- Clicking on the icons on the left side of the screen (display options).
- Pressing the mouse buttons (important complementary commands). In particular a right click opens a contextual menu for the element under the cursor (Zoom, grid and editing of the elements).
- Function keys (`F1`, `F2`, `F3`, `F4`, `Insert` and `Space`). Specifically: `Escape` cancels the command in progress. `Insert` allows the duplication of the last element created.
- Pressing hotkeys. For a list of hotkeys, see the **Help→List Hotkeys** menu entry or press `Ctrl`+`F1`. Many hotkeys select a tool but do not perform the tool's action until the canvas is clicked. This behavior can be changed by unchecking **First hotkey selects tool** in the **Common** Preferences pane. With this option unchecked, pressing a hotkey will select the tool and immediately perform the tool's action at the current cursor location.



Mouse commands

Basic commands

Left button

- Single click: Selects the item under the cursor and displays the item's characteristics in the status bar.
- Double click: edits the item if it is editable.
- Long click (click and hold): opens a pop-up menu to clarify the selection.

Right button

- Opens a pop-up menu. If an item is selected, the items in the menu are related to the selected item. If an item is under the cursor when the right mouse button is clicked, the item is selected.

Selection operations

Schematic editor items can be selected by clicking on them. Multiple items can be selected at once. Add items to the selection with `Shift` + click, and remove items from the selection with `Ctrl`+`Shift` + click.

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

left mouse button	Select item.
`Shift` + left mouse button	Add item to selection.
`Ctrl`+`Shift` + left mouse button	Remove item from selection.
long click	Clarify selection from a pop-up menu.
`Ctrl` + left mouse button	Highlight net.

Items can also be selected by drawing a box around them using the left mouse button.

Dragging from left to right includes all items fully enclosed by the box. Dragging from right to left includes all items touched by the box, even if they are not fully enclosed.

The `Shift` and `Ctrl`+`Shift` modifiers also work with drag selections to add and remove items from the selection, respectively.

Hotkeys

- The `Ctrl`+`F1` displays the current hotkey list.
- All hotkeys can be redefined using the hotkey editor (**Preferences → Preferences... →** Hotkeys).

The default hotkey list is below. Many additional actions do not have hotkeys by default, but hotkeys can be assigned to them with the hotkey editor.

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`.

Action	Default Hotkey	Description
Click	`Return`	Performs left mouse button click
Double-click	`End`	Performs left mouse button double-click
Cursor Down	`Down`	
Cursor Down Fast	`Ctrl` + `Down`	
Cursor Left	`Left`	
Cursor Left Fast	`Ctrl` + `Left`	
Cursor Right	`Right`	
Cursor Right Fast	`Ctrl` + `Right`	
Cursor Up	`Up`	
Cursor Up Fast	`Ctrl` + `Up`	
Switch to Fast Grid 1	`Alt` + `1`	
Switch to Fast Grid 2	`Alt` + `2`	
Switch to Next Grid	`N`	
Switch to Previous Grid	`Shift` + `N`	
Reset Grid Origin	`Z`	
Grid Origin	`S`	Set the grid origin point
New…	`Ctrl` + `N`	Create a new document in the editor
Open…	`Ctrl` + `O`	Open existing document
Pan Down	`Shift` + `Down`	
Pan Left	`Shift` + `Left`	
Pan Right	`Shift` + `Right`	
Pan Up	`Shift` + `Up`	
Print…	`Ctrl` + `P`	Print

Action	Default Hotkey	Description
Reset Local Coordinates	Space	
Save	Ctrl + S	Save changes
Save As…	Ctrl + Shift + S	Save current document to another location
Always Show Cursor	Ctrl + Shift + X	Display crosshairs even in selection tool
Switch units	Ctrl + U	Switch between imperial and metric units
Update PCB from Schematic…	F8	Update PCB with changes made to schematic
Center	F4	Center
Zoom to Objects	Ctrl + Home	Zoom to Objects
Zoom to Fit	Home	Zoom to Fit
Zoom In at Cursor	F1	Zoom In at Cursor
Zoom Out at Cursor	F2	Zoom Out at Cursor
Refresh	F5	Refresh
Zoom to Selection	Ctrl + F5	Zoom to Selection
Change Edit Method	Ctrl + Space	Change edit method constraints
Copy	Ctrl + C	Copy selected item(s) to clipboard
Cut	Ctrl + X	Cut selected item(s) to clipboard
Delete	Del	Deletes selected item(s)
Duplicate	Ctrl + D	Duplicates the selected item(s)
Find	Ctrl + F	Find text
Find and Replace	Ctrl + Alt + F	Find and replace text
Find Next	F3	Find next match
Find Next Marker	Shift + F3	
Paste	Ctrl + V	Paste item(s) from clipboard

Action	Default Hotkey	Description
List Hotkeys…	Ctrl + F1	Displays current hotkeys table and corresponding commands
Preferences…	Ctrl + ,	Show preferences for all open tools
Clear Net Highlighting	~	Clear any existing net highlighting
Edit Library Symbol…	Ctrl + Shift + E	Open the library symbol in the Symbol Editor
Edit with Symbol Editor	Ctrl + E	Open the selected symbol in the Symbol Editor
Highlight Net	`	Highlight net under cursor
Show Datasheet	D	Opens the datasheet in a browser
Add Sheet	S	Add a hierarchical sheet
Add Wire to Bus Entry	Z	Add a wire entry to a bus
Add Global Label	Ctrl + L	Add a global label
Add Hierarchical Label	H	Add a hierarchical label
Add Junction	J	Add a junction
Add Label	L	Add a net label
Add No Connect Flag	Q	Add a no-connection flag
Add Power	P	Add a power port
Add Text	T	Add text
Add Symbol	A	Add a symbol
Add Bus	B	Add a bus
Add Lines	I	Add connected graphic lines
Add Wire	W	Add a wire
Finish Wire or Bus	K	Complete drawing at current segment
Unfold from Bus	C	Break a wire out of a bus
Autoplace Fields	O	Runs the automatic placement algorithm on the symbol or sheet's fields

Action	Default Hotkey	Description
Mirror Vertically	Y	Flips selected item(s) from top to bottom
Properties…	E	Displays item properties dialog
Repeat Last Item	Ins	Duplicates the last drawn item
Rotate Counterclockwise	R	Rotates selected item(s) counter-clockwise
Drag	G	Drags the selected item(s)
Move	M	Moves the selected item(s)
Select Connection	Alt + 4	Select a complete connection
Select Node	Alt + 3	Select a connection item under the cursor
Leave Sheet	Alt + Back	Display the parent sheet in the schematic editor

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`

It is possible to import hotkey settings from a `user.hotkeys` file using menu **Preferences→Preferences…→Hotkeys→Import Hotkeys…**.

Grid

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 preferred grid to place symbols and wires in a schematic, and to place 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 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 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.

Zoom selection

To change the zoom level:

- Right click to open the Pop-up menu and select the desired zoom.
- Or use hotkeys:
  - `F1`: Zoom in
  - `F2`: Zoom out
  - `F4`: Center the view around the cursor pointer position
  - `Home`: Zoom and center the view to fit the entire schematic sheet
  - `Ctrl` + `Home`: Zoom and center the view to fit all of the objects in the schematic
  - `Ctrl` + `F5`: Activate the Zoom to Selection tool
- Window Zoom:
  - Mouse wheel: Zoom in/out
  - Shift+Mouse wheel: Pan up/down
  - Ctrl+Mouse wheel: Pan left/right

Mouse scroll gestures are configurable in the **Mouse and Touchpad** page of the **Preferences** dialog.

Displaying cursor coordinates

The display units are in inches, mils, or millimeters.

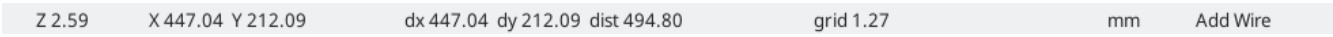
The following information is displayed at the bottom right hand side of the window:

- The zoom factor
- The absolute position of the cursor

The relative position of the cursor

- The grid size
- The active unit system
- The active tool

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



Top menu bar

The top menu bar allows the opening and saving of schematics, program configuration and viewing the documentation.

Upper toolbar

This toolbar gives access to the main functions of the Schematic Editor.

If the Schematic Editor is run in standalone mode, this is the available tool set:

Note that when KiCad runs in project mode, the first two icons are not available as they work with individual files.

	Create a new schematic (only in standalone mode).
	Open a schematic (only in standalone mode).
	Save complete schematic project.
	Set the schematic-specific options.
	Select the sheet size and edit the title block.
	Open print dialog.
	Open plot dialog.
	Paste a copied/cut item or block to the current sheet.
	Undo: Revert the last change.
	Redo: Revert the last undo operation.
	Show the dialog to search symbols and texts in the schematic.

	Show the dialog to search and replace texts in the schematic.
	Refresh screen.
	Zoom in.
	Zoom out.
	Zoom to fit the entire schematic sheet.
	Zoom to fit all objects in the schematic.
	Zoom to fit selected items.
	View and navigate the hierarchy tree.
	Leave the current sheet and go up in the hierarchy.
	Rotate selected items counter-clockwise.
	Rotate selected items clockwise.
	Mirror selected items vertically.
	Mirror selected items horizontally.
	Call the symbol library editor to view and modify libraries and symbols.
	Browse symbol libraries.
	Open the footprint library editor to view and modify libraries and footprints.
	Annotate symbols.
	Electrical Rules Checker (ERC), automatically validate electrical connections.
	Open the footprint assignment tool to assign footprints to symbols.
	Bulk edit symbol fields in a spreadsheet interface.
	Generate the Bill of Materials (BOM).
	Open the PCB editor.
	Open the Python scripting console.

Right toolbar icons

This toolbar contains tools to:

- Place symbols, wires, buses, junctions, labels, text, etc.

- Create hierarchical subsheets and connection symbols.

	Cancel the active command or tool and go into selection mode.
	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.
	Display the symbol selector dialog to select a new symbol to be placed.
	Display the power symbol selector dialog to select a power symbol to be placed.
	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. It is done to notify the Electrical Rules Checker that lack of connection for a particular pin is intentional and should not be reported.
	Place a junction. This connects two crossing wires or a wire and a pin, when it can be ambiguous (i.e. if a wire end or a pin is not directly connected to another wire end).
	Place a local label. Local label connects items located **in the same sheet**. For connections between two different sheets, you have to 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 parent sheet that contains it.
	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 a line. These are only graphical and do not connect anything.
	Place a text comment.
	Place a bitmap image.
	Delete clicked items.

Left toolbar icons

This toolbar manages the display options:

	Toggle grid visibility.
	Switch units to inches.
	Switch units to mils (0.001 inches).
	Switch units to millimeters.
	Choose the cursor shape (full screen/small).
	Toggle visibility of "invisible" pins.
	Toggle free angle/90 degrees wires and buses placement.

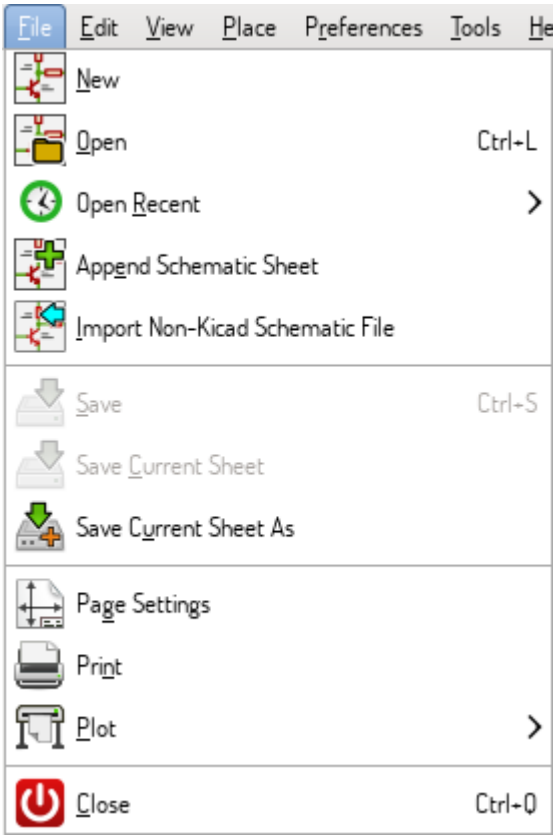
Pop-up menus and quick editing

A right-click opens a contextual menu for the selected element. This contains:

- Zoom factor.
- Grid adjustment.
- Copy/Paste/Delete commands.
- Add Wire/Bus.
- Commonly edited parameters of the selected element.

Main top menu

File menu

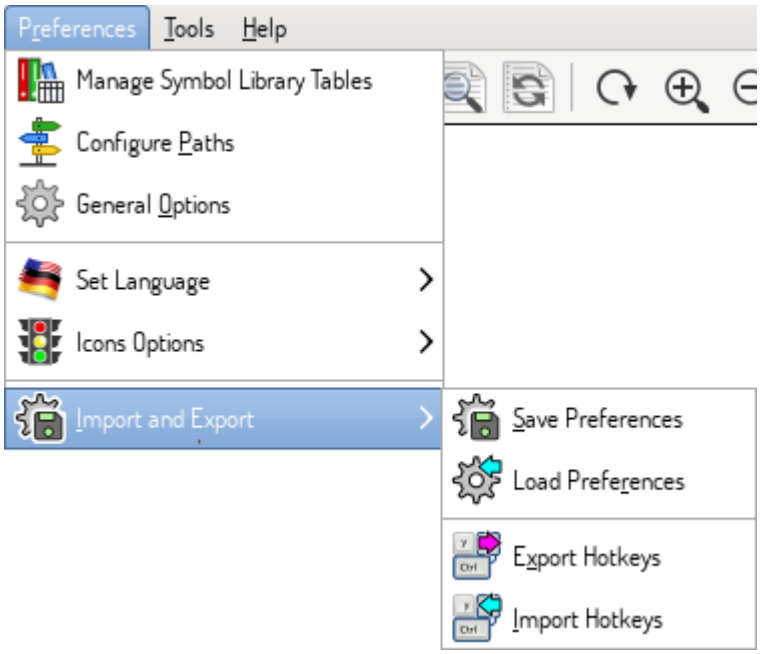


New	**Close current schematic and start a new one (only in standalone mode).**
Open	Load a schematic project (only in standalone mode).
Open Recent	Open a schematic project from the list of recently opened files (only in standalone mode).
Save	Save current sheet and all its subsheets.
Save As…	Save the current sheet under a new name (only in standalone mode).
Save Current Sheet Copy As…	Save a copy of the current sheet under a new name (only in project mode).
Insert Schematic Sheet Content…	Insert the contents of another schematic sheet into the current sheet (only in standalone mode).
Import	Import a non-KiCad schematic or a footprint assignment file.
Export	Export a netlist or a drawing of the schematic to the clipboard.
Schematic Setup…	Set up schematic formatting, electrical rules, net classes, and text variables.
Page Settings…	Configure page dimensions and title block.
Print	Print schematic project (See also chapter Plot and Print).
Plot	Export to PDF, PostScript, HPGL or SVG format (See chapter Plot and Print).
Quit	Terminate the application.

Schematic Setup

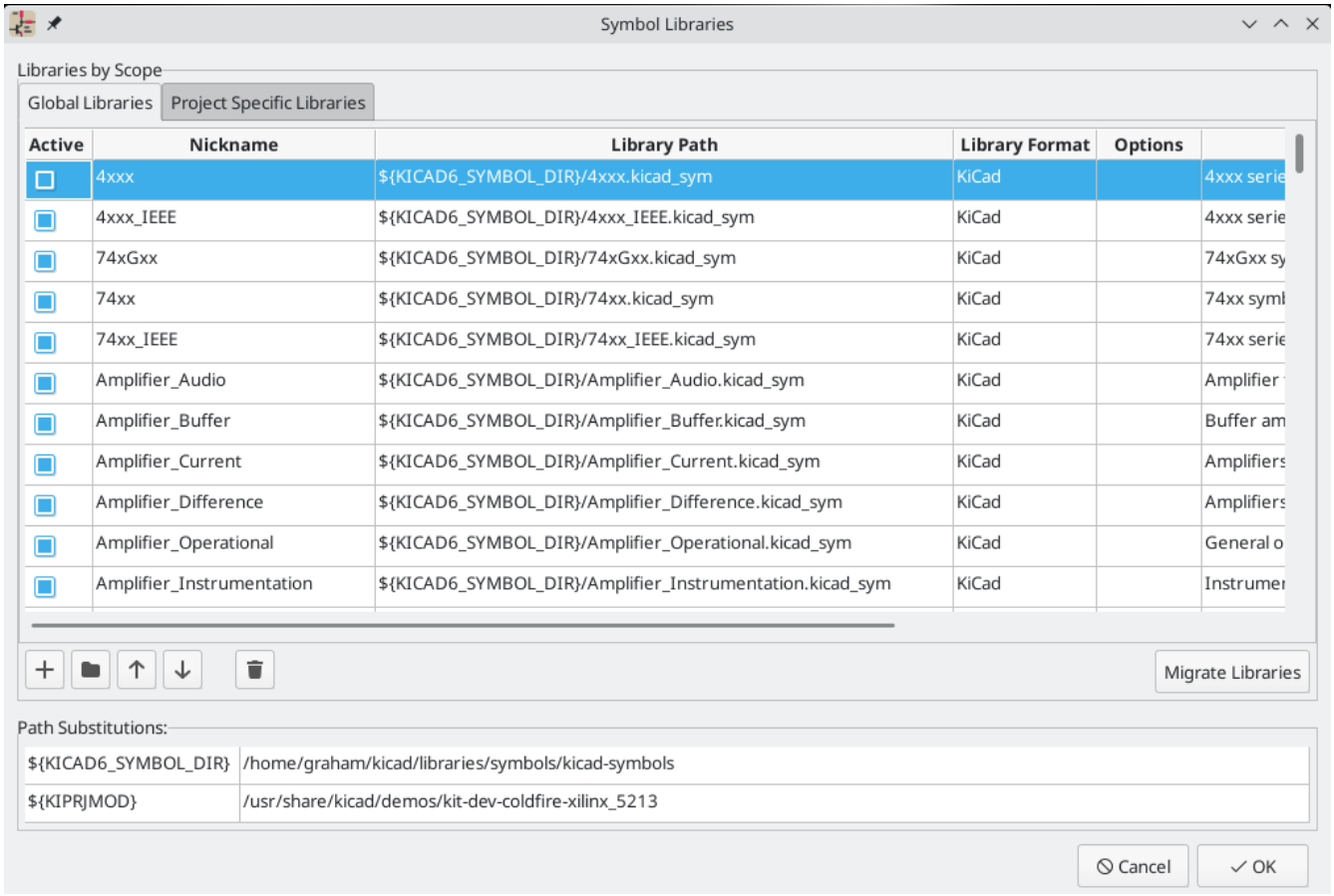
The Schematic Setup window is used to set schematic options that are specific to the currently active schematic. For example, the Schematic Setup window contains formatting options, electrical rule configuration, netclass setup, and schematic text variable setup.

Preferences menu



Configure Paths...	Set the default search paths.
Manage Symbol Library Tables...	Add/remove symbol libraries.
Preferences...	Preferences (units, grid size, field names, etc.).
Set Language	Select interface language.

Manage Symbol Library Tables



KiCad uses two library tables to store the list of available symbol libraries, which differ by the scope:

Global Libraries

Libraries listed in the Global Library table are available to every project. They are saved in the `sym-lib-table` in the KiCad configuration directory, which is system-dependent:

- Windows: `%APPDATA%\kicad\6.0\sym-lib-table`
- Linux: `~/.config/kicad/6.0/sym-lib-table`
- macOS: `~/Library/Preferences/kicad/6.0/sym-lib-table`

Project Specific Libraries

Libraries listed in Project Specific Libraries table are available to the currently opened project. They are saved in a `sym-lib-table` file in the project directory.

Both library tables are visible by clicking on **Global Libraries** or **Project Specific Libraries** tab in the Manage Library Tables window.

Add a new library

Add a library either by clicking the 🗀 button and selecting a file or clicking the + button and typing a path to a library file. The selected library will be added to the currently opened library table (Global/Project Specific).

Remove a library

Remove a library by selecting one or more libraries and clicking the 🗑 button.

Library properties

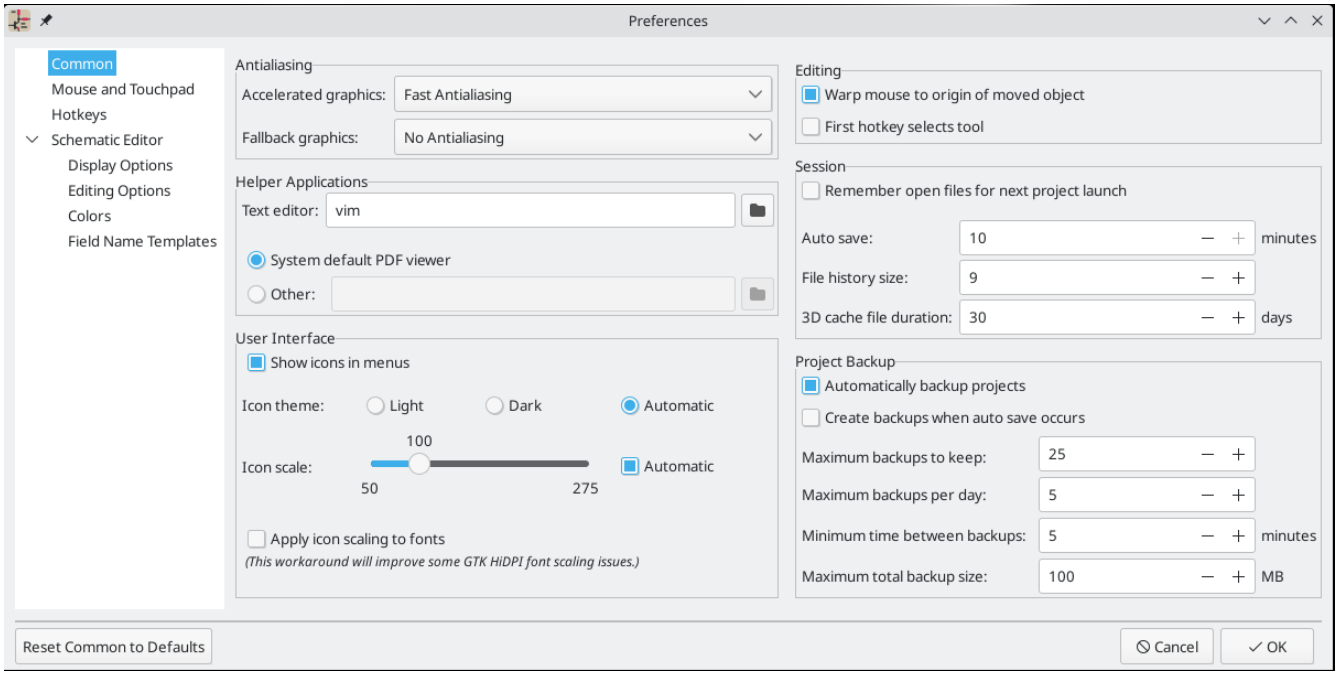
Each row in the table stores several fields describing a library:

Active	Enables/disables the library. It is useful to temporarily reduce the loaded library set.
Nickname	Nickname is a short, unique identifier used for assigning symbols to components. Symbols are represented by '<Library Nickname>:<Symbol Name>' strings.
Library Path	Path points to the library location.
Plugin Type	Determines the library file format. KiCad 6.0 libraries use the "KiCad" format, while KiCad 5.x libraries use the "Legacy" format. Legacy libraries are read-only.
Options	Stores library specific options, if used by plugin.
Description	Briefly characterizes the library contents.

Preferences

Common Preferences

> **NOTE** TODO: write this section

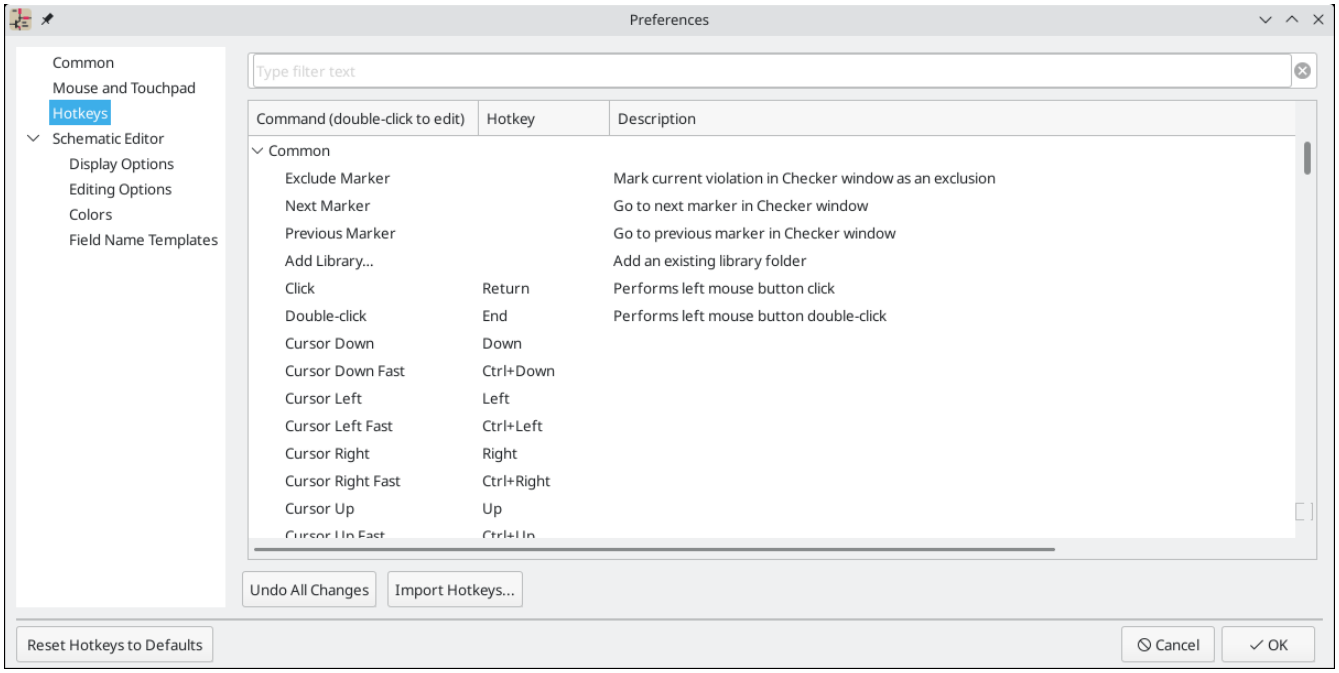


Mouse and Touchpad

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 `Ctrl`. Otherwise scroll wheels zoom in/out and `Ctrl`/`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.

Hotkeys

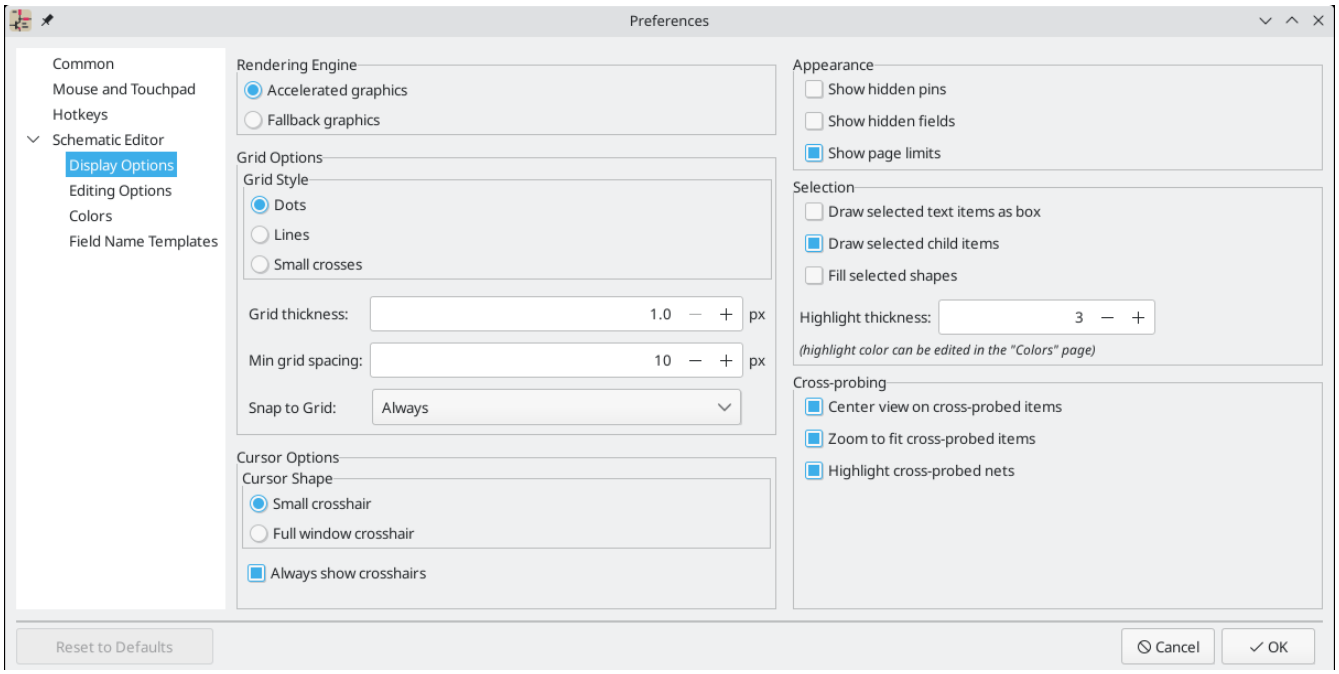
Redefine hotkeys.



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

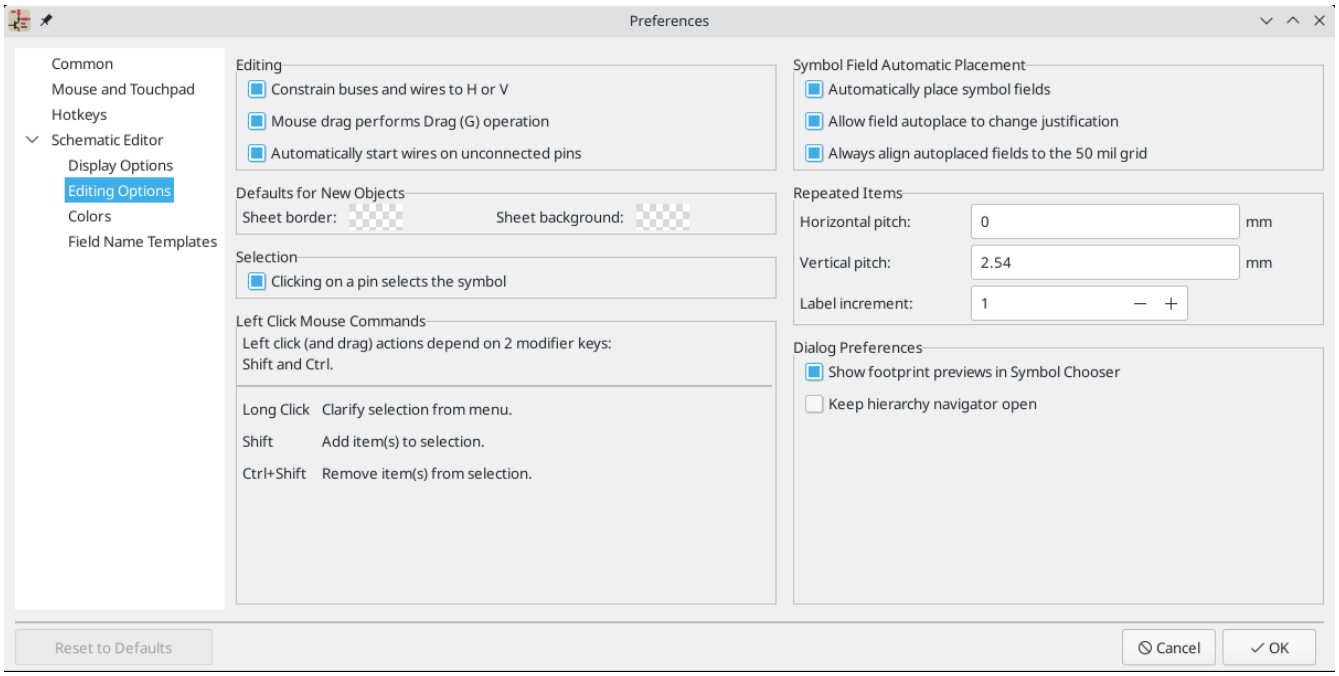
Edit	Define a new hotkey for the action (same as double click).
Undo Changes	Reverts the recent hotkey changes for the action.
Clear Assigned Hotkey	
Restore Default	Sets the action hotkey to its default value.

Display Options



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.

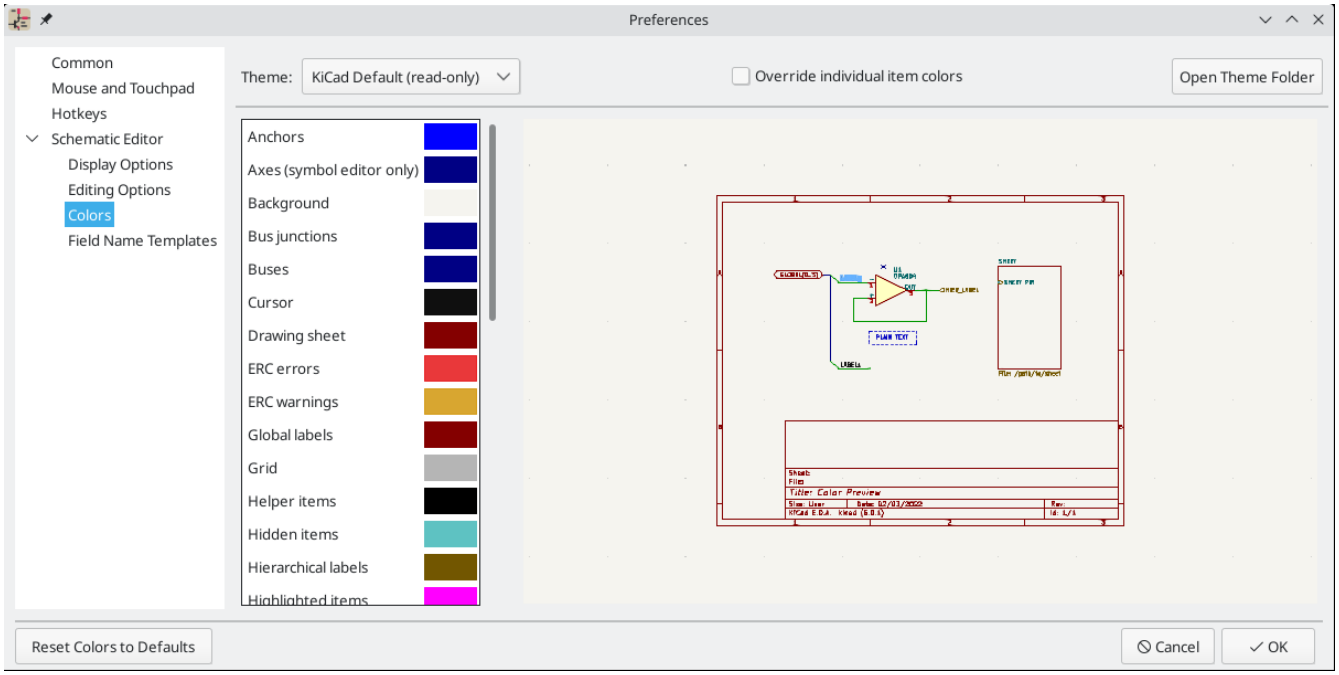
Editing Options



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 `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.

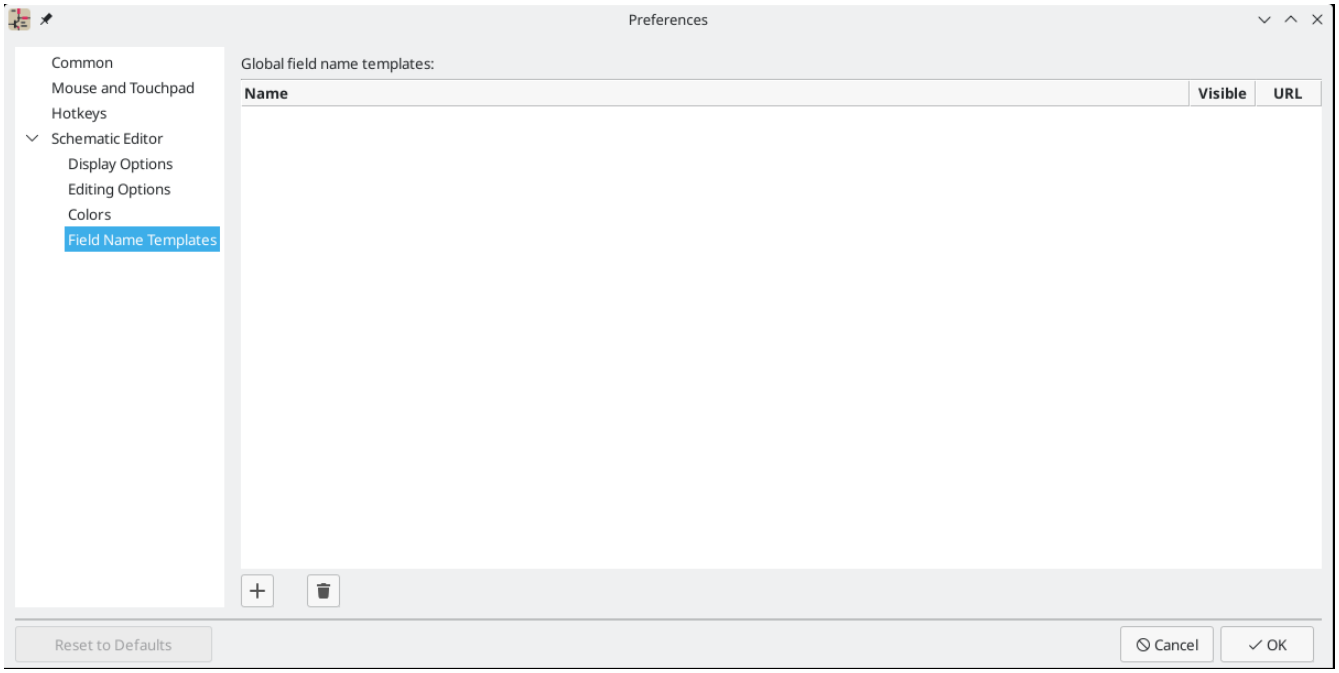
Colors

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



Default Fields

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



Help menu

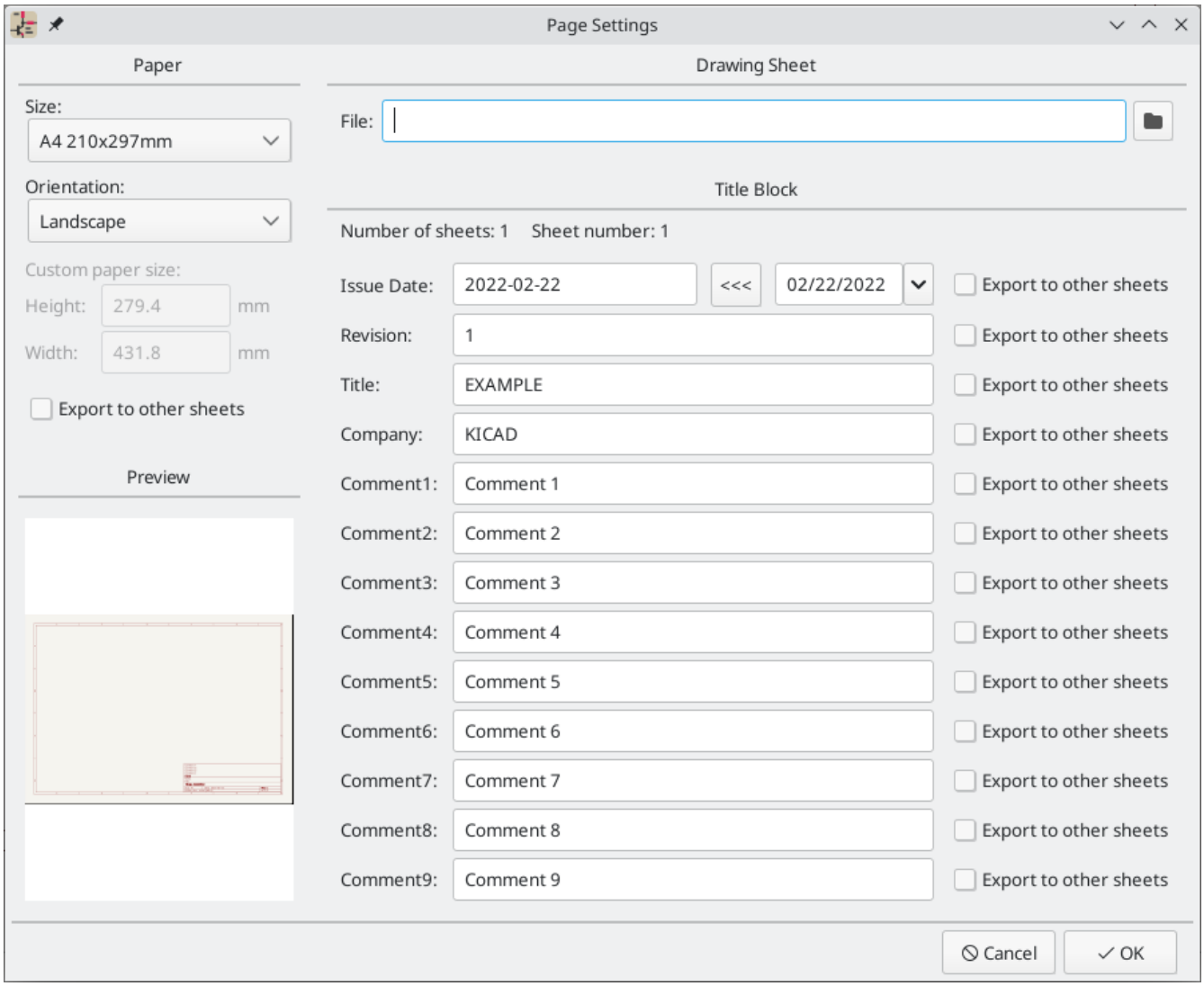
Access to on-line help (this document) for an extensive tutorial about KiCad.

Use the **Report a Bug** item to report a bug online. Full KiCad version and user system information is available via the **Copy Version Info** button in the **About KiCad** window.

General Top Toolbar

Sheet management

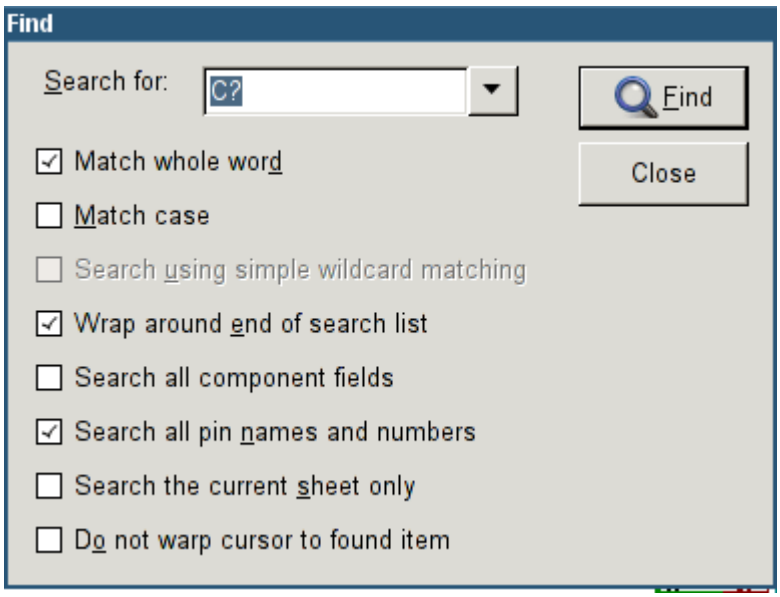
The Sheet Settings icon ( ) allows you to define the sheet size and the contents of the title block.



Sheet numbering is automatically updated. You can set the date to today by pressing the left arrow button by "Issue Date", but it will not be automatically changed.

Search tool

The Find icon ( ) can be used to access the search tool.



You can search for a reference, a value or a text string in the current sheet or in the whole hierarchy. Once found, the cursor will be positioned on the found element in the relevant sub-sheet.

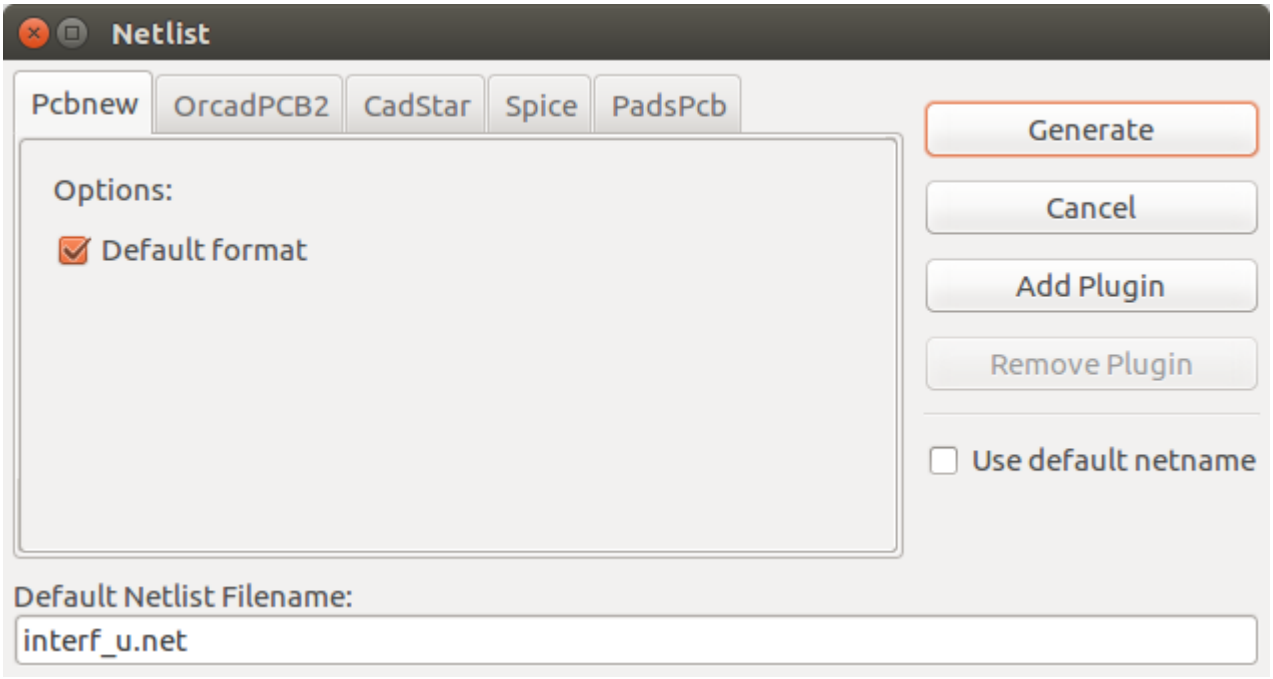
Netlist tool

The Netlist icon ( ) opens the netlist generation tool.

The tool creates a file which describe all connections in the entire hierarchy.

In a multisheet hierarchy, any local label is visible only inside the sheet to which it belongs. For example: the label LABEL1 of sheet 3 is different from the label LABEL1 of sheet 5 (if no connection has been intentionally introduced to connect them). This is due to the fact that the sheet name path is internally associated with the local label.

NOTE	Even though there is no text length limit for labels in KiCad, please take into account that other programs reading the generated netlist may have such constraints.
NOTE	Avoid spaces in labels, because they will appear as separated words in the generated file. It is not a limitation of KiCad, but of many netlist formats, which often assume that a label has no spaces.



Option:

Default Format	Check to select Pcbnew as the default format.

Other formats can also be generated:

- Orcad PCB2
- CadStar
- Spice (simulators)

External plugins can be added to extend the netlist formats list (PadsPcb Plugin was added in the picture above).

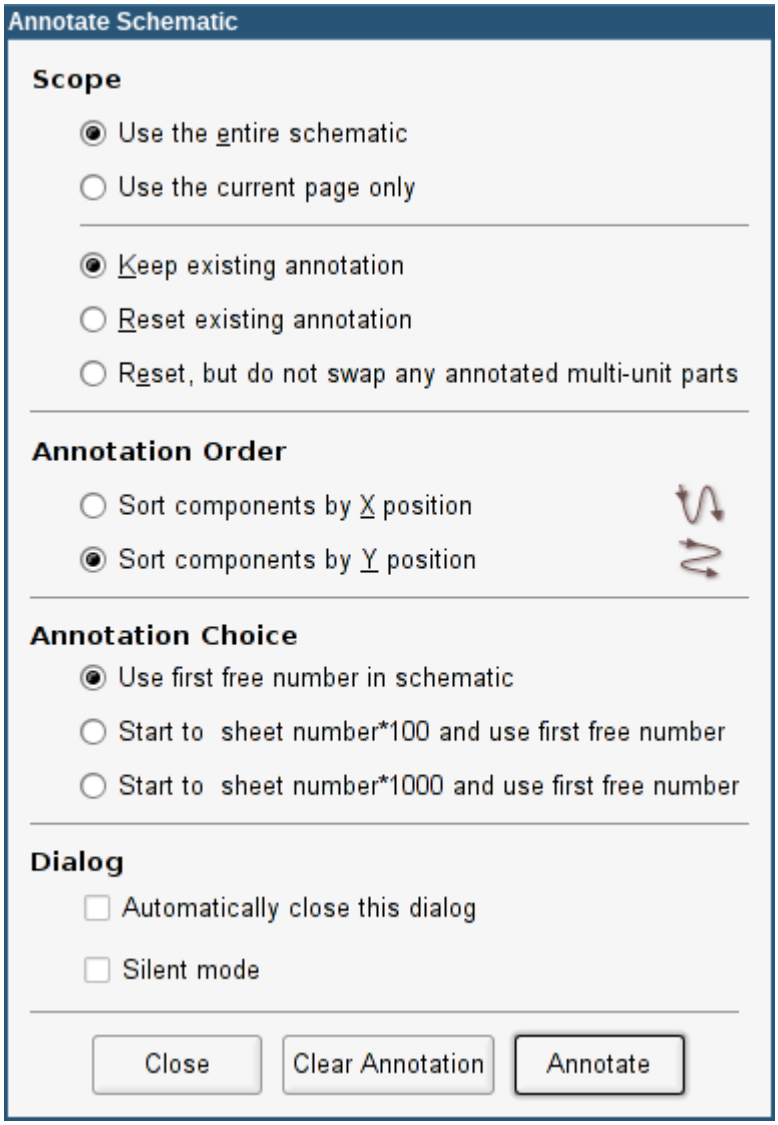
There is more information about creating netlists in Create a Netlist chapter.

Annotation tool

The icon launches the annotation tool. This tool assigns references to components.

For multi-part components (such as 7400 TTL which contains 4 gates), a multi-part suffix is also allocated (thus a 7400 TTL designated U3 will be divided into U3A, U3B, U3C and U3D).

You can unconditionally annotate all the components or only the new components, i.e. those which were not previously annotated.



Scope

Use the entire schematic	**All sheets are re-annotated (default).**
Use the current page only	Only the current sheet is re-annotated (this option is to be used only in special cases, for example to evaluate the amount of resistors in the current sheet.).
Keep existing annotation	Conditional annotation, only the new components will be re-annotated (default).
Reset existing annotation	Unconditional annotation, all the components will be re-annotated (this option is to be used when there are duplicated references).
Reset, but do not swap any annotated multi-unit parts	Keeps all groups of multiple units (e.g. U2A, U2B) together when reannotating.

Annotation Order

Selects the order in which components will be numbered (either horizontally or vertically).

Annotation Choice

Selects the assigned reference format.

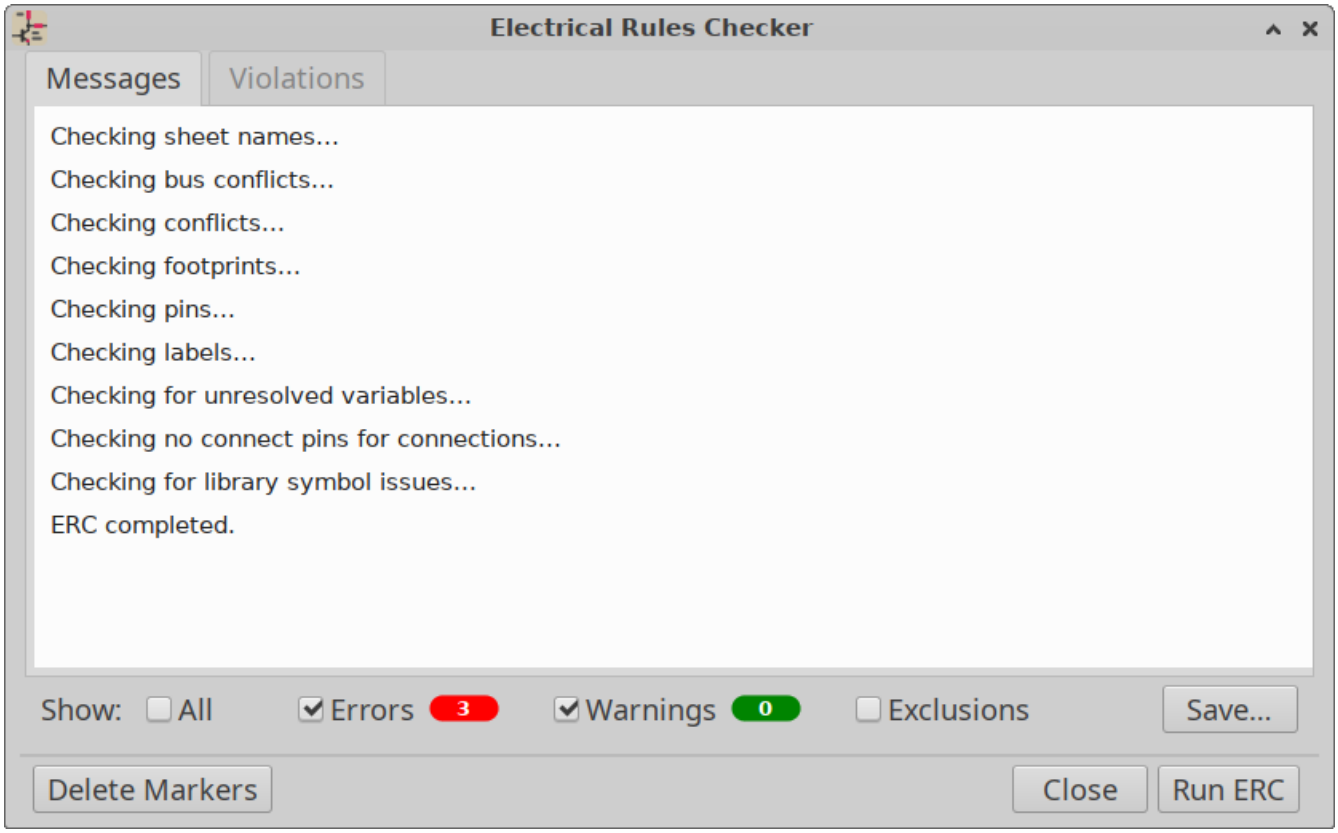
Electrical Rules Check tool

The icon launches the electrical rules check (ERC) tool.

This tool performs a design verification and is able to detect forgotten connections, and inconsistencies.

Once you have run the ERC, KiCad places markers to highlight problems. The error description is displayed after left clicking on the marker. An error report file can also be generated.

Main ERC dialog



Errors are displayed in the Electrical Rules Checker dialog:

- Total count of errors and warnings.
- Errors count.
- Warnings count.

Option:

Create ERC file report	Check this option to generate an ERC report file.

Commands:

Delete Markers	Remove all ERC error/warnings markers.
Run	Start an Electrical Rules Check.
Close	Close the dialog.

- Clicking on an error message jumps to the corresponding marker in the schematic.

ERC options dialog

This tab allows you to define the connectivity rules between pins; you can choose between 3 options for each case:

- No error
- Warning

Error

Each square of the matrix can be modified by clicking on it.

Option:

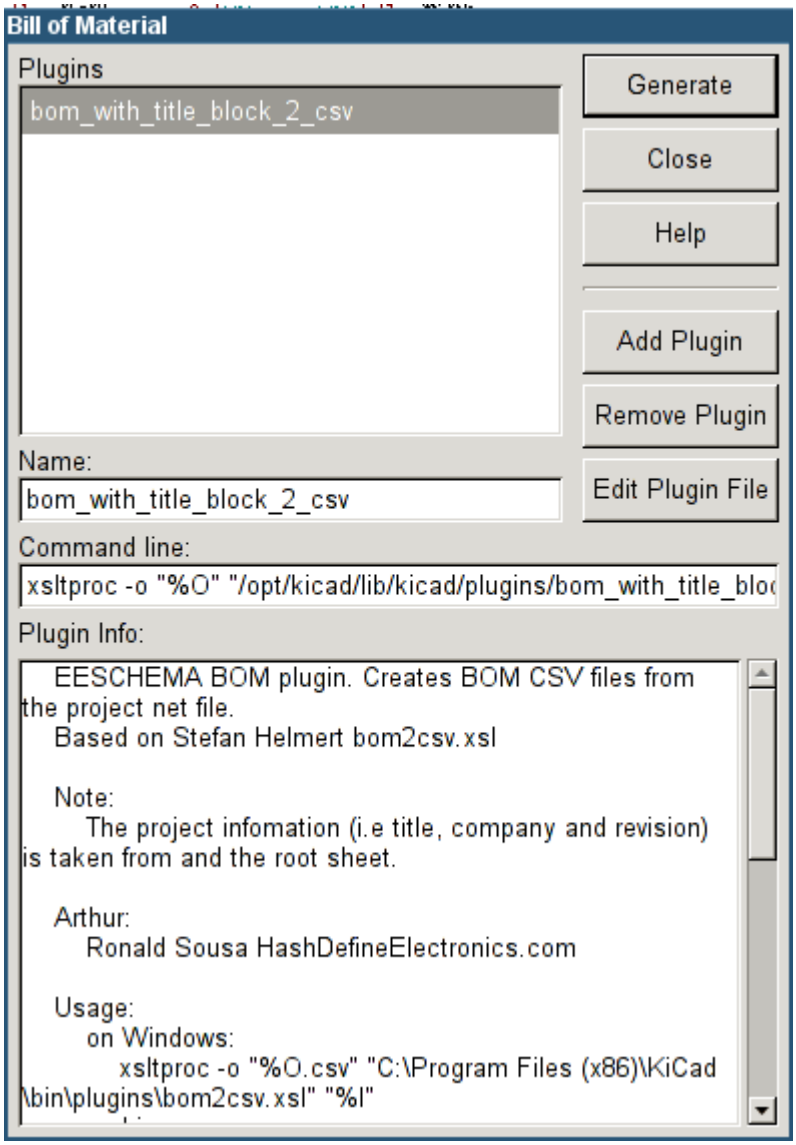
Test similar labels	Report labels that differ only by letter case (e.g. label/Label/LaBeL). Net names are case-sensitive therefore such labels are treated as separate nets.
Test unique global labels	Report global lables that occur only once for a particular net. Normally it is required to have at least two make a connection.

Commands:

Initialize to Default	Restores the original settings.

Bill of Material tool

The icon  launches the bill of materials (BOM) generator. This tool generates a file listing the components and/or hierarchical connections (global labels).

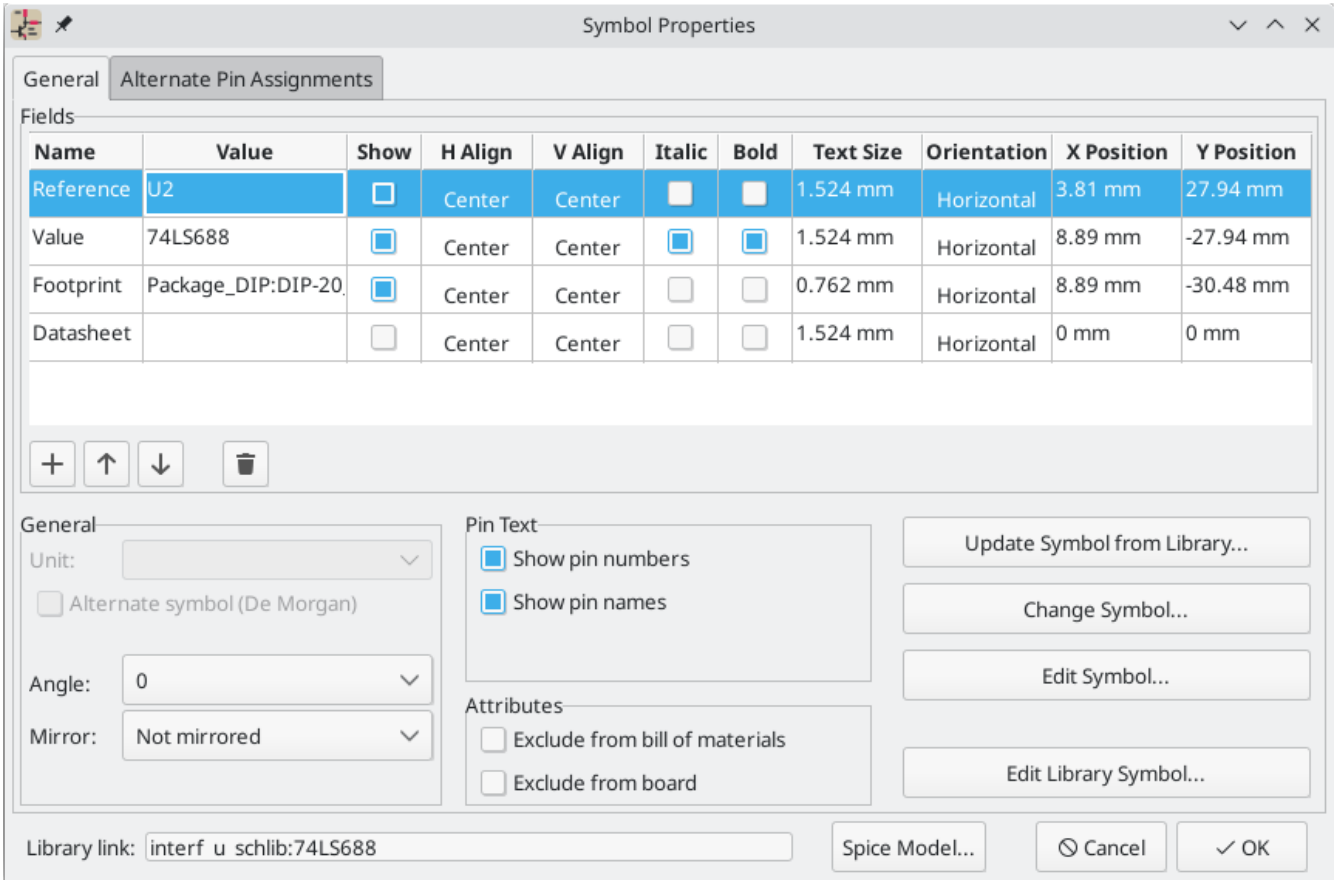


The Schematic Editor's BOM generator makes use of external plugins, either as XSLT or Python scripts. There are a few examples installed inside the KiCad program files directory.

A useful set of component properties to use for a BOM are:

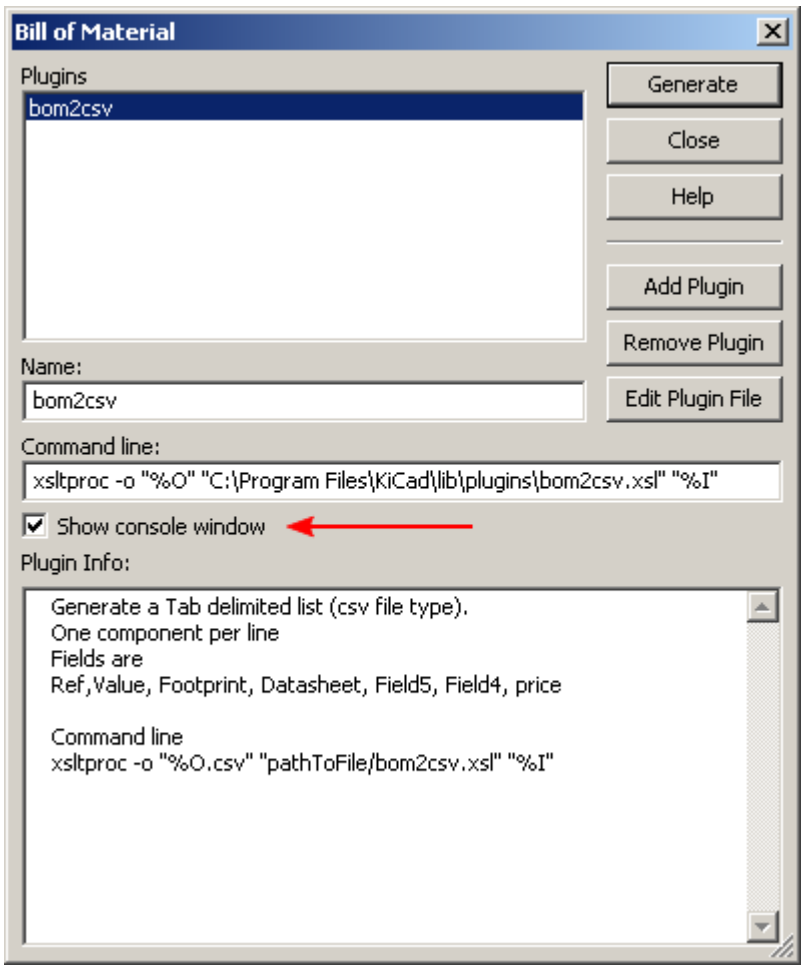
- Value - unique name for each part used.
- Footprint - either manually entered or back-annotated (see below).
- Field1 - Manufacturer's name.
- Field2 - Manufacturer's Part Number.
- Field3 - Distributor's Part Number.

For example:



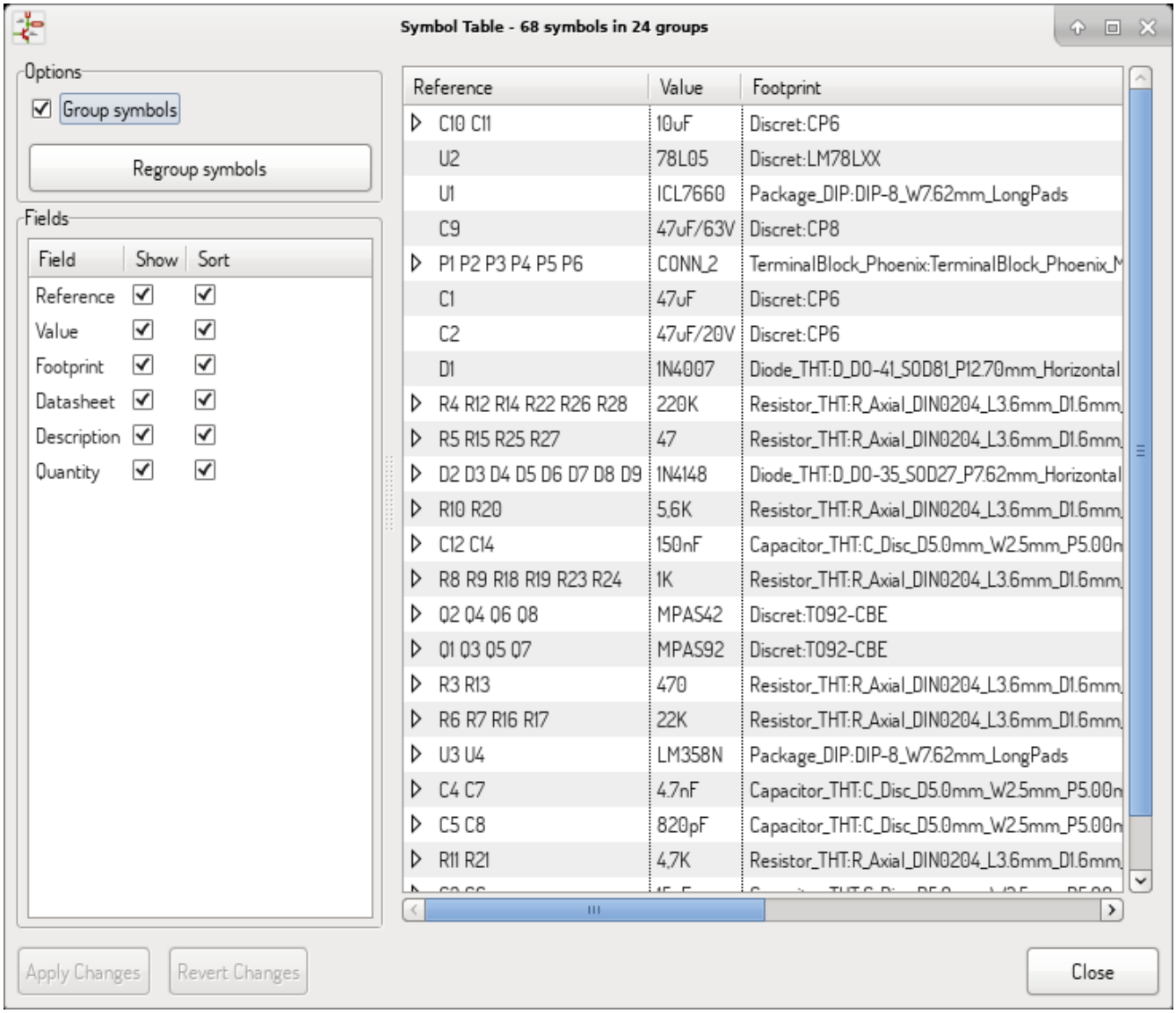
On **MS Windows**, BOM generator dialog has a special option (pointed by red arrow) that controls visibility of external plugin window.

By default, BOM generator command is executed console window hidden and output is redirected to *Plugin info* field. Set this option to show the window of the running command. It may be necessary if plugin has provides a graphical user interface.



Edit Fields tool

The icon  opens a spreadsheet to view and modify field values for all symbols.

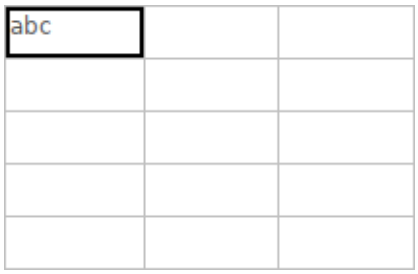
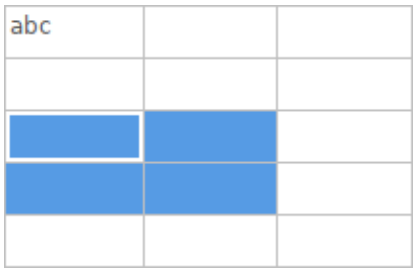
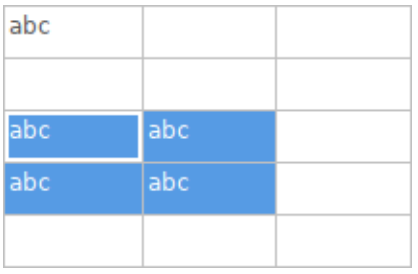
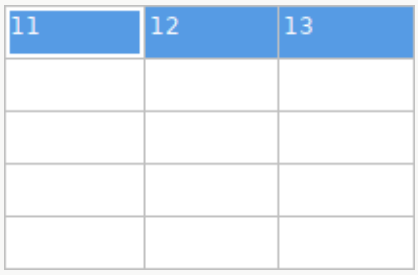
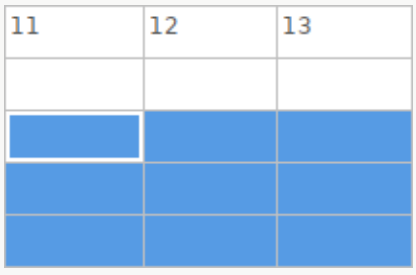
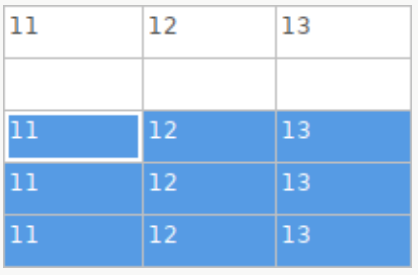
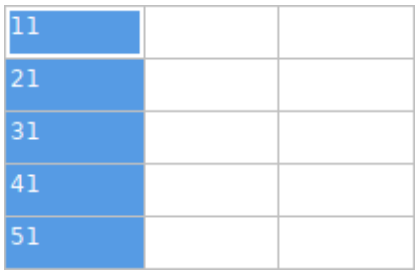
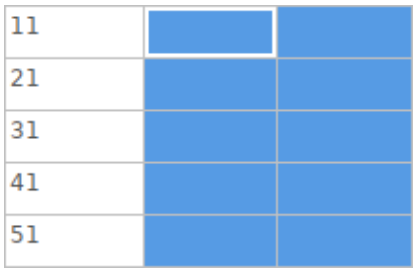
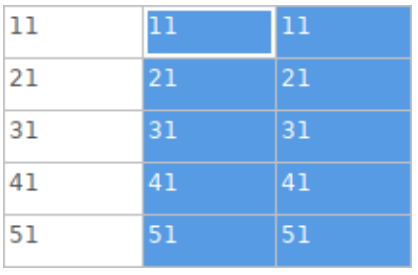
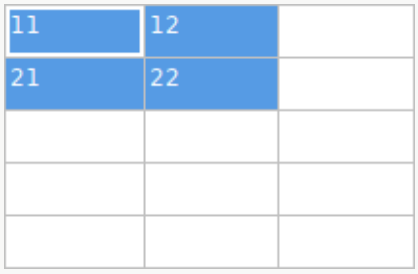
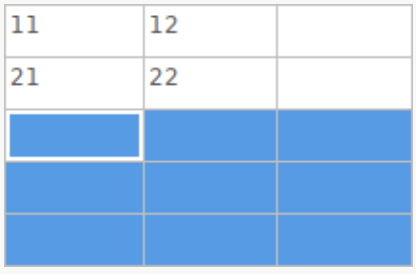
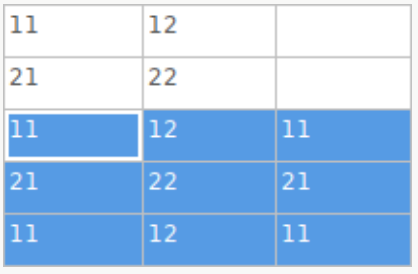
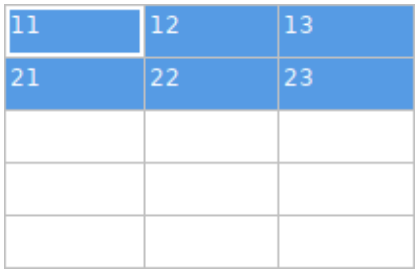
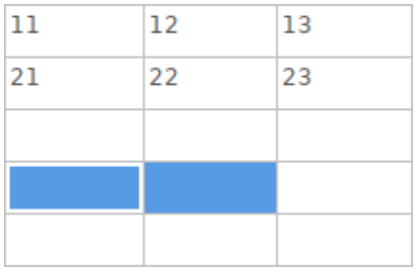


Once you modify field values, you need to either accept changes by clicking on 'Apply' button or undo them by clicking on 'Revert' button.

Tricks to simplify fields filling

There are several special copy/paste methods in spreadsheet. They may be useful when entering field values that are repeated in a few components.

These methods are illustrated below.

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

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

Import tool for footprint assignment

Access:

The icon  launches the back-annotate tool.

This tool allows footprint changes made in the PCB Editor to be imported back into the footprint fields in the Schematic Editor.

Manage Symbol Libraries

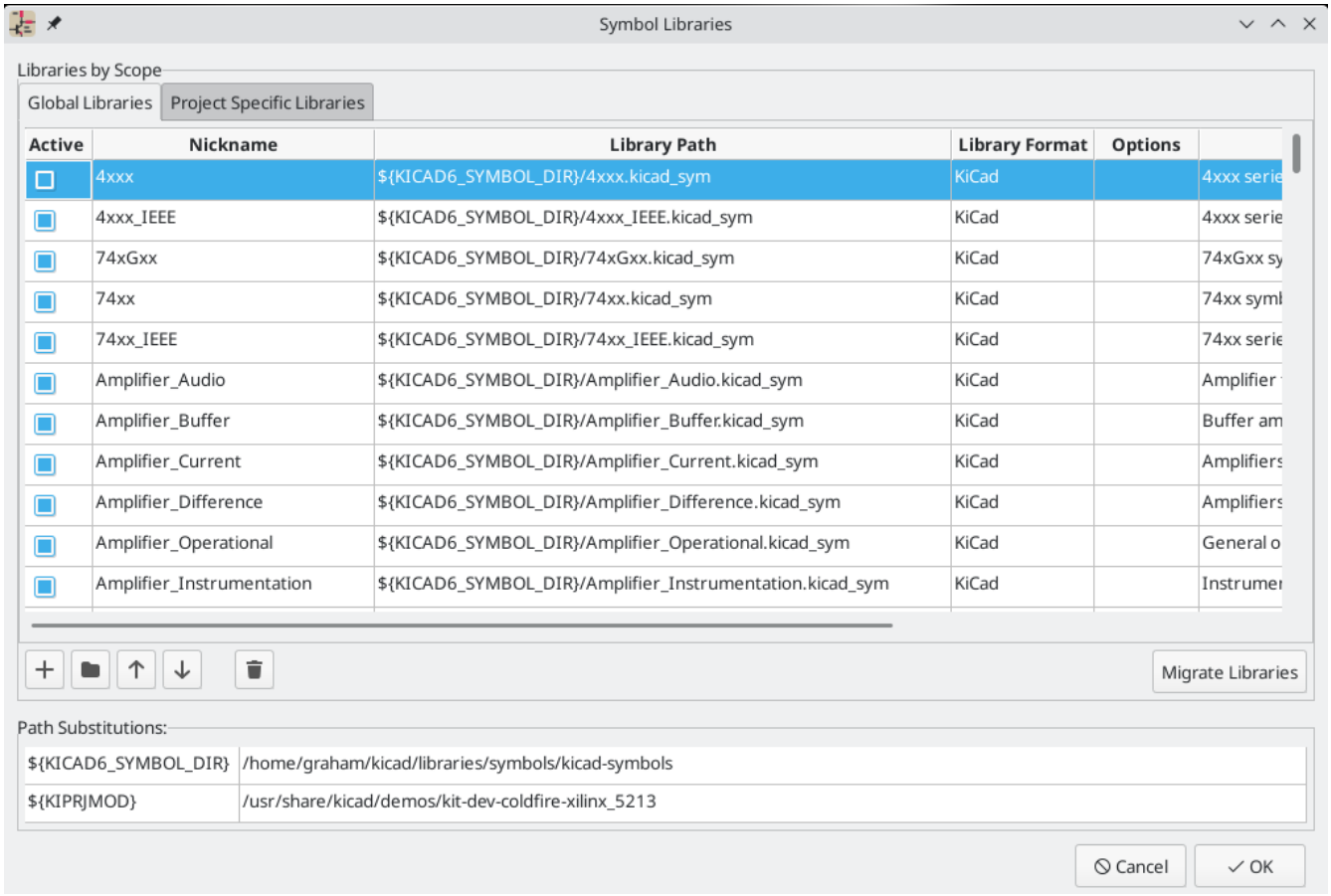
Symbol libraries hold collections of symbols used when creating schematics. Each symbol in a schematic is uniquely identified by a full name that is composed of a library nickname and a symbol name. An example is `Audio:AD1853`.

Symbol Library Table

The symbol library table holds a list of all library files KiCad knows about. The symbol library table is constructed from the global symbol library table file and the project specific symbol library table file.

When a symbol is loaded, KiCad uses the library nickname, `Audio` in our example, to lookup the library location in the symbol library table.

The image below shows the symbol library table editing dialog which can be opened by invoking the **Manage Symbol Libraries...** entry in the **Preferences** menu.



Global Symbol Library Table

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

Project Specific Symbol Library Table

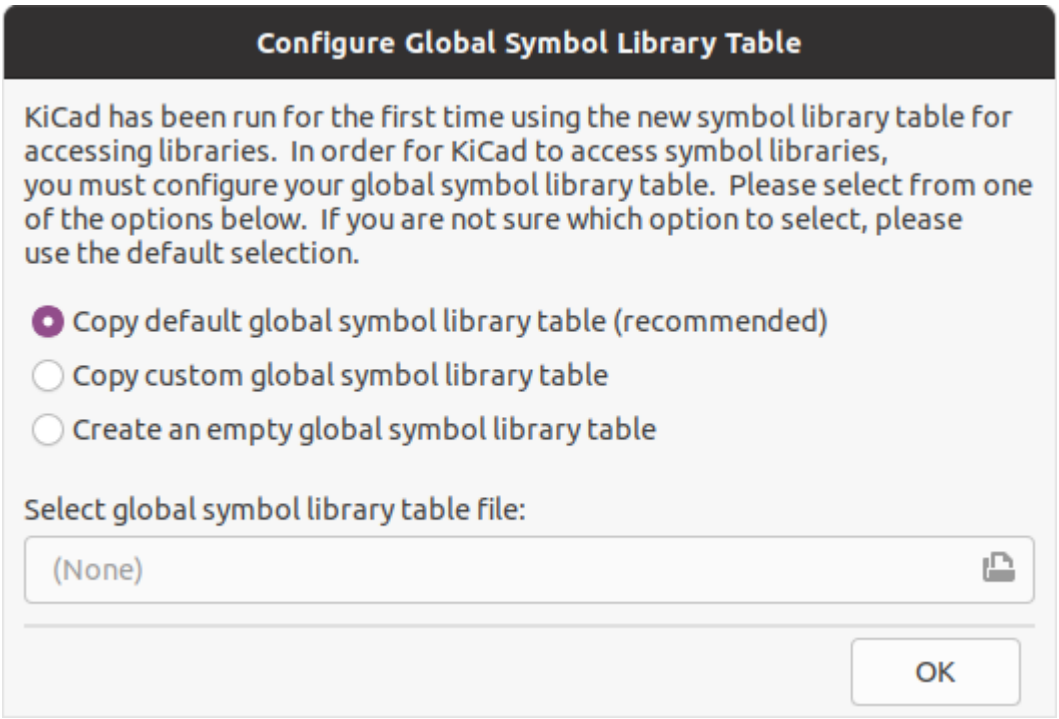
The project specific symbol library table contains the list of libraries that are available specifically for the currently loaded project file. The project specific symbol library table can only be edited when it is loaded

along with the project file. If no project file is loaded or there is no symbol library table file in the current project path, an empty table is created which can be edited and later saved along with the project file.

Initial Configuration

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 present the "Configure Global Symbol Library Table" dialog to the user. The dialog presents the user with three options.

- **Copy default global symbol library table (recommended).** If this option is selected, KiCad will copy the default symbol library table file stored in the system's Kicad template folder to the file `sym-lib-table` in the user's KiCad configuration folder. If the default template `sym-lib-table` file cannot be found, this option will be grayed out. The missing default table is usually caused by the KiCad default libraries not being installed (on some systems they are installed by a separate package). If the libraries are installed in a non-standard location, use the second option and browse to the library table location manually.
- **Copy custom global symbol library table.** If this option is selected, the user must browse to the desired symbol library table file, which will be copied to the user's KiCad configuration directory.
- **Create an empty global symbol library table.** An empty symbol library table file will be created in the user's KiCad configuration directory. The user must add libraries to the table manually.



NOTE

The default symbol library table includes all of the symbol libraries that are installed as part of KiCad. This may or may not be desirable depending on usages and the speed of the system. The amount of time required to load the symbol libraries is proportional to the number of libraries in the symbol library table. If symbol library load times are excessive, remove rarely and/or never used libraries from the global library table and add them to the project library table as required.

Adding Table Entries

In order to use a symbol library, it must first be added to either the global table or the project specific table. The project specific table is only applicable when you have a project file open.

NOTE	Each library entry must have a unique nickname.

The library nickname does not have to be related in any way to the actual library file name or path. The colon `:` and `\` characters cannot be used anywhere in the library nickname. Each library entry must have a valid path and/or file name depending on the type of library. Paths can be defined as absolute, relative, or by environment variable substitution (see section below).

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 Legacy Libraries).

There is also a 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.

- Please note that you cannot have duplicate library nicknames in the same table. However, you can have duplicate library nicknames in both the global and project specific symbol library table.
- The project specific table entry will take precedence over the global table entry when duplicate nicknames occur.
- When entries are defined in the project specific table, a sym-lib-table file containing the entries will be written into the folder of the currently open project file.

Environment Variable Substitution

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.

By default, at run time KiCad defines two environment variables relevant for locating symbol libraries:

- the `$KIPRJMOD` environment variable that always points to the currently open project directory. `$KIPRJMOD` cannot be modified.
- the `$KICAD6_SYMBOL_DIR` environment variable. This points to the path where the default symbol libraries that were installed with KiCad.

You can override `$KICAD6_SYMBOL_DIR` by redefining it in **Preferences → Configure Paths...**. This is useful for using libraries installed in a nonstandard location.

`$KIPRJMOD` allows you to store libraries in the project path without having to define the absolute path (which is not always known) to the library in the project specific symbol library table.

Usage Patterns

Symbol libraries can be defined either globally or specifically to the currently loaded project. Symbol libraries defined in the user's global table are always available and are stored in the `sym-lib-table` file in

the user's KiCad configuration folder. The project-specific symbol library table is active only for the currently open project file.

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 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 `Ctrl`-clicking or `shift`-clicking.

Libraries can also be converted one at a time by opening them in the Symbol Editor and saving them as a new library.

Legacy Project Remapping

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

- 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.

Schematic Creation and Editing

Introduction

A schematic can be represented by a single sheet, but, if big enough, it will require several sheets.

A schematic represented by several sheets is hierarchical, and all its sheets (each one represented by its own file) constitute a complete KiCad schematic. The manipulation of hierarchical schematics will be described in the Hierarchical Schematics chapter.

General considerations

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:

- Validating against a set of rules (Electrical Rules Check) to detect errors and omissions.
- Automatically generating a bill of materials (BOM).
- Generating a netlist for simulation software such as SPICE.
- Defining a circuit for transferring to PCB layout.

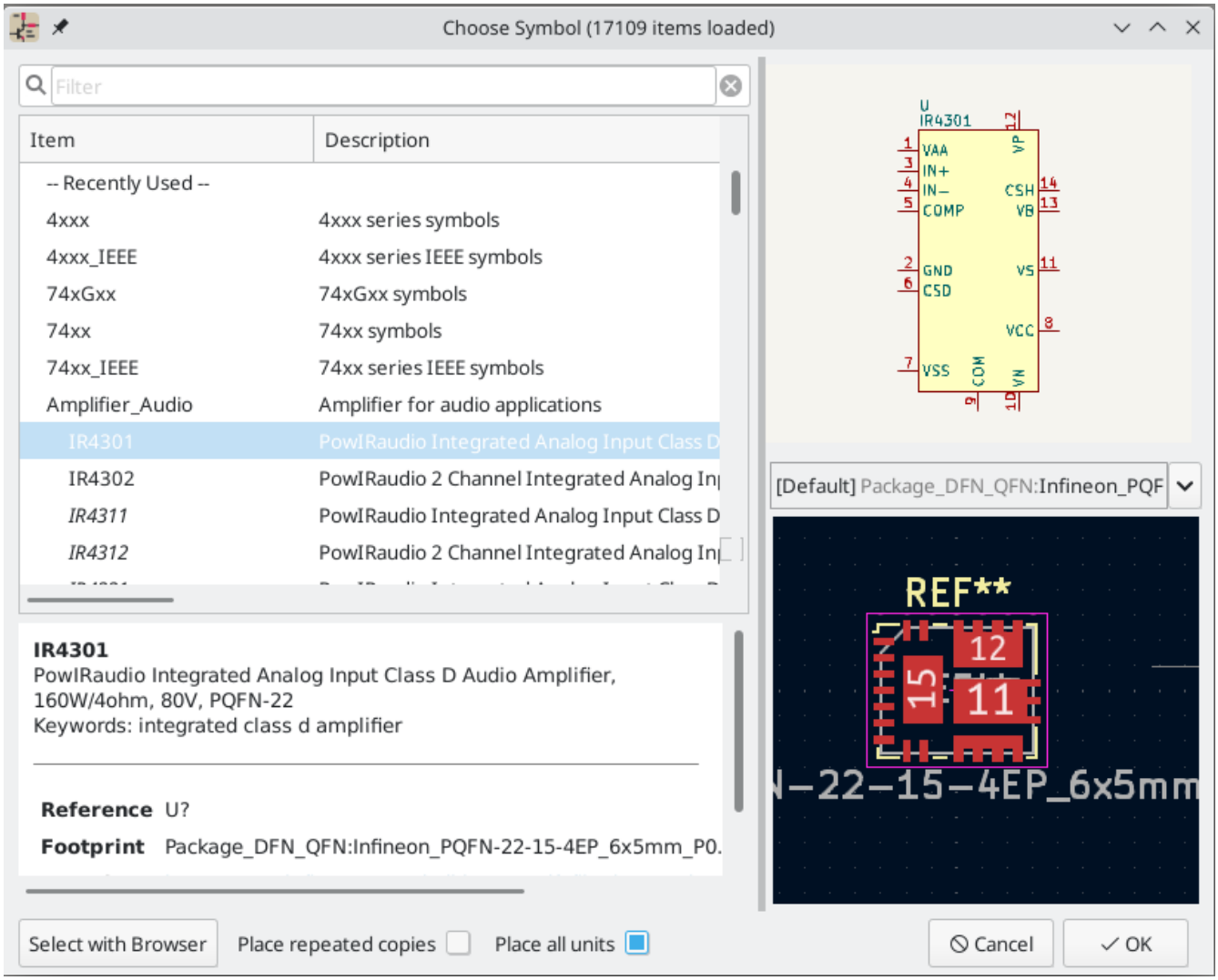
A schematic mainly consists of symbols, wires, labels, junctions, buses and power ports. For clarity in the schematic, you can place purely graphical elements like bus entries, comments, and polylines.

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.

Symbol placement and editing

Find and place a symbol

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. Advanced filters can be used just by typing them:

- **Wildcards:** use the characters `?` and `*` respectively to mean "any character" and "any number of characters".
- **Relational:** 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}

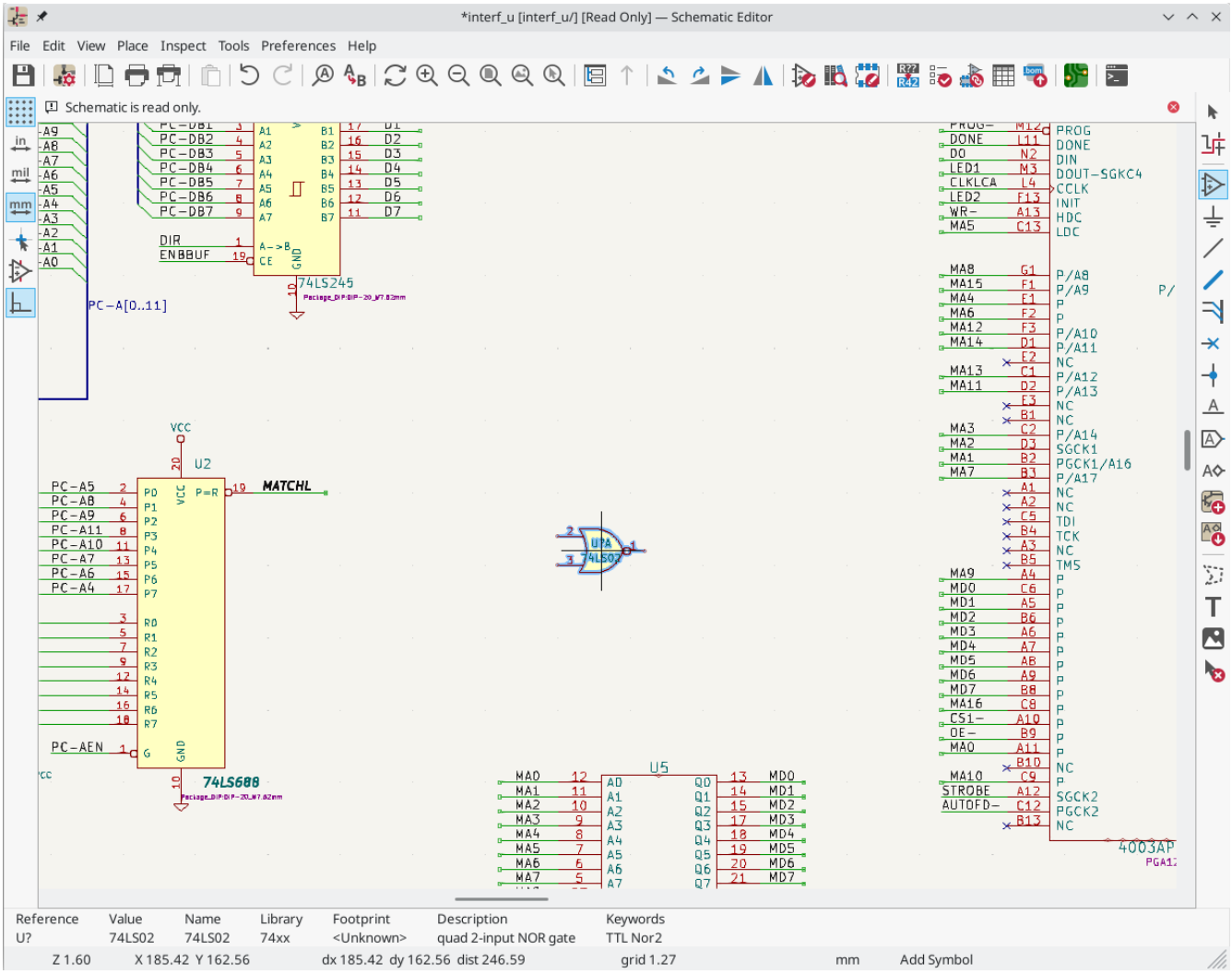
ki	mi	gi	ti
2^{10}	2^{20}	2^{30}	2^{40}

- **Regular expression:** 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.

Symbol Editing and Modification (already placed component)

There are two ways to edit a symbol:

- Modification of the symbol itself: position, orientation, unit selection on a multi-unit symbol.
- Modification of one of the fields of the symbol: reference, value, footprint, etc.

When a symbol has just been placed, you may have to modify its value (particularly for resistors, capacitors, etc.), but it is useless to assign to it a reference number right away, or to select the unit (except for components with locked units, which you have to assign manually). This can be done automatically by the annotation function.

Symbol modification

To modify some feature of a symbol, position the cursor on the symbol, and then either:

- Double-click on the symbol to open the full editing dialog.
- Right-click to open the context menu and use one of the commands: Move, Orientation, Edit, Delete, etc.
- Use a hotkey to perform an action on the symbol (<kbd>E</kbd> to open the properties dialog, <kbd>R</kbd> to rotate, etc.). Note that hotkeys act on the selected symbol; if no symbol is selected hotkeys act on the symbol under the cursor.

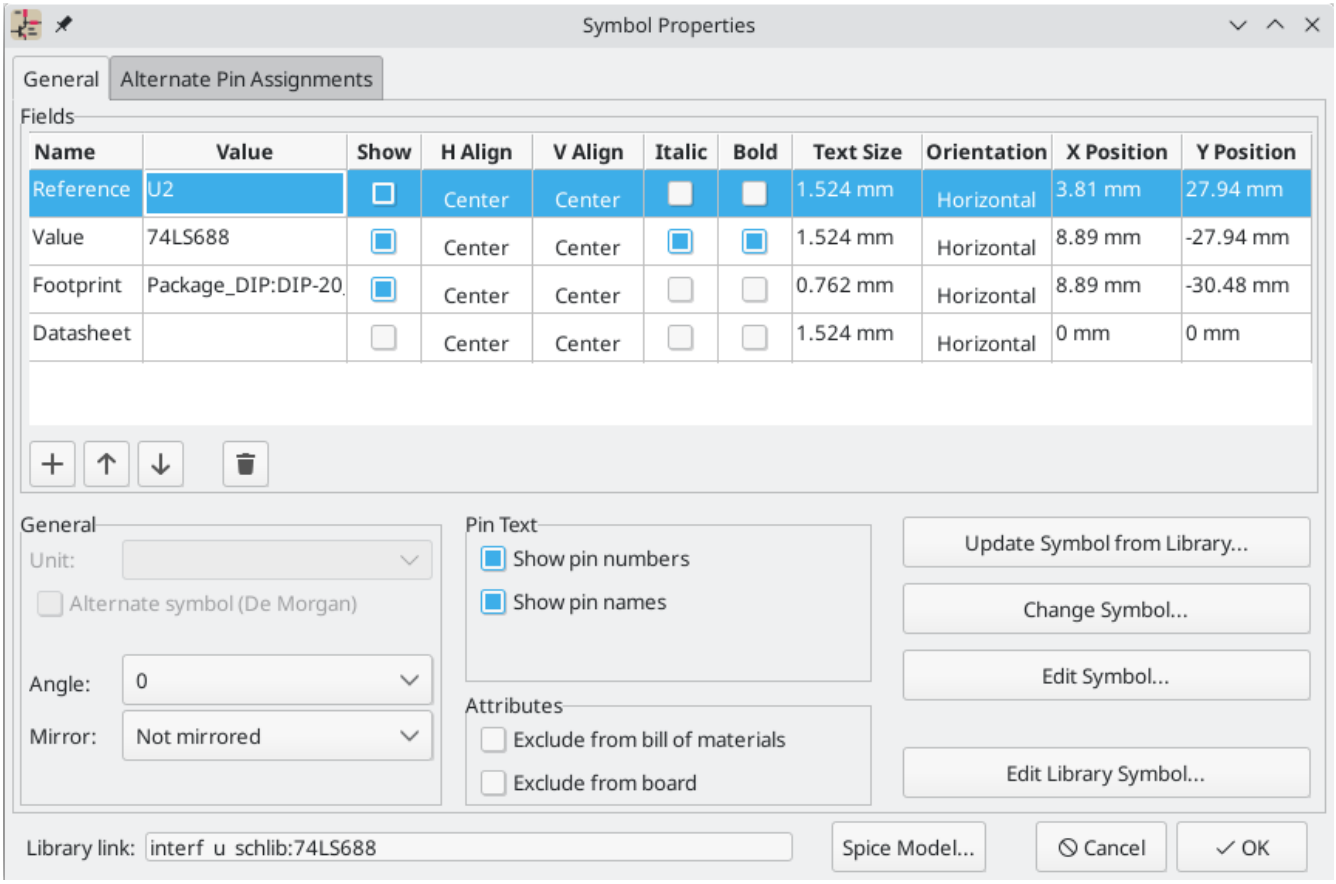
Symbols can also be selected by clicking on them or drag-selecting them. Selected symbols can be modified by clicking relevant buttons in the top toolbar or using a hotkey.

Text fields modification

You can modify the reference, value, position, orientation, text size and visibility of the fields:

- Double-click on the text field to modify it.
- Right-click to open the context menu and use one of the commands: Move, Rotate, Edit, Delete, etc.
- Position the cursor over the field (if nothing is selected) or select the field and press <kbd>E</kbd> to edit the field.
- Position the cursor over the symbol (if nothing is selected) or select the symbol and press <kbd>V</kbd>, <kbd>U</kbd>, or <kbd>F</kbd> hotkeys to directly edit the symbol's value, reference designator, or footprint fields, respectively.

For more options, or in order to create fields, double-click on the symbol to open the Symbol Properties dialog.



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.

Electrical Connections

Introduction

There are a number of elements that can be added to a schematic to electrically connect components. All of these elements can be placed with the buttons on the vertical right toolbar or using hotkeys.

These elements are:

- **Wires:** direct connection between pins.
- **Buses:** connections for a group of signals.

Bus entries: connections between wires and buses.

- **No-connection flags:** terminations for pins or wires that are intentionally unconnected. These flags prevent ERC violations for unconnected pins.
- **Junctions:** connections between crossing wires or buses.
- **Net labels:** local name for a signal. Signals within a sheet that have the same net label are connected.
- **Global labels:** global name for a signal. Signals with the same global label are connected even if they are not in the same sheet.
- **Hierarchical labels:** a label for a signal in a subsheet that enables the signal to be accessed in a parent sheet. See the Hierarchical Schematics section for more information about hierarchical labels, sheets, and pins.
- **Hierarchical sheets:** an instantiation of a subsheet within a parent sheet. The parent sheet can connect to the subsheet through the subsheet's hierarchical pins.
- **Hierarchical pins:** connection points between a parent sheet and a subsheet. Hierarchical pins appear at the parent sheet's level and correspond to hierarchical labels in the subsheet.

Several other types of items can be placed on the schematic but do not affect connectivity:

- **Graphical lines:** graphical lines for presentation.
- **Text:** textual comments and annotations.
- **Bitmap images:** raster graphics from an external file.

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

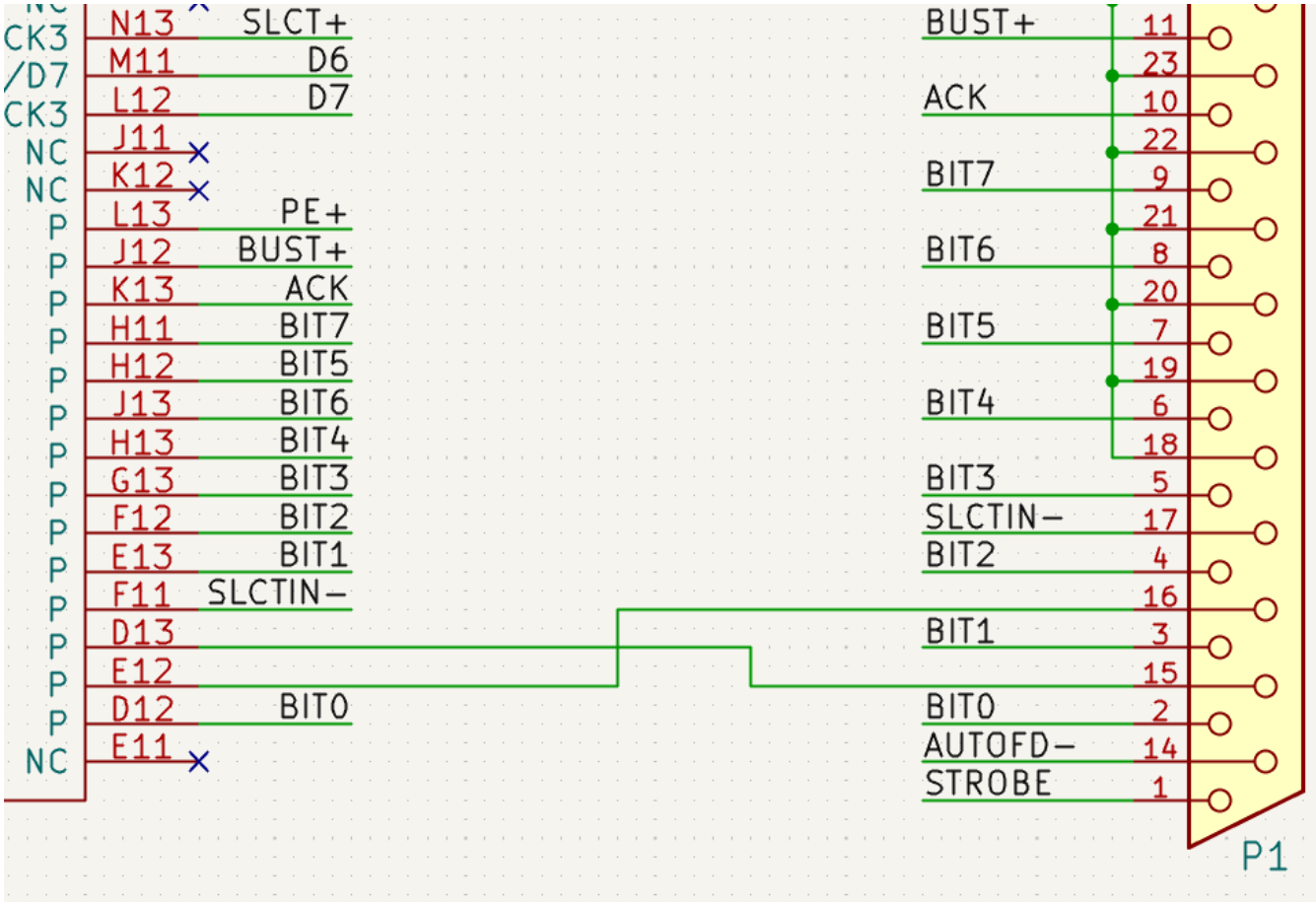
- **Power ports**: symbols for connecting wires to a power or ground net.
- **PWR_FLAG**: a specific symbol for indicating that a net is powered when it is not connected to a power output pin (for example, a power net that is supplied by an off-board connector).

Connections (Wires and Labels)

There are two ways to establish connection:

- Pin to pin wires.
- Labels.

The following figure shows the two methods:



Label Connections

The point of "contact" of a label is the small square in the corner of the label. The square disappears when the label is connected. The position of the connection point relative to the label text can be changed by choosing a different label orientation in the label properties, or by mirroring/rotating the label.

The label's connection point must be in contact with a wire or the end of a pin for the label to be connected.

Wire Connections

To establish a connection, a segment of wire must be connected by its ends to an another segment or to a pin.

If there is overlapping (if a wire passes over a pin, but without being connected to the pin end) there is no connection.

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**.

Wire Junctions

Wires that cross are not implicitly connected. It is necessary to join them with a junction dot if a connection is desired. Junction dots will be automatically added to wires that start or end on top of an existing wire.

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

Nets with Multiple Names

A signal 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.

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 automatically be connected to the global `VCC` net.

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, the use of invisible power pins in symbols is not recommended outside of power port symbols, and is only supported for compatibility with legacy designs and symbols.

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

It is not recommended to use labels for power connection. These only have a "local" connection scope, and will not connect to invisible power pins.

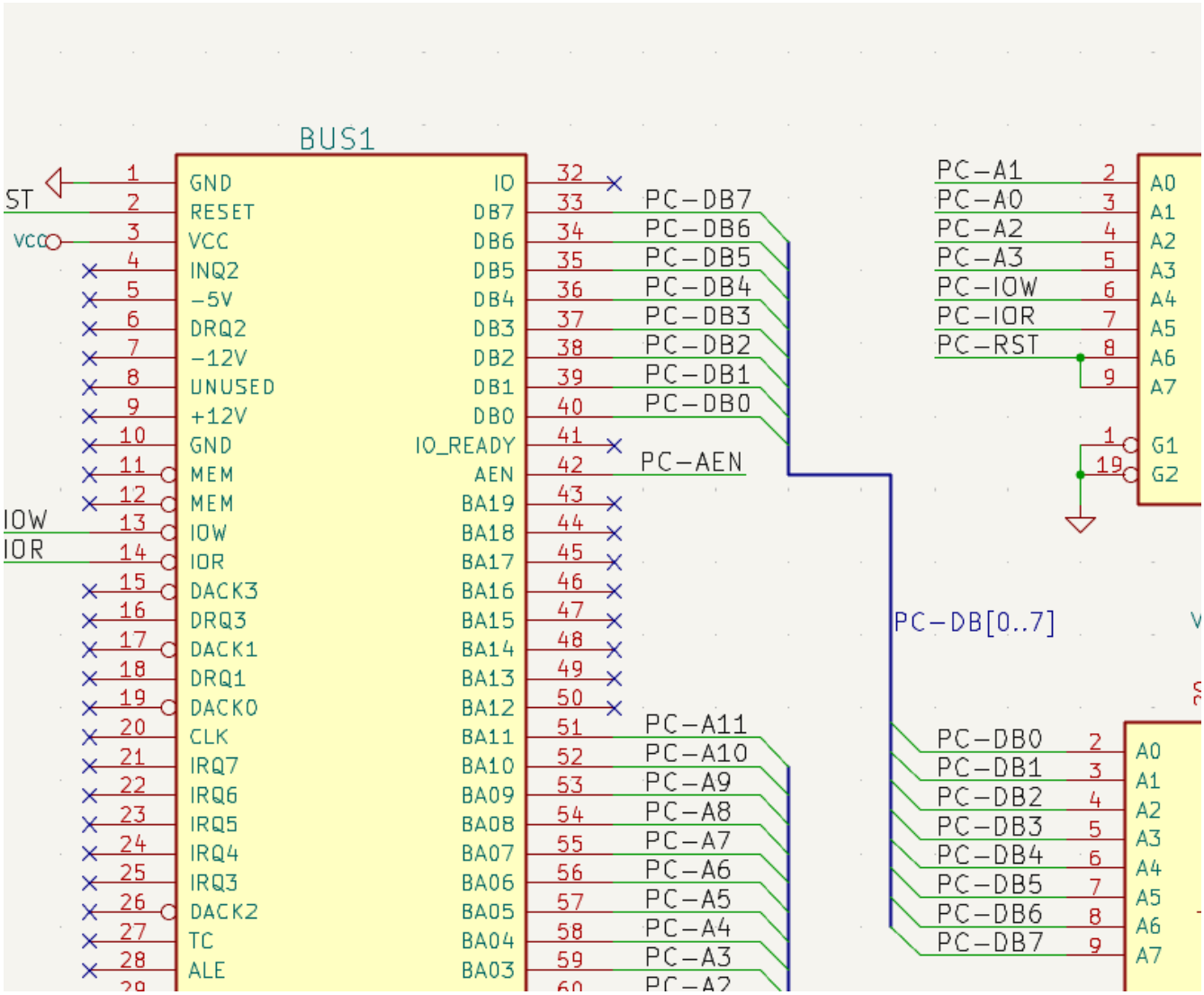
Wiring

To begin connecting elements, you may either use the 'Wire' or 'Bus' tools from the right-hand toolbar, or you can auto-start a new wire from any existing pin or unconnected wire.

The wire drag action will drag the entire wire if you start dragging from the middle of the wire. Alternatively, it will drag just one corner if you start the drag action over a corner where two wires connect

Connections (Buses)

In the following schematic, many pins are connected to buses.



Bus members

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. 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.

Connections between bus members

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.

In the example above, connections are made by the labels placed on wires connected to the pins. Bus entries (wire segments at 45 degrees) to buses are graphical only, and are not necessary to form logical connections.

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.
- Draw the wire under the first label. Then use the repetition command to place the other wires under the labels.
- If needed, place the bus entries by the same way (Place the first entry, then use the repetition command).

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.

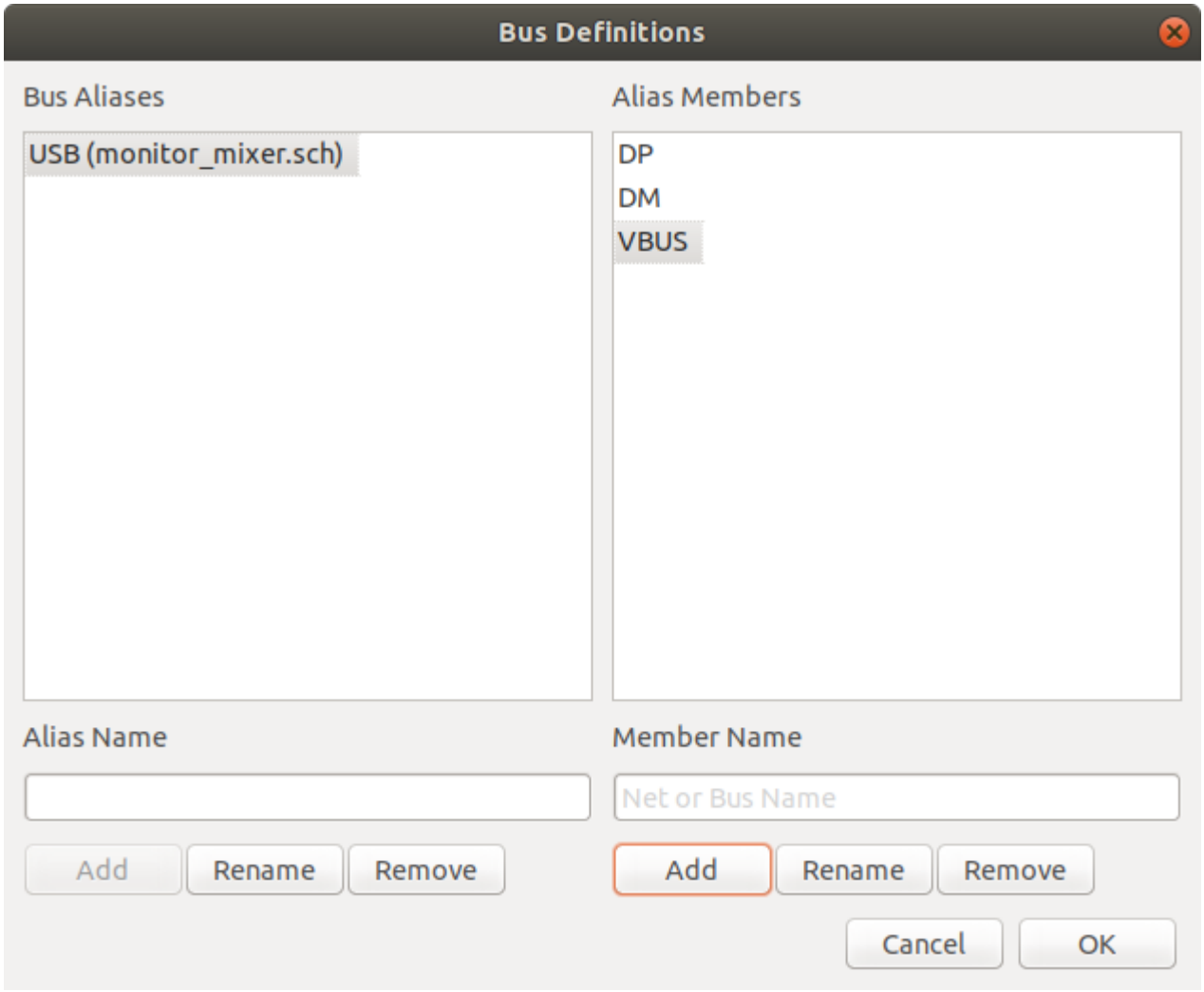
After selecting the bus member, the next click will place the bus member label at the desired location. The tool automatically generates a bus entry and wire leading up to the label location. After placing the label, you can continue placing additional wire segments (for example, to connect to a component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

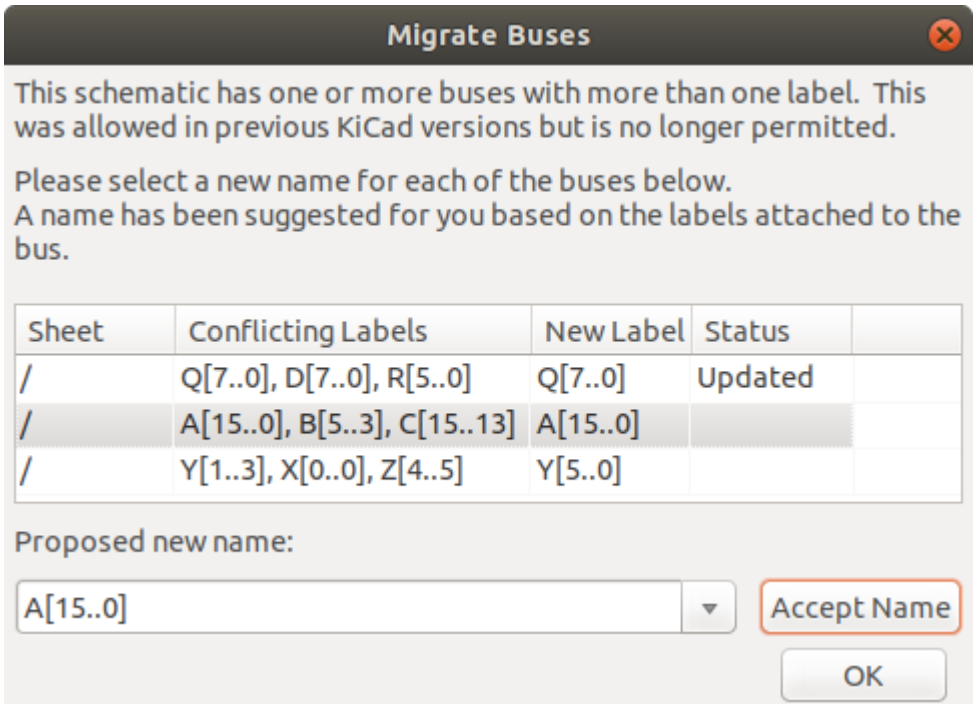
After defining an alias, it can be used in a group bus label by putting the alias name inside the curly braces of the group bus: `{USB}`. This has the same effect as labeling the bus `{DP DM VBUS}`. You can also add a prefix name to the group, such as `USB1{USB}`, which results in nets such as `USB1.DP`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



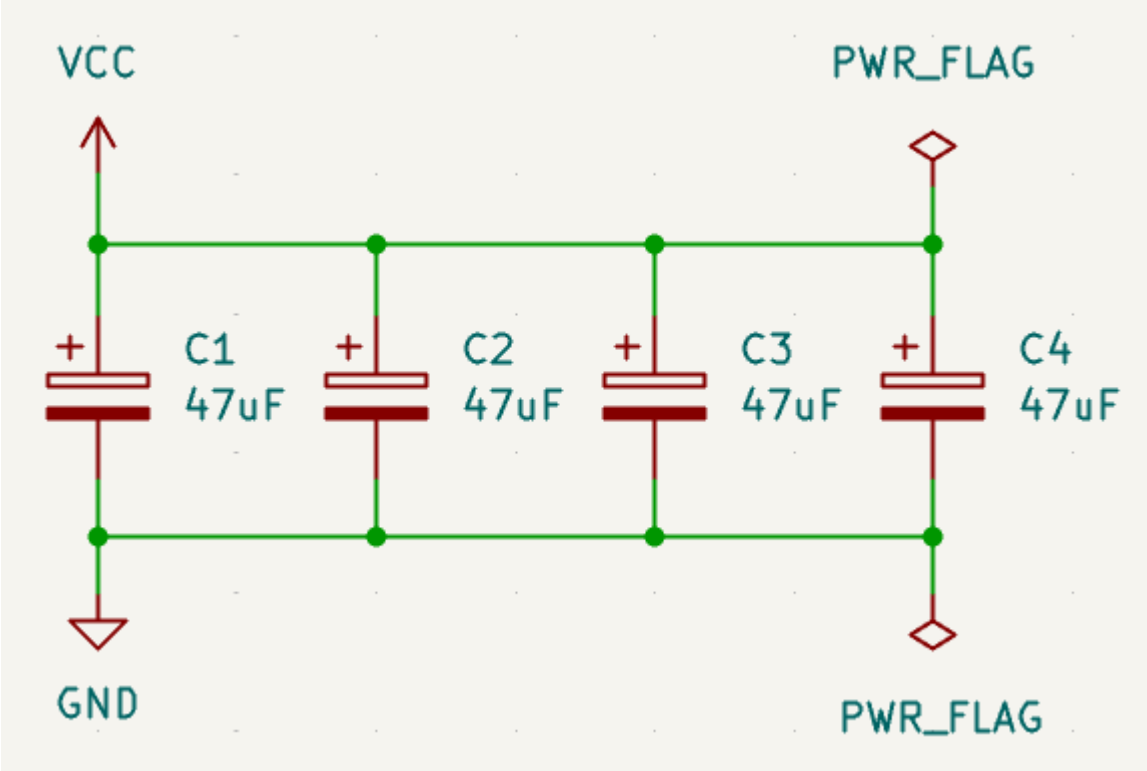
For each set of bus wires that has more than one label, you must choose the label to keep. The drop-down name box lets you choose between the labels that exist in the design, or you can choose a different name by manually entering it into the new name field.

Power Ports

Power port symbols are conventionally used to connect pins to power nets. Power port symbols have a single pin which is invisible and marked as a power input. As described in the hidden power pins section, any wire connected to the pin of a power port is therefore automatically connected to the power net with the same name as the port's pin.

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 hidden pin marked as a power input. Name the pin according to the desired power net.

The figure below shows an example of power port connections.



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

Power port symbols are found in the `power` symbol library. They can also be created by drawing a symbol with a hidden "power input" pin that has the name of the desired power net.

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

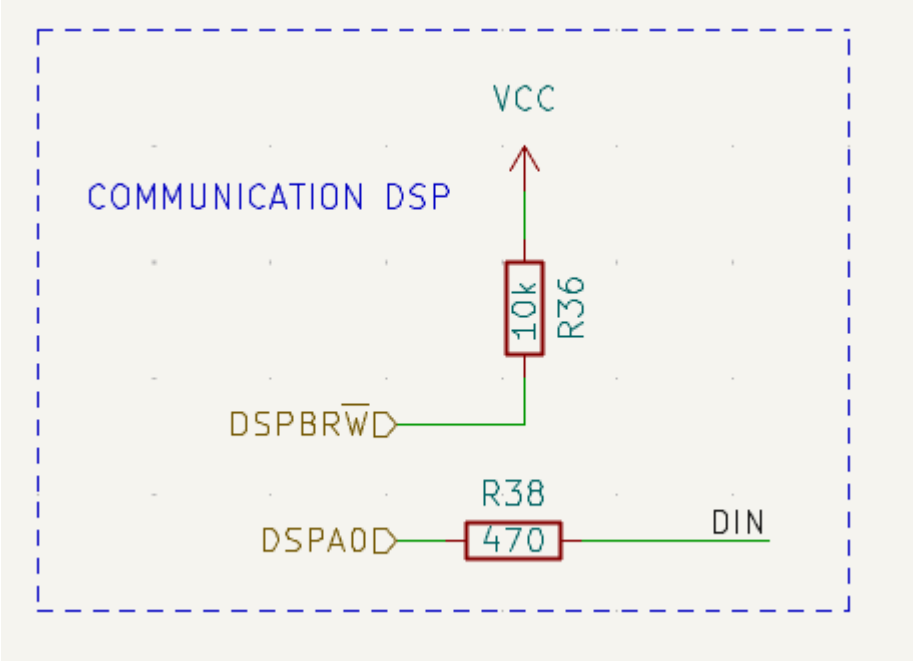
No-connection flags ( ) are used to indicate that a pin is intentionally unconnected. These flags do not have any effect on the schematic's connectivity, but they prevent "unconnected pin" ERC warnings for pins that are intentionally unconnected.

Drawing Complements

Text comments and graphic lines

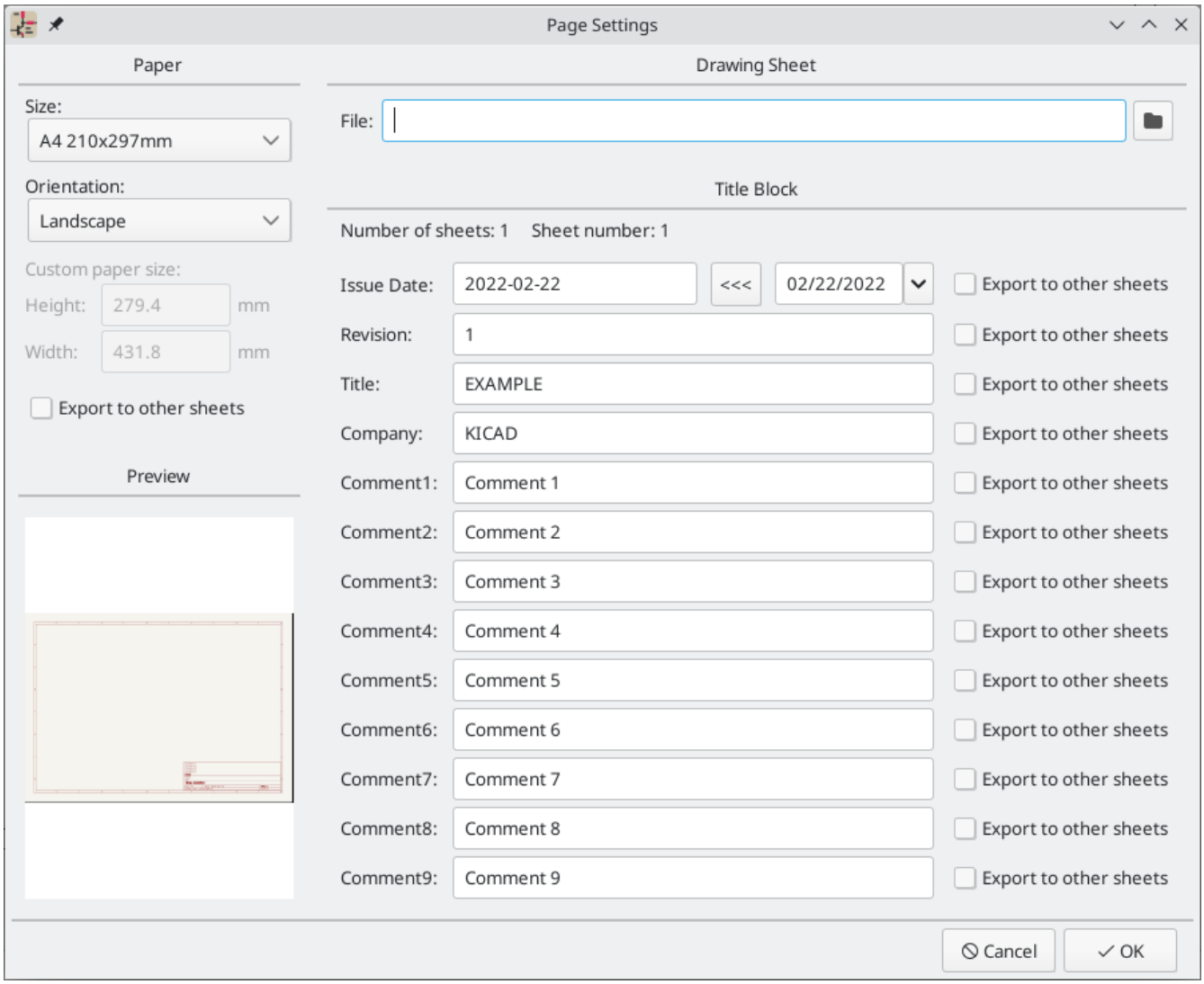
It can be useful to place annotations such as text fields and frames to aid in understanding the schematic. Text fields ( ) and graphic lines ( ) are intended for this use, as opposed to labels and wires, which are connection elements.

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



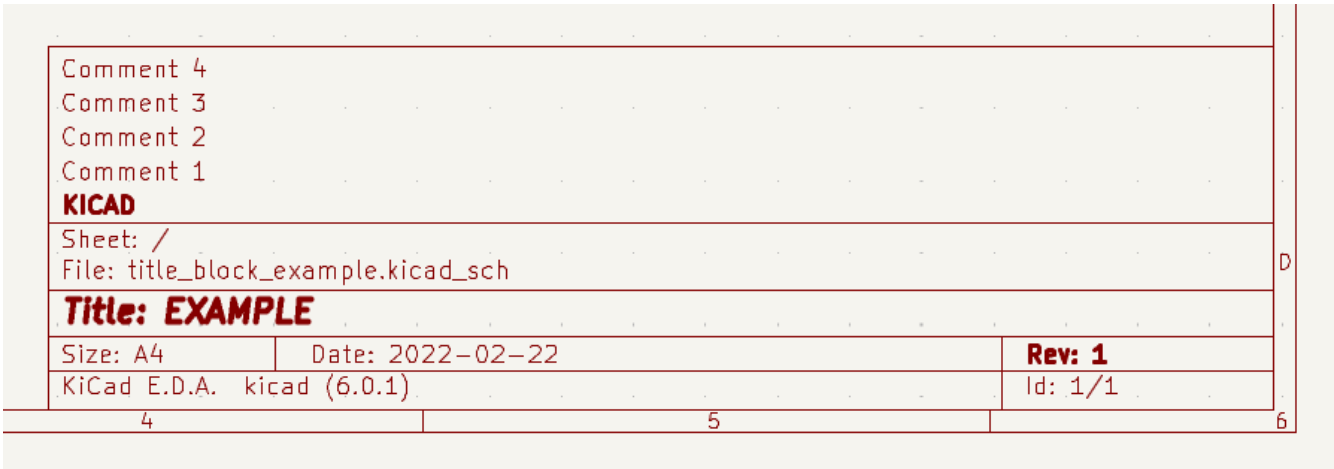
Sheet title block

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.

A drawing sheet template file can also be selected.

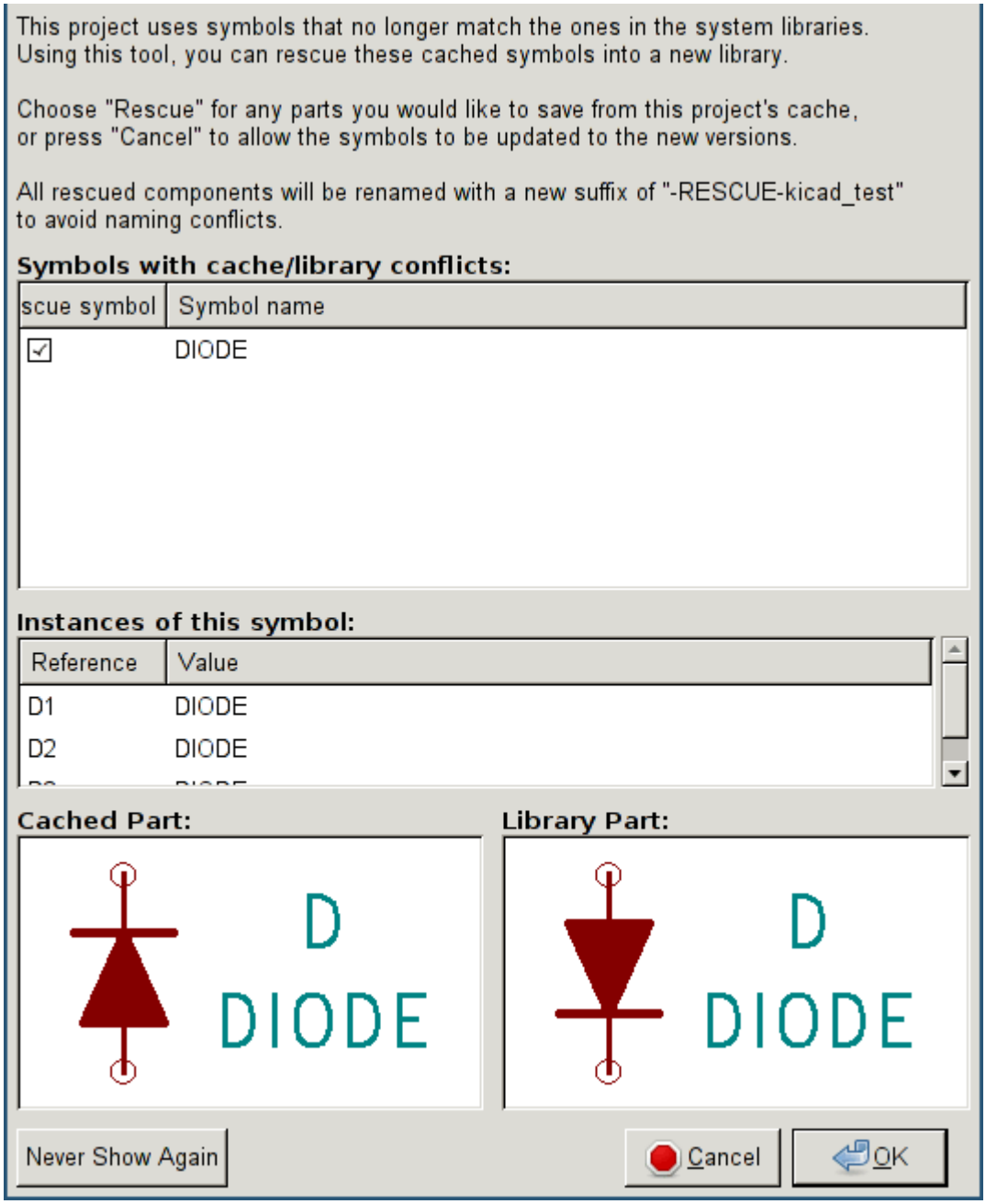


The sheet number (Sheet X/Y) is automatically updated, but sheet page numbers can also be manually set using **Edit → Edit Sheet Page Number...**.

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.

Hierarchical schematics

Introduction

A hierarchical representation is generally a good solution for projects bigger than a few sheets. If you want to manage this kind of project, it will be necessary to:

- Use large sheets, which results in printing and handling problems.
- Use several sheets, which leads you to a hierarchy structure.

The complete schematic then consists in a main schematic sheet, called root sheet, and sub-sheets constituting the hierarchy. Moreover, a skillful subdividing of the design into separate sheets often improves on its readability.

From the root sheet, you must be able to find all sub-sheets. Hierarchical schematics management is very easy with KiCad, thanks to an integrated "hierarchy navigator" accessible via the icon  of the top toolbar.

There are two types of hierarchy that can exist simultaneously: the first one has just been evoked and is of general use. The second consists in creating symbols in the library that appear like traditional symbols in the schematic, but which actually correspond to a schematic which describes their internal structure.

This second type is used to develop integrated circuits, because in this case you have to use function libraries in the schematic you are drawing.

KiCad currently doesn’t treat this second case.

A hierarchy can be:

- simple: a given sheet is used only once
- complex: a given sheet is used more than once (multiples instances)
- flat: which is a simple hierarchy, but connections between sheets are not drawn.

KiCad can deal with all these hierarchies.

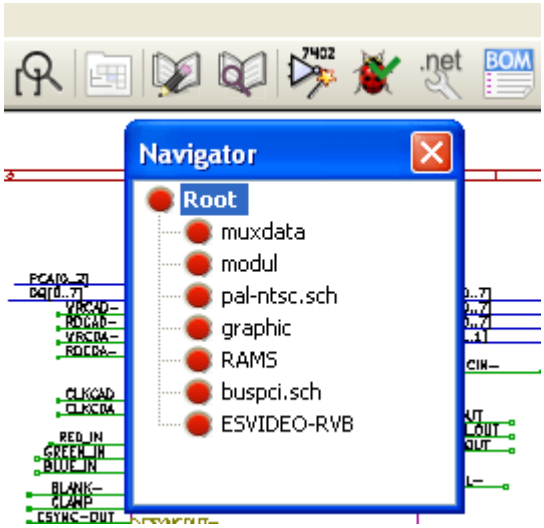
The creation of a hierarchical schematic is easy, the whole hierarchy is handled starting from the root schematic, as if you had only one schematic.

The two important steps to understand are:

- How to create a sub-sheet.
- How to build electrical connections between sub-sheets.

Navigation in the Hierarchy

Navigation among sub-sheets is acheived by using the navigator tool accessible via the button  on the top toolbar.



Each sheet is reachable by clicking on its name. For quick access, right click on a sheet name, and choose to Enter Sheet or double click within the bounds of the sheet.

In order to exit the current sheet to the parent sheet, right click anywhere in the schematic where there is no object and select "Leave Sheet" in the context menu or press Alt+Backspace.

Local, hierarchical and global labels

Properties

Local labels, tool A, are connecting signals only within a sheet. Hierarchical labels (tool A◇) are connecting signals only within a sheet and to a hierarchical pin placed in the parent sheet.

Global labels (tool A▷) are connecting signals across all the hierarchy. Power pins (type *power in* and *power out*) invisible are like global labels because they are seen as connected between them across all the hierarchy.

NOTE	Within a hierarchy (simple or complex) one can use both hierarchical labels and/or global labels.

Summary of hierarchy creation

You have to:

- Place in the root sheet a hierarchy symbol called "sheet symbol".
- Enter into the new schematic (sub-sheet) with the navigator and draw it, like any other schematic.
- Draw the electric connections between the two schematics by placing Global Labels (HLabels) in the new schematic (sub-sheet), and labels having the same name in the root sheet, known as SheetLabels. These SheetLabels will be connected to the sheet symbol of the root sheet to the other elements of the schematic like standard symbol pins.

Sheet symbol

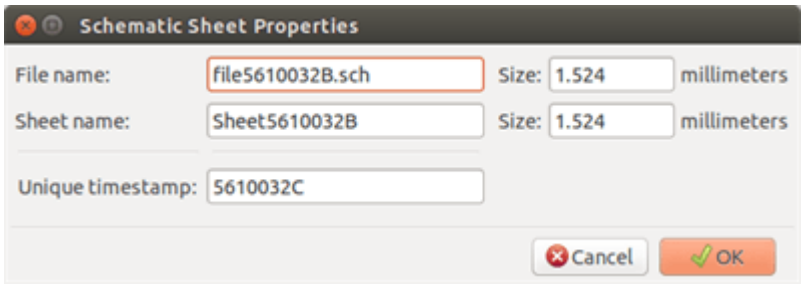
Draw a rectangle defined by two diagonal points symbolizing the sub-sheet.

The size of this rectangle must allow you to place later particular labels, hierarchy pins, corresponding to the global labels (HLabels) in the sub-sheet.

These labels are similar to usual symbol pins. Select the tool .

Click to place the upper left corner of the rectangle. Click again to place the lower right corner, having a large enough rectangle.

You will then be prompted to type a file name and a sheet name for this sub-sheet (in order to reach the corresponding schematic, using the hierarchy navigator).



You must give at least a file name. If there is no sheet name, the file name will be used as sheet name (usual way to do that).

Connections - hierarchical pins

You will create here points of connection (hierarchy pins) for the symbol which has been just created.

These points of connection are similar to normal symbol pins, with however the possibility to connect a complete bus with only one point of connection.

Importing Hierarchical Sheet Pins

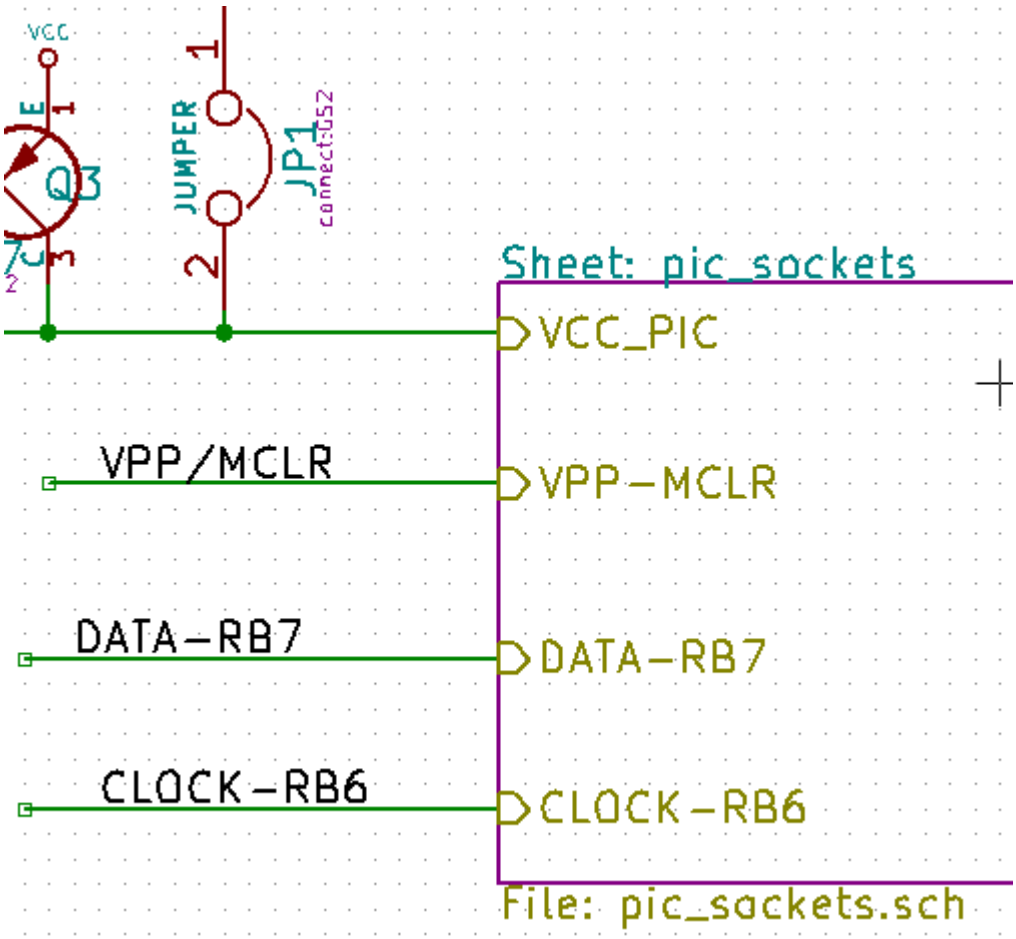
- Select the tool .
- Click on the hierarchical sheet from where you want to import the pins corresponding to hierarchical labels placed in the corresponding schematic. A hierarchical pin appears, if a new hierarchical label exists, i.e. not corresponding to an already placed pin.
- Click where you want to place this pin.

All necessary pins can thus be placed quickly and without error. Their aspect is in accordance with corresponding hierarchical labels.

Connections - hierarchical labels

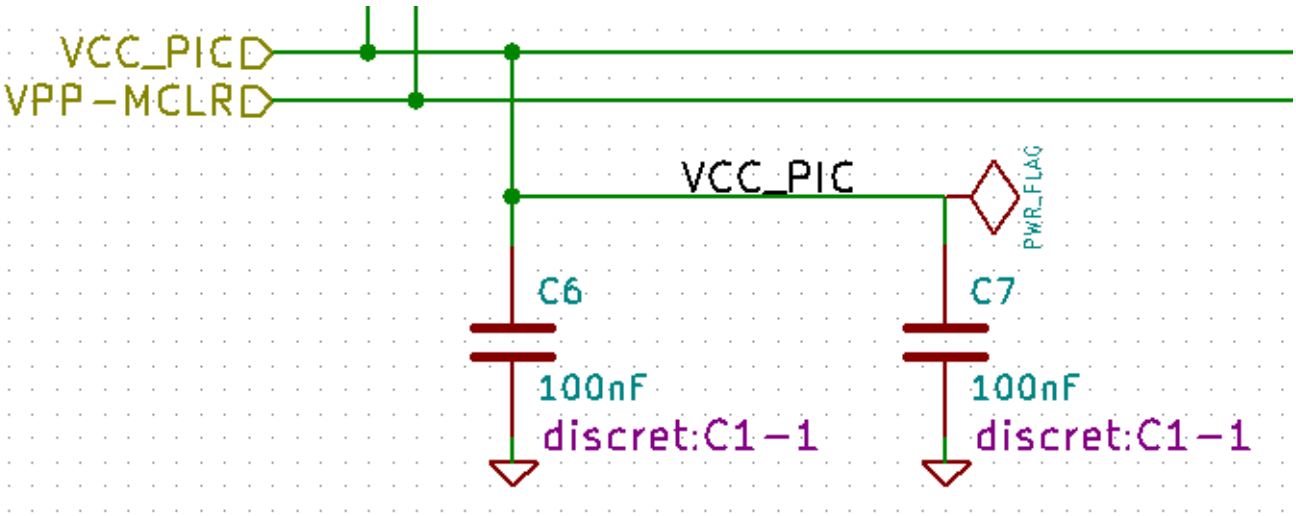
Each pin of the sheet symbol just created, must correspond to a label called hierarchical Label in the sub-sheet. Hierarchical labels are similar to labels, but they provide connections between sub-sheet and root sheet. The graphical representation of the two complementary labels (pin and hierarchical labels) is similar. Hierarchical labels are made with the tool .

See below a root sheet example:



Notice pin VCC_PIC, connected to connector JP1.

Here are the corresponding connections in the sub-sheet :



You find again, the two corresponding hierarchical labels, providing connection between the two hierarchical sheets.

NOTE	You can use hierarchical labels and hierarchy pins to connect two buses, according to the syntax (Bus [N. .m]) previously described.

Labels, hierarchical labels, global labels and invisible power pins

Here are some comments on various ways to provide connections, other than wire connections.

Simple labels

Simple labels have a local capacity of connection, i.e. limited to the schematic sheet where they are placed. This is due to the fact that :

- Each sheet has a sheet number.
- This sheet number is associated to a label.

Thus, if you place the label "TOTO" in sheet n° 3, in fact the true label is "TOTO_3". If you also place a label "TOTO" in sheet n° 1 (root sheet) you place in fact a label called "TOTO_1", different from "TOTO_3". This is always true, even if there is only one sheet.

Hierarchical labels

What is said for the simple labels is also true for hierarchical labels.

Thus in the same sheet, a hierarchical label "TOTO" is considered to be connected to a local label "TOTO", but not connected to a hierarchical label or label called "TOTO" in another sheet.

A hierarchical label is considered to be connected to the corresponding sheet pin symbol in the hierarchical symbol placed in the parent sheet.

Invisible power pins

It was seen that invisible power pins were connected together if they have the same name. Thus all the power pins declared "Invisible Power Pins" and named VCC are connected all symbol invisible power pins named VCC only within the sheet they are placed.

This means that if you place a VCC label in a sub-sheet, it will not be connected to VCC pins, because this label is actually VCC_n, where n is the sheet number.

If you want this label VCC to be really connected to the VCC for the entire schematic, it will have to be explicitly connected to an invisible power pin via a VCC power symbol.

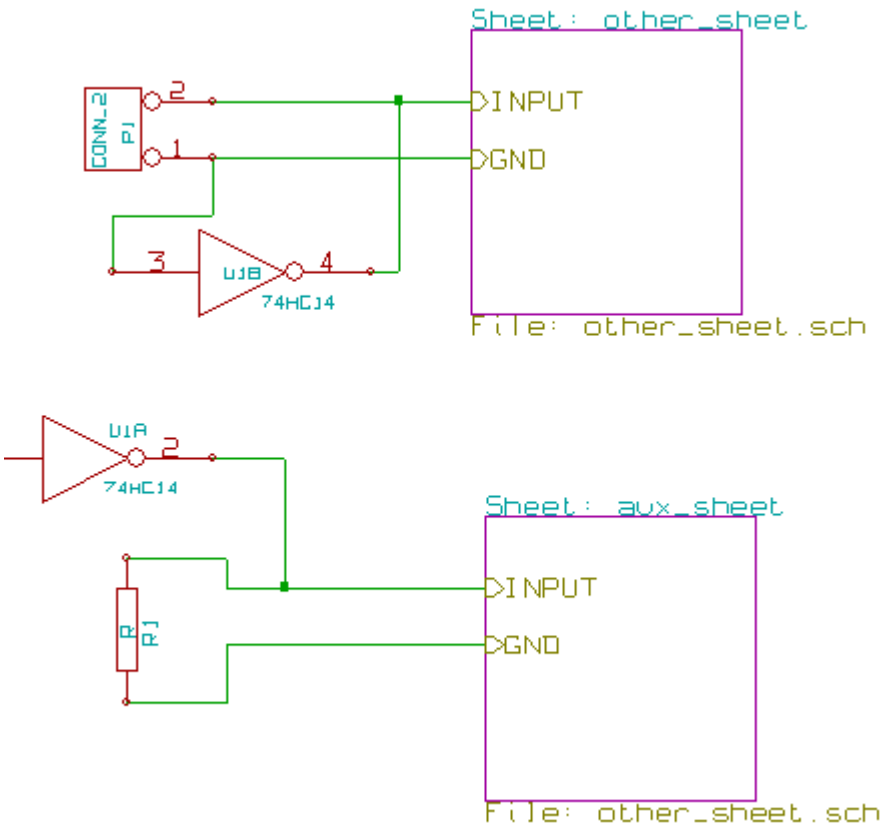
Global labels

Global labels that have an identical name are connected across the whole hierarchy.

(power labels like vcc ... are global labels)

Complex Hierarchy

Here is an example. The same schematic is used twice (two instances). The two sheets share the same schematic because the file name is the same for the two sheets (``other_sheet.sch''). The sheet names must be unique.

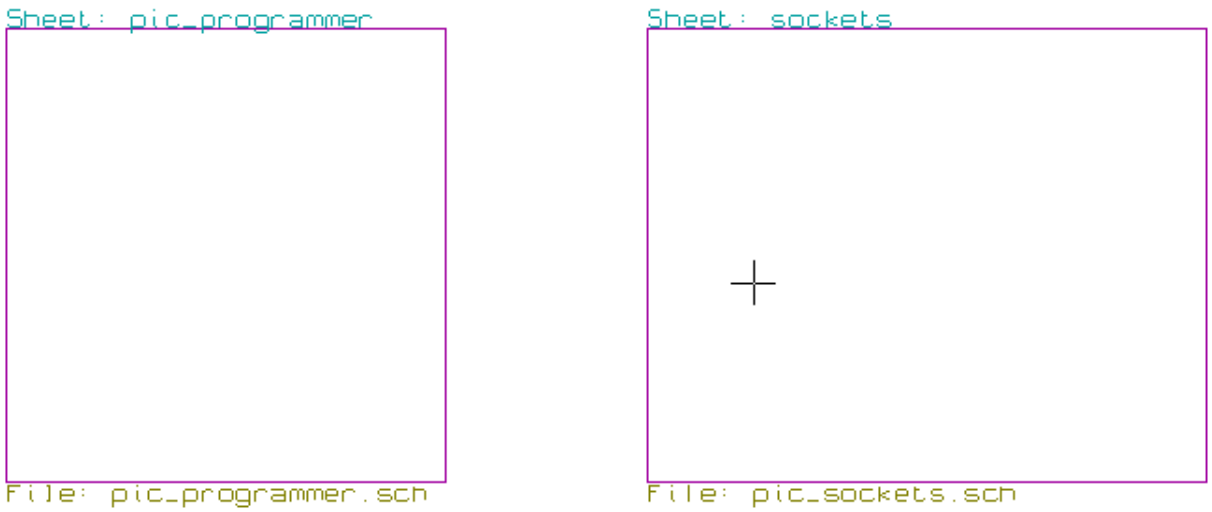


Flat hierarchy

You can create a project using many sheets without creating connections between these sheets (flat hierarchy) if the following rules are observed:

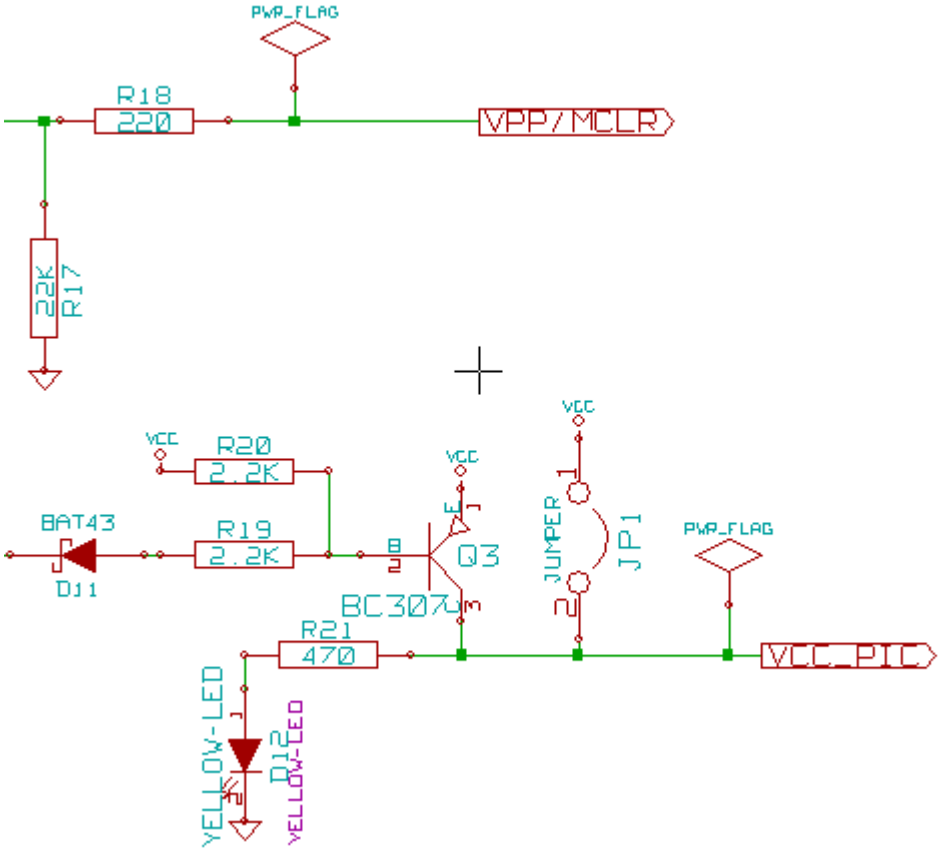
- Create a root sheet containing the other sheets which acts as a link between others sheets.
- No explicit connections are needed.
- Use global labels instead of hierarchical labels in all sheets.

Here is an example of a root sheet.

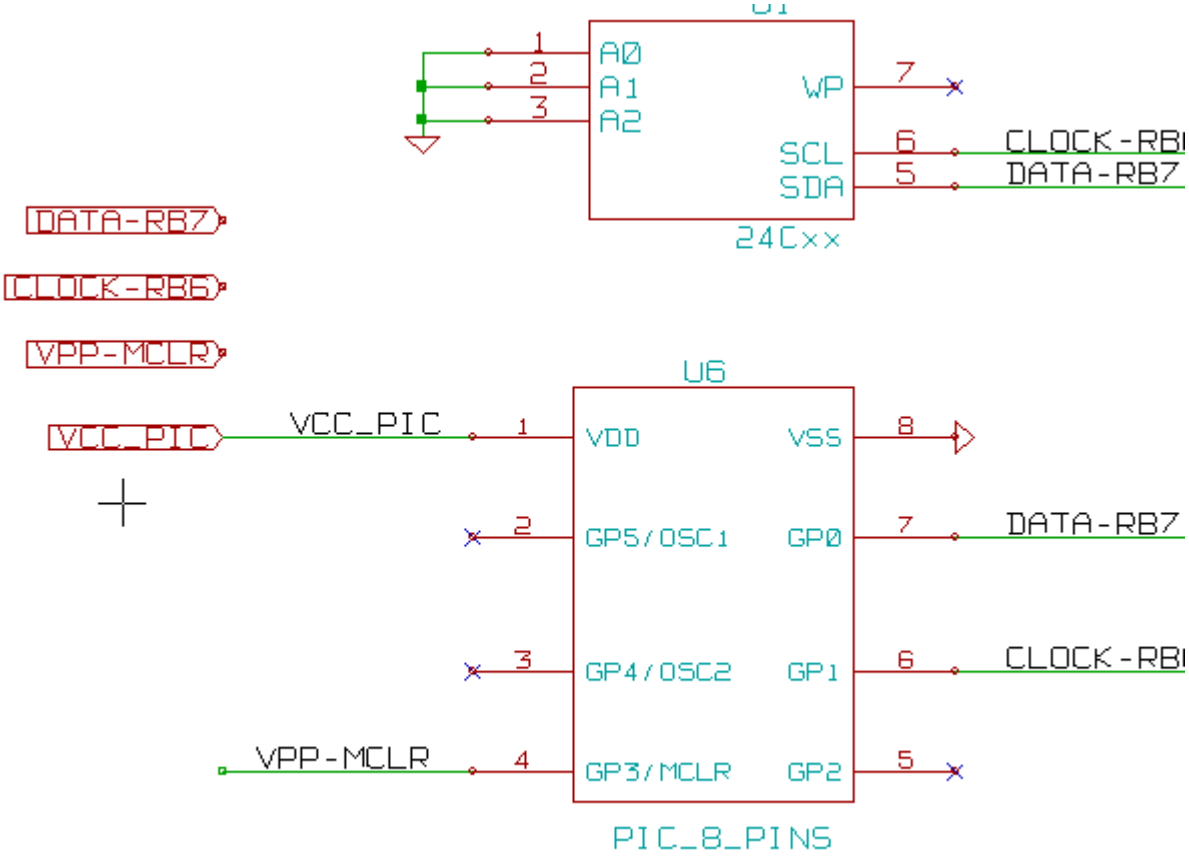


Here is the two pages, connected by global labels.

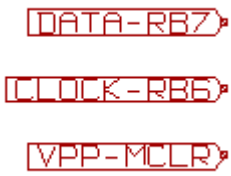
Here is the pic_programmer.sch.



Here is the pic_sockets.sch.



Look at global labels.

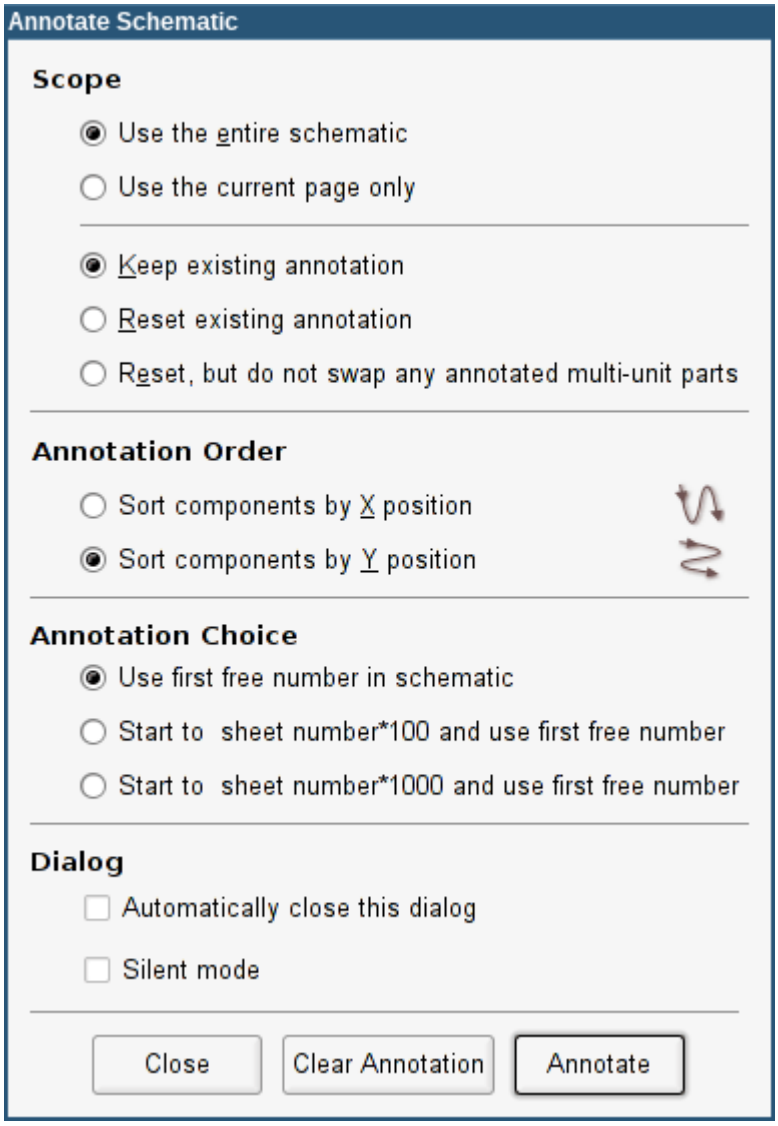


Symbol Annotation Tool

Introduction

The annotation tool allows you to automatically assign a designator to symbols in your schematic. Annotation of symbols with multiple units will assign a unique suffix to minimize the number of these symbols. The annotation tool is accessible via the icon 

. Here you find its main window.



Available annotation schemes:

- Annotate all the symbols (reset existing annotation option)
- Annotate all the symbols, but do not swap any previously annotated multi-unit parts.
- Annotate only symbols that are currently not annotated. Symbols that are not annotated will have a designator which ends with a '?' character.
- Annotate the whole hierarchy (use the entire schematic option).
- Annotate the current sheet only (use current page only option).

The ``Reset, but do not swap any annotated multi-unit parts'' option keeps all existing associations between symbols with multilple units. For example, U2A and U2B may be reannotated to U1A and U1B respectively but they will never be reannotated to U1A and U2A, nor to U2B and U2A. This is useful if you want to ensure that pin groupings are maintained.

The annotation order choice gives the method used to set the reference number inside each sheet of the hierarchy.

Except for particular cases, an automatic annotation applies to the whole project (all sheets) and to the new components, if you don’t want to modify previous annotations.

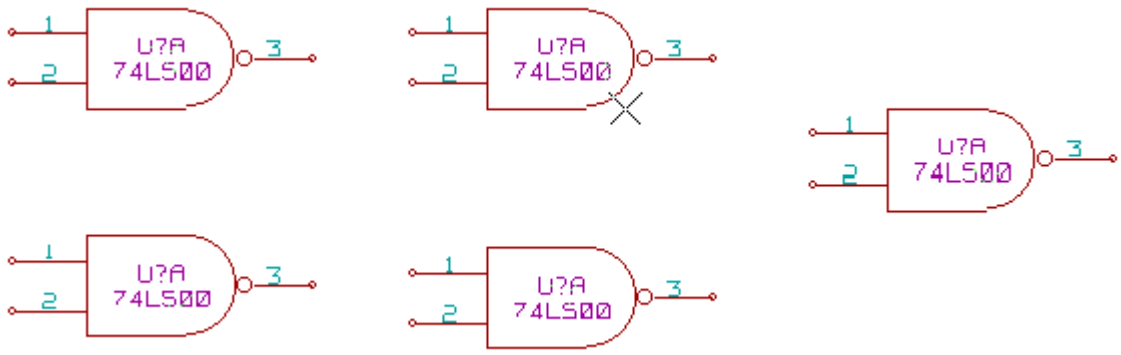
The Annotation Choice gives the method used to calculate reference:

- Use first free number in schematic: components are annotated from 1 (for each reference prefix). If a previous annotation exists, only unused numbers will be used.
- Start to sheet number*100 and use first free number: annotation start from 101 for the sheet 1, from 201 for the sheet 2, etc. If there are more than 99 items having the same reference prefix (U, R) inside the sheet 1, the annotation tool uses the number 200 and more, and annotation for sheet 2 will start from the next free number.
- Start to sheet number*1000 and use first free number. Annotation start from 1001 for the sheet 1, from 2001 for the sheet 2.

Some examples

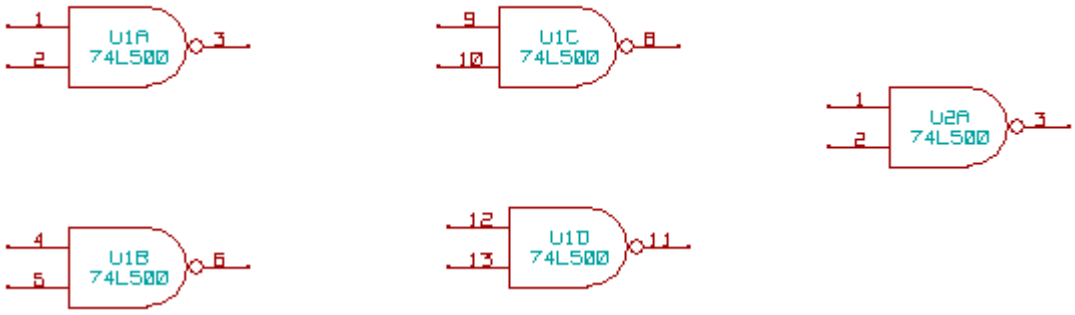
Annotation order

This example shows 5 elements placed, but not annotated.

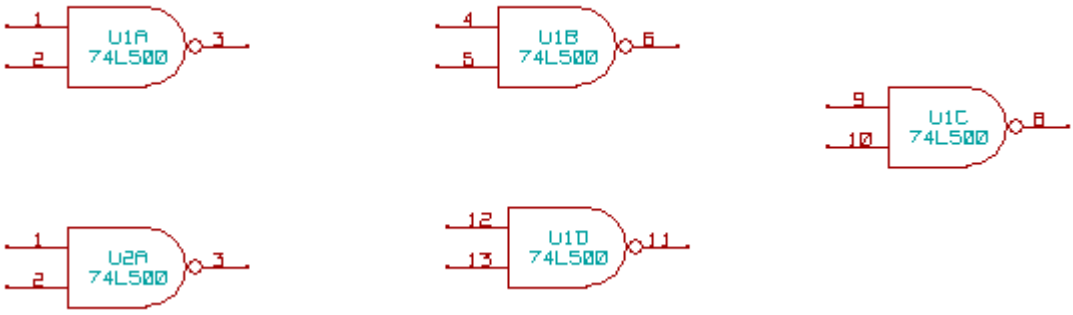


After the annotation tool Is executed, the following result is obtained.

Sort by X position.



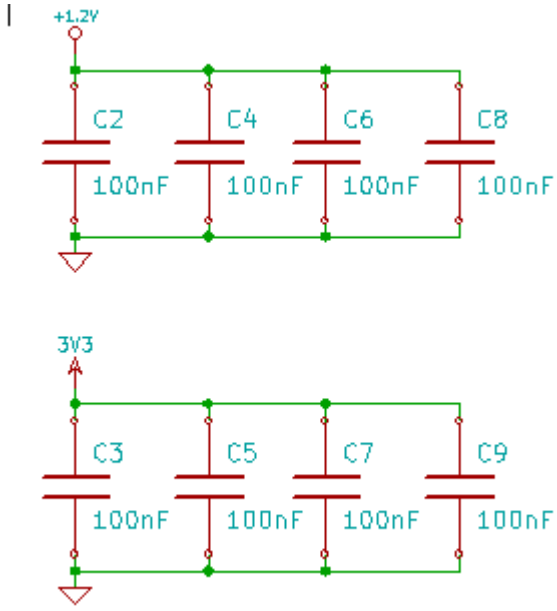
Sort by Y position.



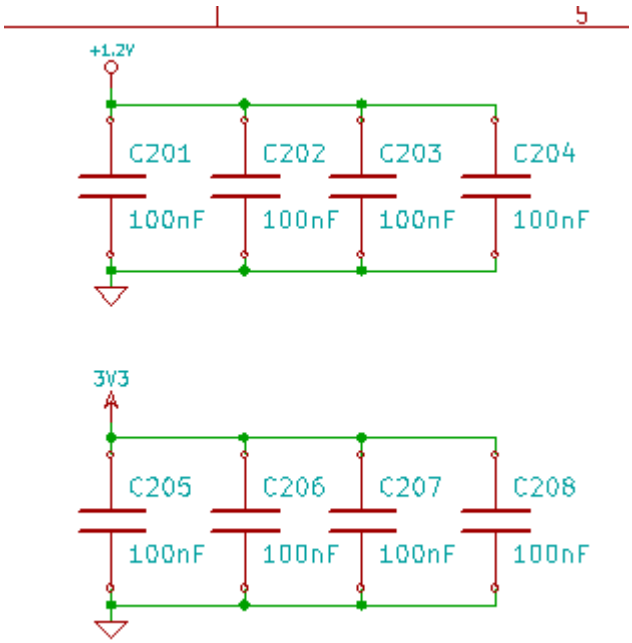
You can see that four 74LS00 gates were distributed in U1 package, and that the fifth 74LS00 has been assigned to the next, U2.

Annotation Choice

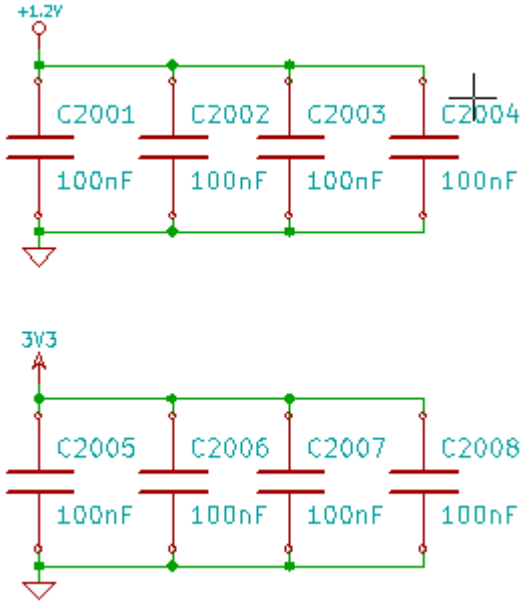
Here is an annotation in sheet 2 where the option use first free number in schematic was set.



Option start to sheet number*100 and use first free number give the following result.



The option start to sheet number*1000 and use first free number gives the following result.

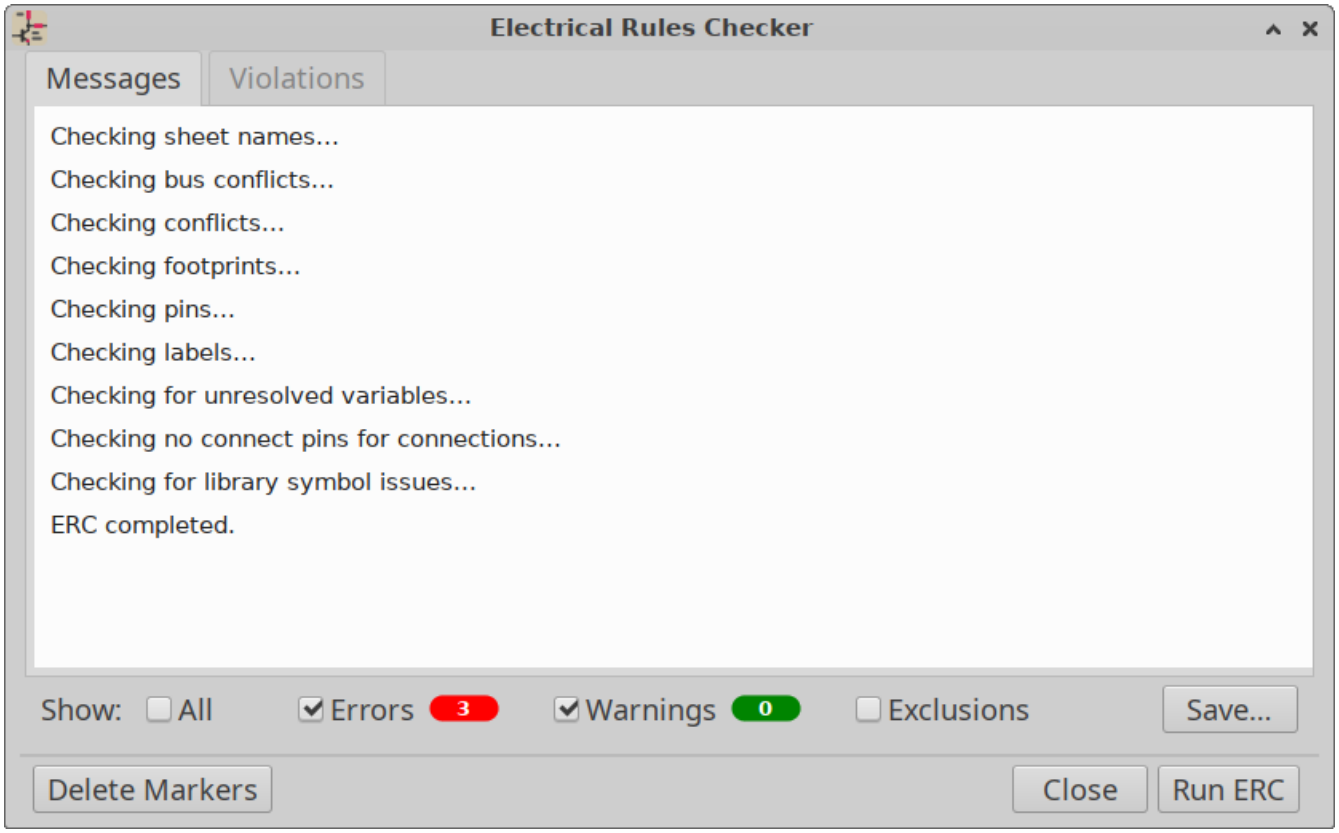


Design verification with Electrical Rules Check

Introduction

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. Naturally, an automatic check is not infallible, and the software that makes it possible to detect all design errors is not yet 100% complete. Such a check is very useful, because it allows you to detect many oversights and small errors.

In fact all detected errors must be checked and then corrected before proceeding as normal. The quality of the ERC is directly related to the care taken in declaring electrical pin properties during symbol library creation. ERC output is reported as `errors'' or` warnings".



How to use ERC

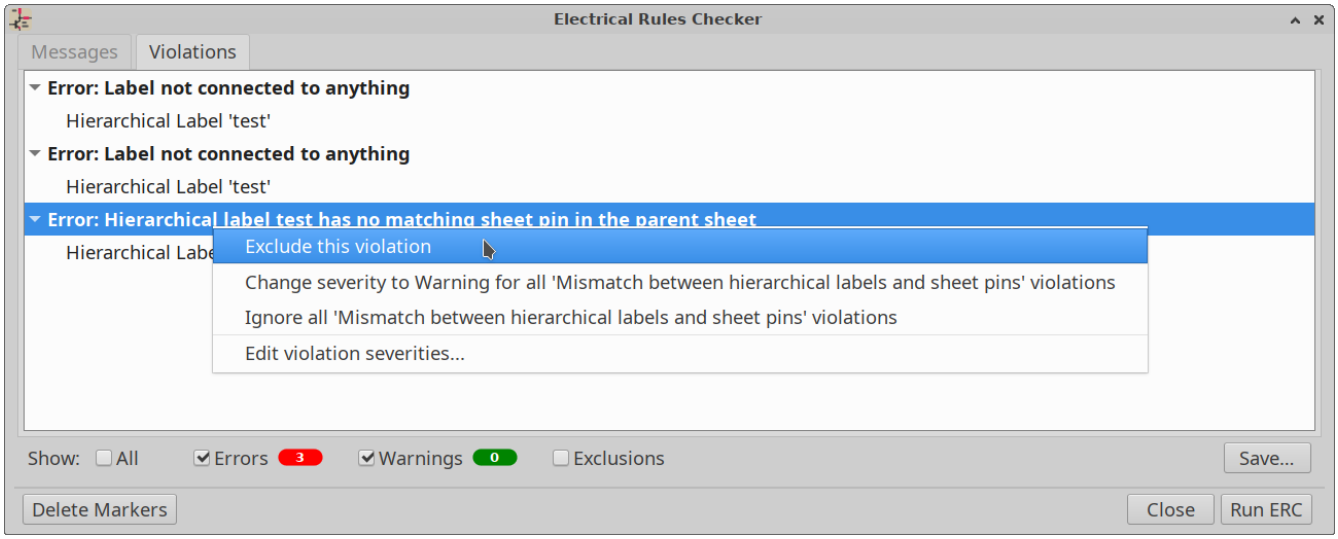
ERC can be started by clicking on the icon .

Warnings are placed on the schematic elements raising an ERC error (pins or labels).

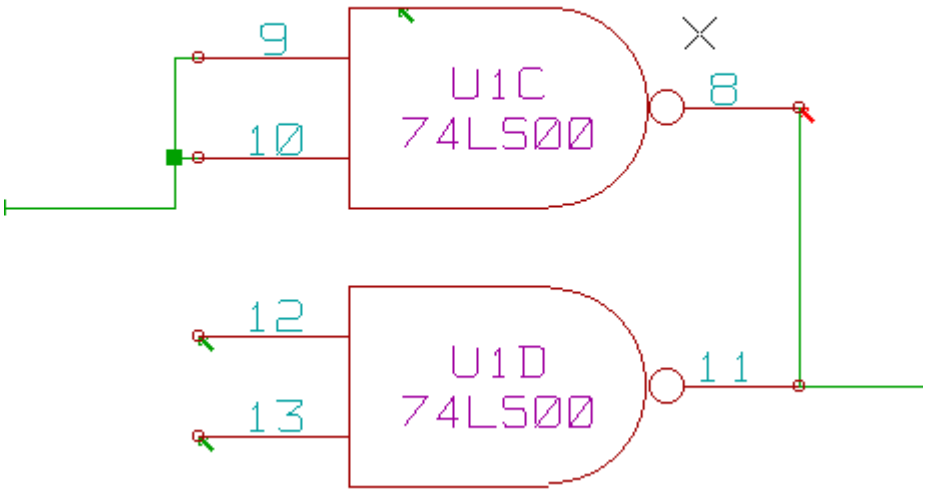
NOTE

- In this dialog window, when clicking on an error message you can jump to the corresponding marker in the schematic.
- In the schematic right-click on a marker to access the corresponding diagnostic message.

You can also delete error markers from the dialog and set specific ERC messages to be suppressed by using the right-click context menu.



Example of ERC

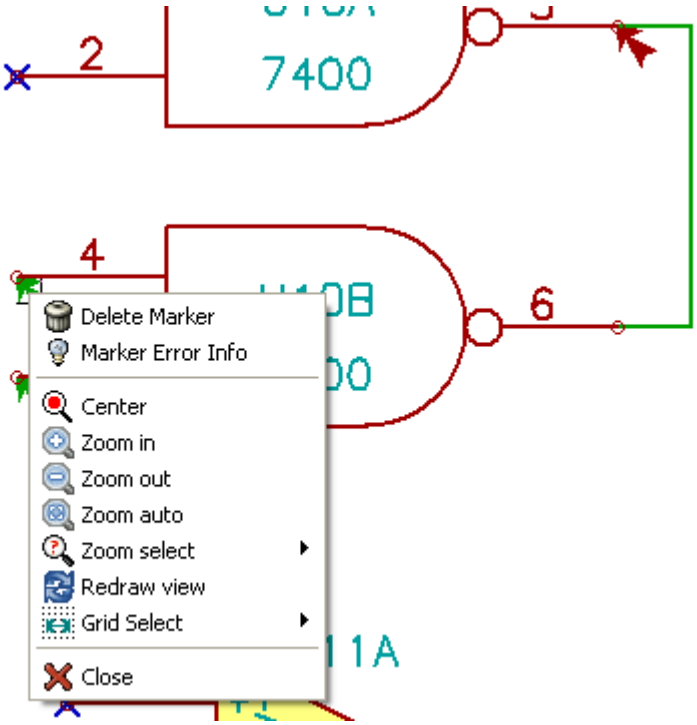


Here you can see four errors:

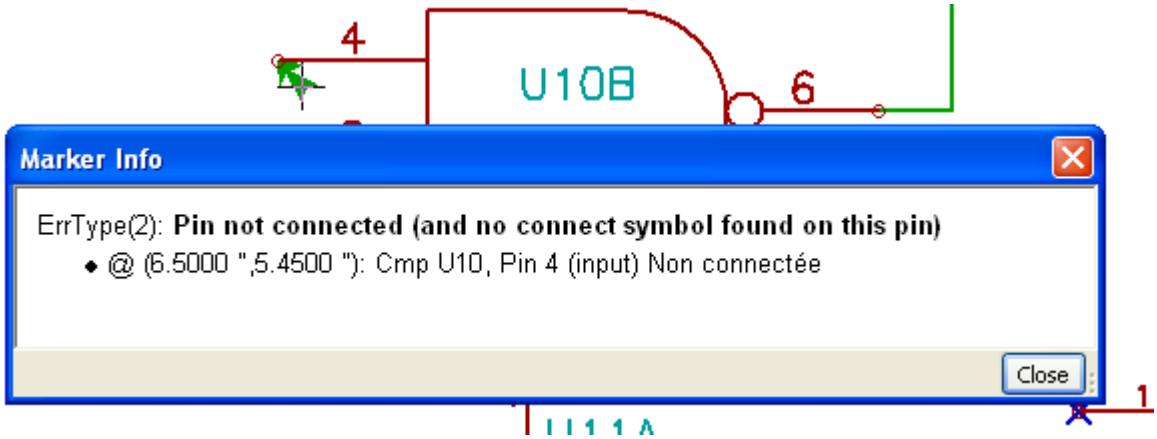
- Two outputs have been erroneously connected together (red arrow).
- Two inputs have been left unconnected (green arrow).
- There is an error on an invisible power port, power flag is missing (green arrow on the top).

Displaying diagnostics

By right-clicking on a marker the pop-up menu allows you to access the ERC marker diagnostic window.



and when clicking on Marker Error Info you can get a description of the error.

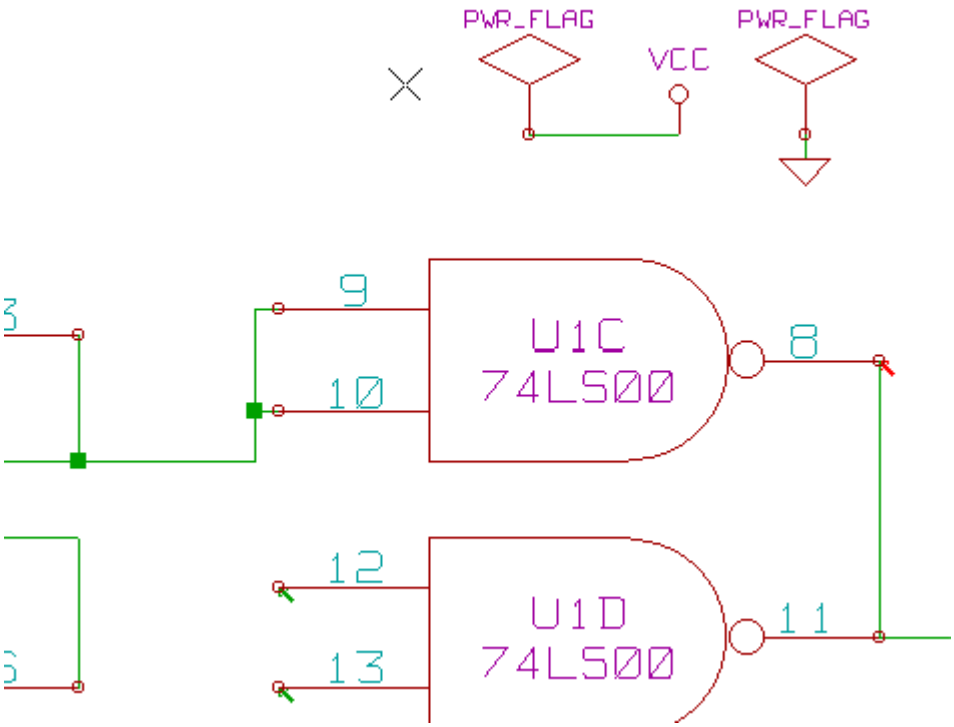


Power pins and Power flags

It is common to have an error or a warning on power pins, even though all seems normal. See example above. This happens because, in most designs, the power is provided by connectors that are not power sources (like regulator output, which is declared as Power out).

The ERC thus won't detect any Power out pin to control this wire and will declare them not driven by a power source.

To avoid this warning you have to place a "PWR_FLAG" on such a power port. Take a look at the following example:

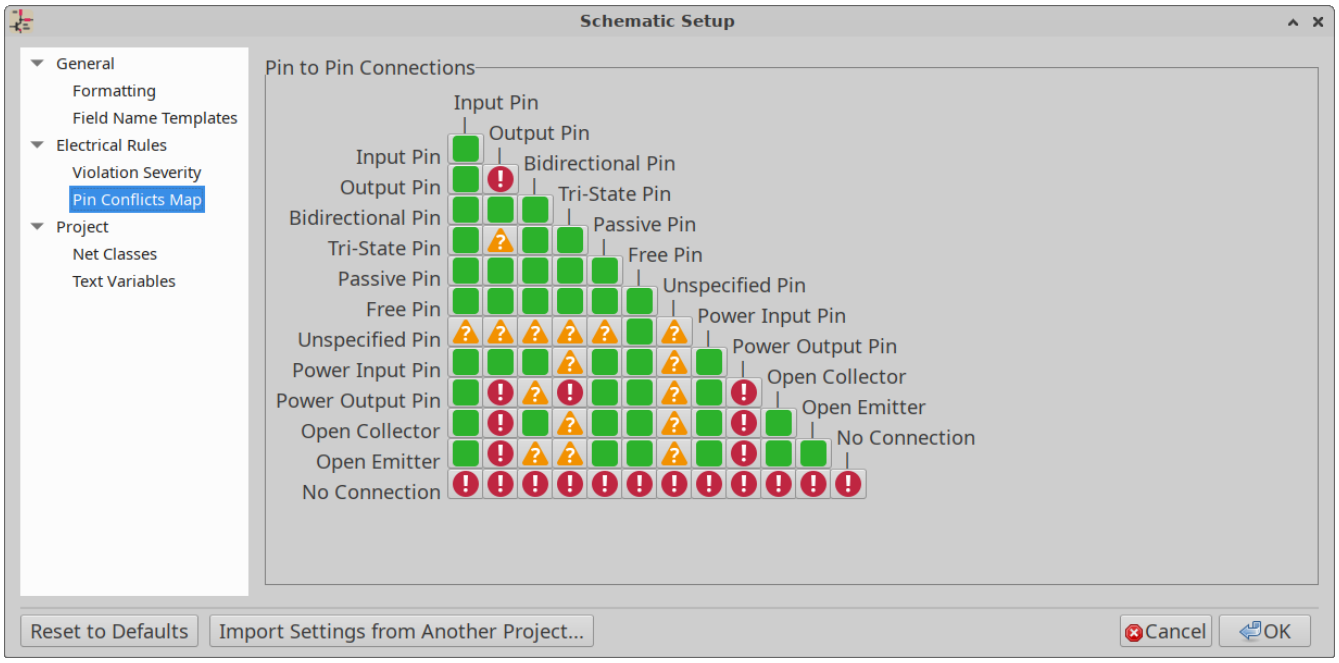


The error marker will then disappear.

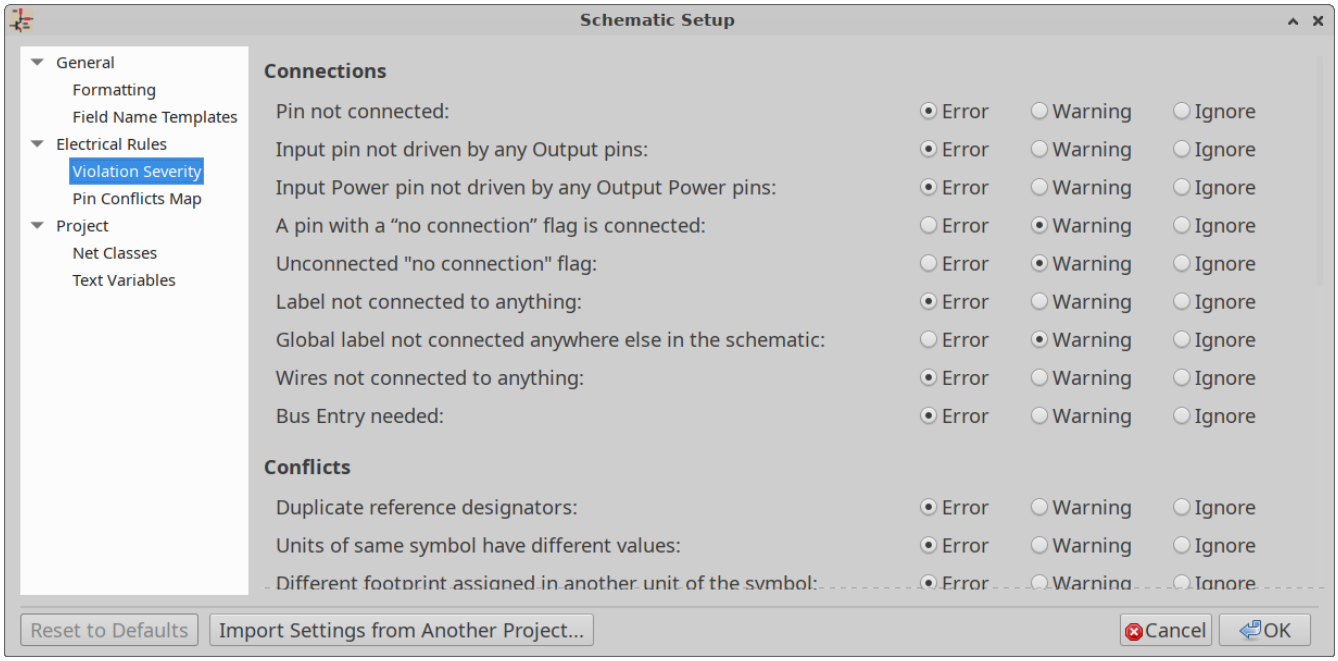
Most of the time, a PWR_FLAG must be connected to GND, because regulators have outputs declared as power out, but ground pins are never power out (the normal attribute is power in), so grounds never appear connected to a power source without a power flag symbol.

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



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



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

ERC report file

An ERC report file can be generated and saved by checking the option Write ERC report. The file extension for ERC report files is .erc. Here is an example ERC report file.

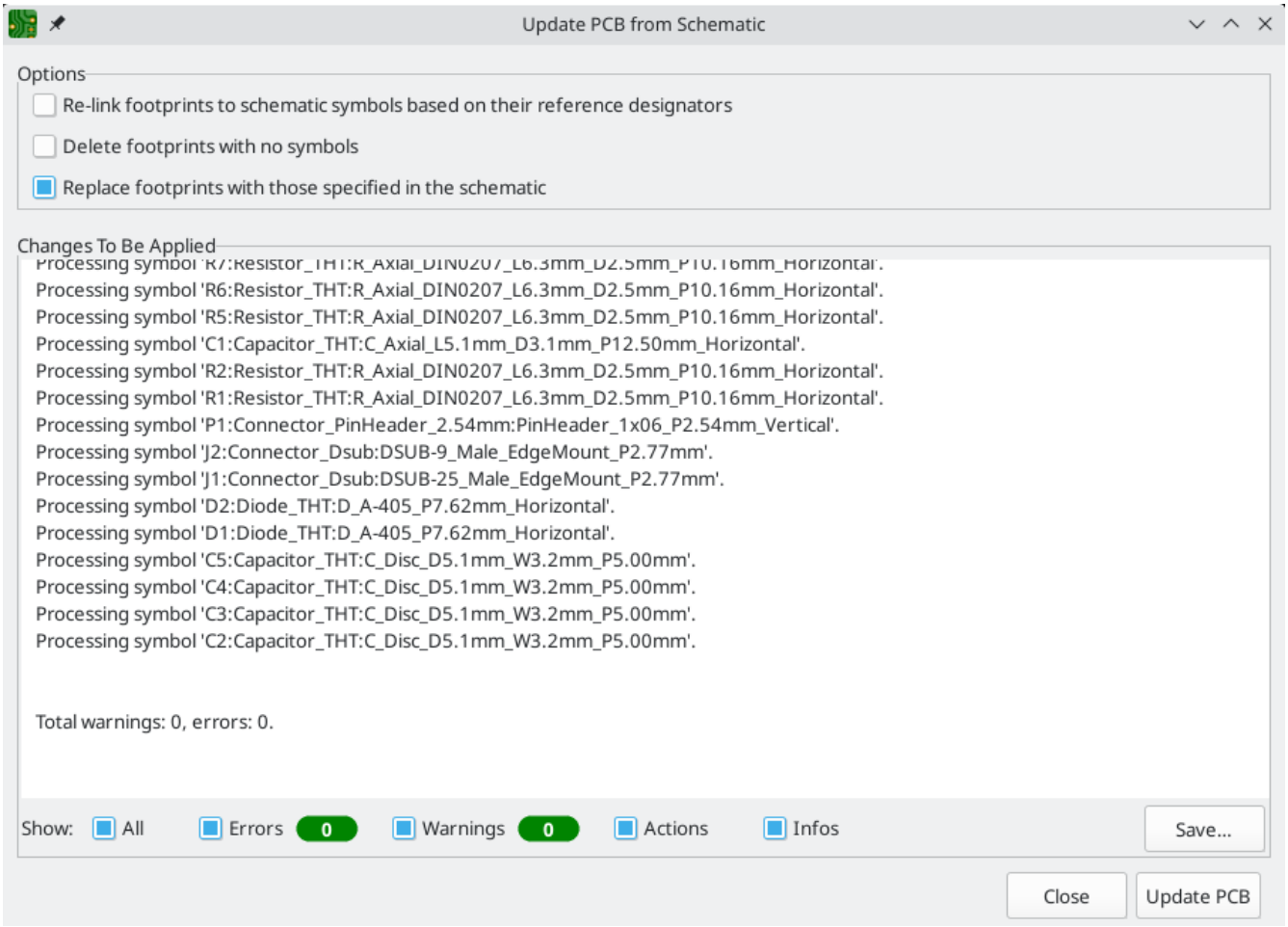
```
ERC control (4/1/1997-14:16:4)

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

>> Errors ERC: 4
```

Transfer Schematic to PCB

Overview

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

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



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

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

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

Options

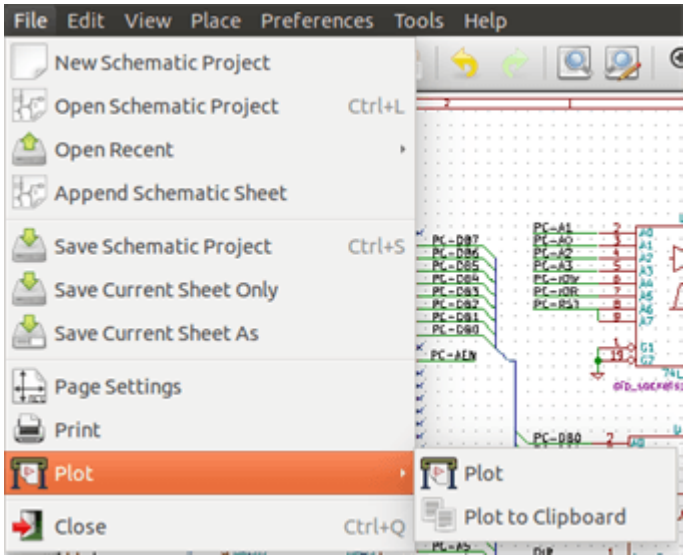
The tool has several options to control its behavior.

Option	Description
Re-link footprints to schematic symbols based on their reference designators	Footprints are normally linked to schematic symbols via a unique identifier created when the symbol is added to the schematic. A symbol's unique identifier cannot be changed. If checked, each footprint in the PCB will be re-linked to the symbol that has the same reference designator as the footprint. If unchecked, footprints and symbols will be linked by unique identifier as usual, rather than by reference designator. Each footprint's reference designator will be updated to match the reference designator of its linked symbol. This option should generally be left unchecked. It is useful for specific workflows that rely on changing the links between schematic symbols and footprints, such as refactoring a schematic for easier layout or replicating layout between identical channels of a design.
Delete footprints with no symbols	If checked, any footprint in the PCB without a corresponding symbol in the schematic will be deleted from the PCB. Footprints with the "Not in schematic" attribute will be unaffected. If unchecked, footprints without a corresponding symbol will not be deleted.
Replace footprints with those specified in the schematic	If checked, footprints in the PCB will be replaced with the footprint that is specified in the corresponding schematic symbol. If unchecked, footprints that are already in the PCB will not be changed, even if the schematic symbol is updated to specify a different footprint.

Plot and Print

Introduction

You can access both print and plot commands via the file menu.



The suported output formats are Postscript, PDF, SVG, DXF and HPGL. You can also directly print to your printer.

Common printing commands

Plot Current Page

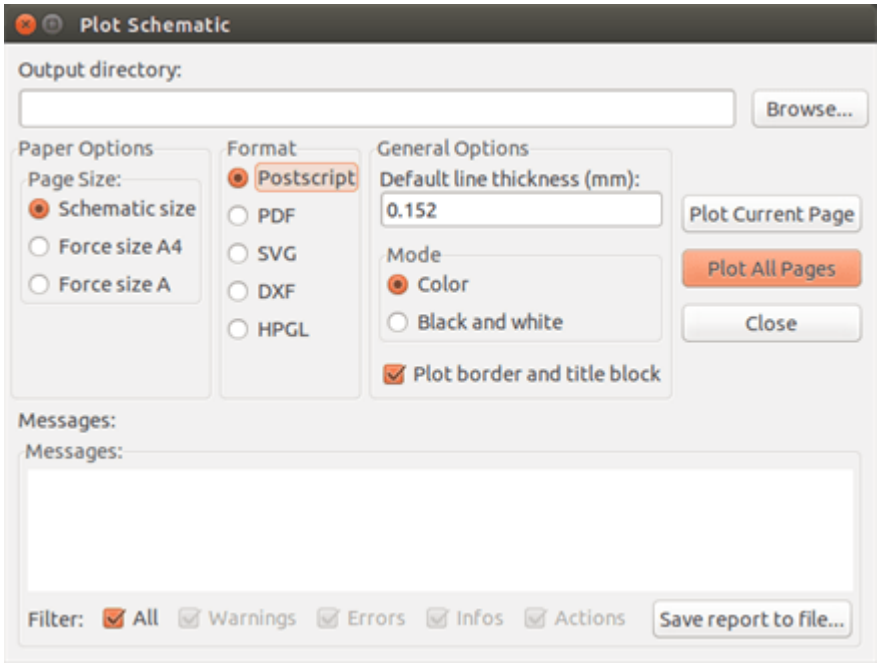
prints one file for the current sheet only.

Plot All Pages

allows you to plot the whole hierarchy (one print file is generated for each sheet).

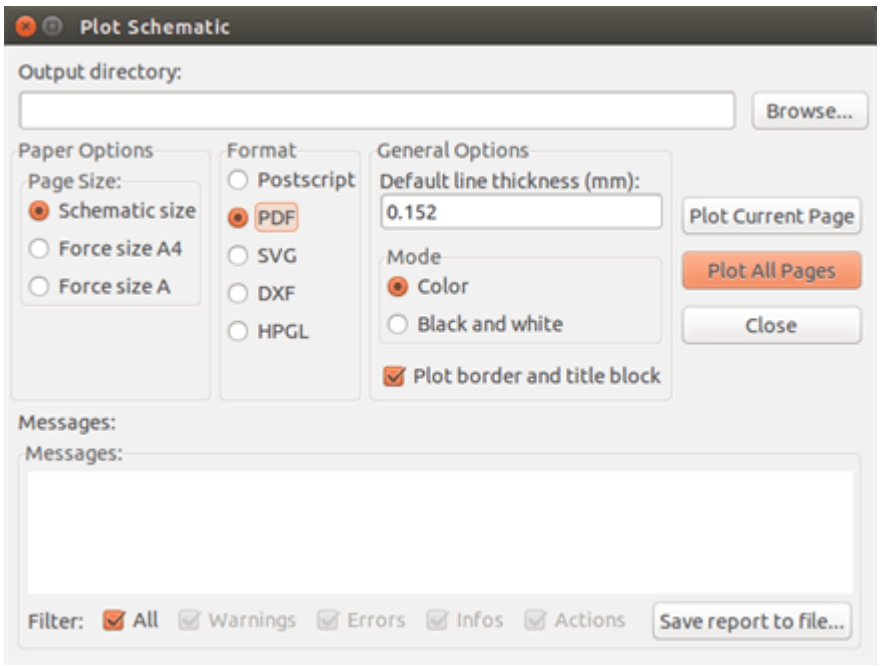
Plot in Postscript

This command allows you to create PostScript files.



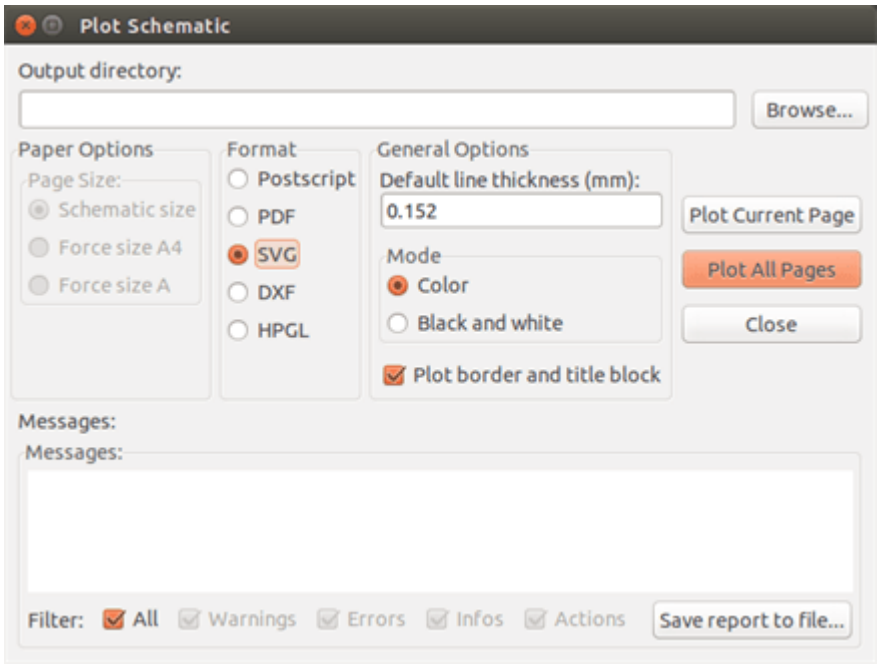
The file name is the sheet name with an extension .ps. You can disable the option "Plot border and title block". This is useful if you want to create a postscript file for encapsulation (format .eps) often used to insert a diagram in a word processing software. The message window displays the file names created.

Plot in PDF



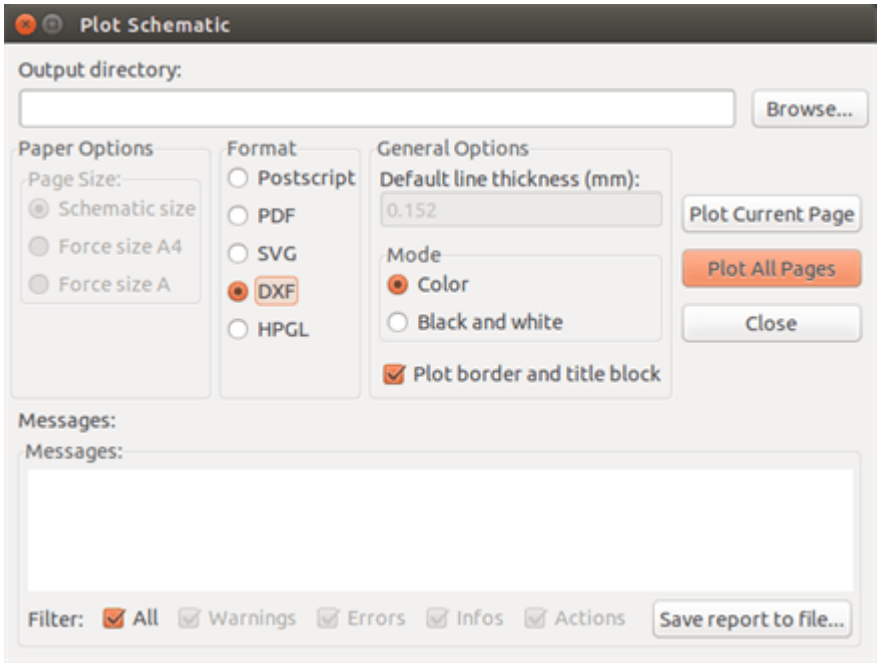
Allows you to create plot files using the format PDF. The file name is the sheet name with an extension .pdf.

Plot in SVG



Allows you to create plot files using the vectored format SVG. The file name is the sheet name with an extension .svg.

Plot in DXF



Allows you to create plot files using the format DXF. The file name is the sheet name with an extension .dxf.

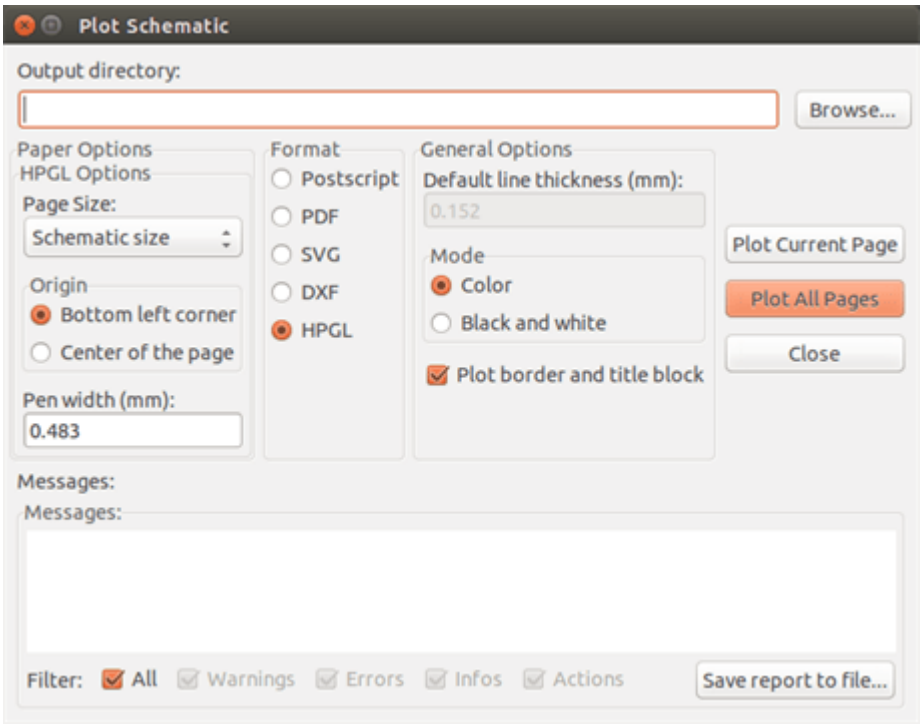
Plot in HPGL

This command allows you to create an HPGL file. In this format you can define:

- Page size.
- Origin.

Pen width (in mm).

The plotter setup dialog window looks like the following:



The output file name will be the sheet name plus the extension .plt.

Sheet size selection

Sheet size is normally checked. In this case, the sheet size defined in the title block menu will be used and the chosen scale will be 1. If a different sheet size is selected (A4 with A0, or A with E), the scale is automatically adjusted to fill the page.

Offset adjustments

For all standard dimensions, you can adjust the offsets to center the drawing as accurately as possible. Because plotters have an origin point at the center or at the lower left corner of the sheet, it is necessary to be able to introduce an offset in order to plot properly.

Generally speaking:

- For plotters having their origin point at the center of the sheet the offset must be negative and set at half of the sheet dimension.
- For plotters having their origin point at the lower left corner of the sheet the offset must be set to 0.

To set an offset:

- Select sheet size.
- Set offset X and offset Y.
- Click on accept offset.

Print on paper

This command, available via the icon , allows you to visualize and generate design files for the standard printer.

The "Print sheet reference and title block" option enables or disables sheet references and title block.

The "Print in black and white" option sets printing in monochrome. This option is generally necessary if you use a black and white laser printer, because colors are printed into half-tones that are often not so readable.

Symbol Editor

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

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.
- Fields such as references, values, corresponding footprint names for PCB design, 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.

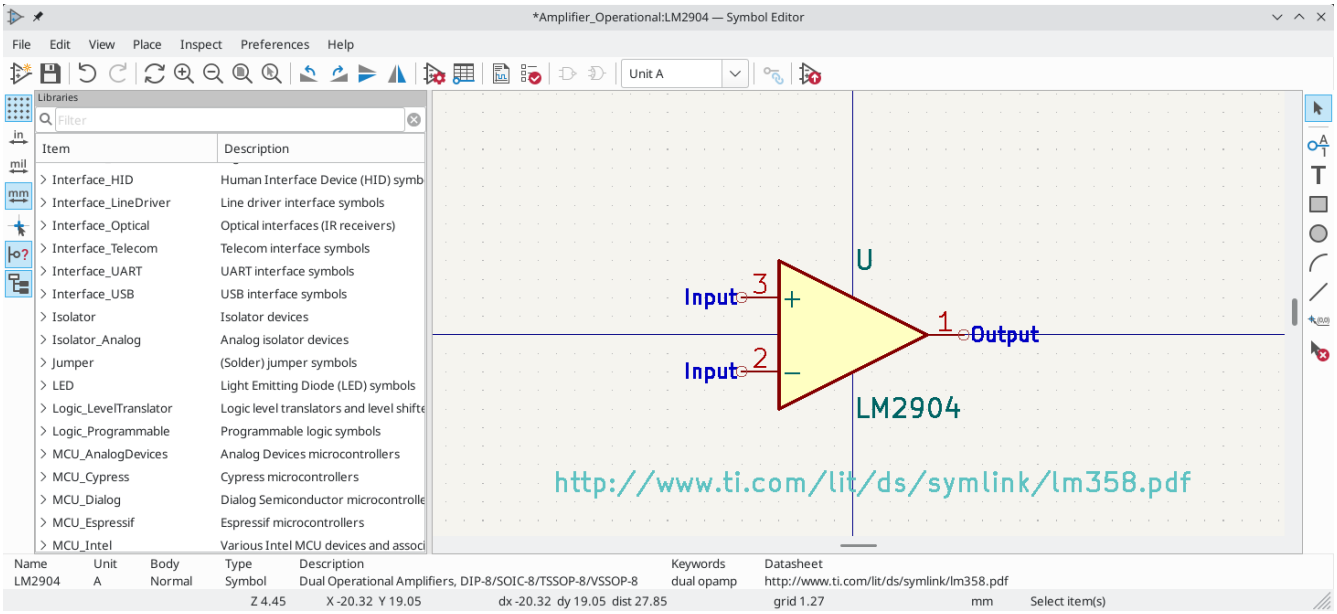
Proper symbol designing requires:

- 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).
- Designing its symbolic representation using lines, rectangles, circles, polygons and text.
- Adding pins by carefully defining each pin's graphical elements, name, number, and electrical property (input, output, tri-state, power port, etc.).
- Determining if the symbol should be derived from another symbol with the same graphical design and pin definition.
- Adding optional fields such as the name of the footprint used by the PCB design software and/or defining their visibility.
- Documenting the symbol by adding a description string and links to data sheets, etc.
- Saving it in the desired library.

Symbol Library Editor Overview

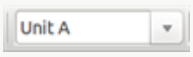
The symbol library editor main window is shown below. It consists of three tool bars for quick access to common features and a symbol viewing/editing area. Not all commands are available on the tool bars but

can be accessed using the menus.



Main Toolbar

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.

	Create a new symbol in the selected library.
	Save the currently selected library. All modified symbols in the library will be saved.
	Undo last edit.
	Redo last undo.
	Refresh display.
	Zoom in.
	Zoom out.
	Zoom to fit symbol in display.
	Zoom to fit selection.
	Rotate counter-clockwise.
	Rotate clockwise.
	Mirror horizontally.
	Mirror vertically.
	Edit the current symbol properties.
	Edit the symbol's pins in a tablular interface.
	Open the symbol's datasheet. The button will be disabled if no datasheet is defined for the current symbol.
	Test the current symbol for design errors.
	Select the normal body style. The button is disabled if the current symbol does not have an alternate body style.
	Select the alternate body style. The button is disabled if the current symbol does not have an alternate body style.
	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.
	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 interchangable units and cannot be enabled for symbols with only one unit.

Element Toolbar

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.

	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.
	Pin tool. Left-click to add a new pin.
	Graphical text tool. Left-click to add a new graphical text item.
	Rectangle tool. Left-click to begin drawing the first corner of a graphical rectangle. Left-click again to place the opposite corner of the rectangle.
	Circle tool. Left-click to begin drawing a new graphical circle from the center. Left-click again to define the radius of the circle.
	Arc tool. Left-click to begin drawing a new graphical arc item from the first arc end point. Left-click again to define the second arc end point. Adjust the radius by dragging the arc center point.
	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.
	Anchor tool. Left-click to set the anchor position of the symbol.
	Delete tool. Left-click to delete an object from the current symbol.

Options Toolbar

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

	Toggle grid visibility on and off.
	Set units to inches.
	Set units to mils (0.001 inch).
	Set units to millimeters.
	Toggle full screen cursor on and off.
	Toggle display of pin electrical types.
	Toggle display of libraries and symbols.

Library Selection and Maintenance

The selection of the current library is possible via the  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

Symbol Selection

Clicking the  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

After modification, a symbol can be saved in the current library or a different library.

To save the modified symbol in the current library, click the  icon. 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

Create a New Symbol

A new symbol can be created by clicking the  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

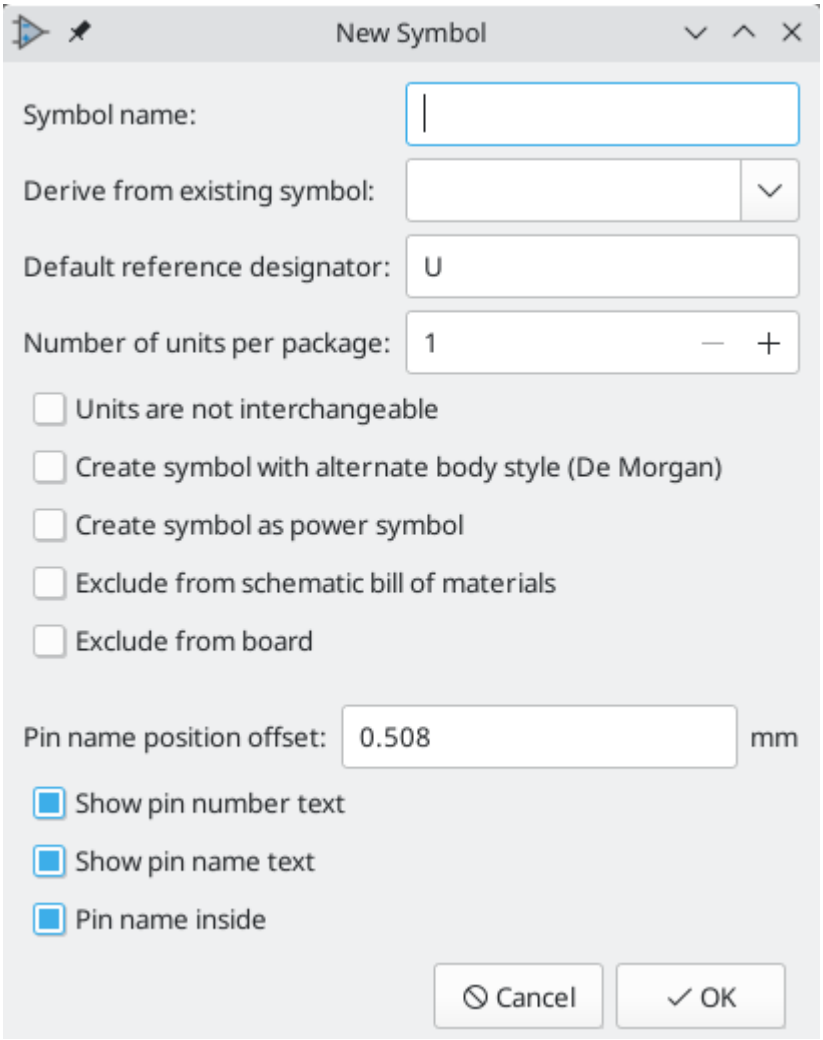
footprint and they are not added to the PCB, and they are not included in the bill of materials.

- Whether the symbol should be excluded from the bill of materials.
- Whether the symbol should be excluded from the PCB.

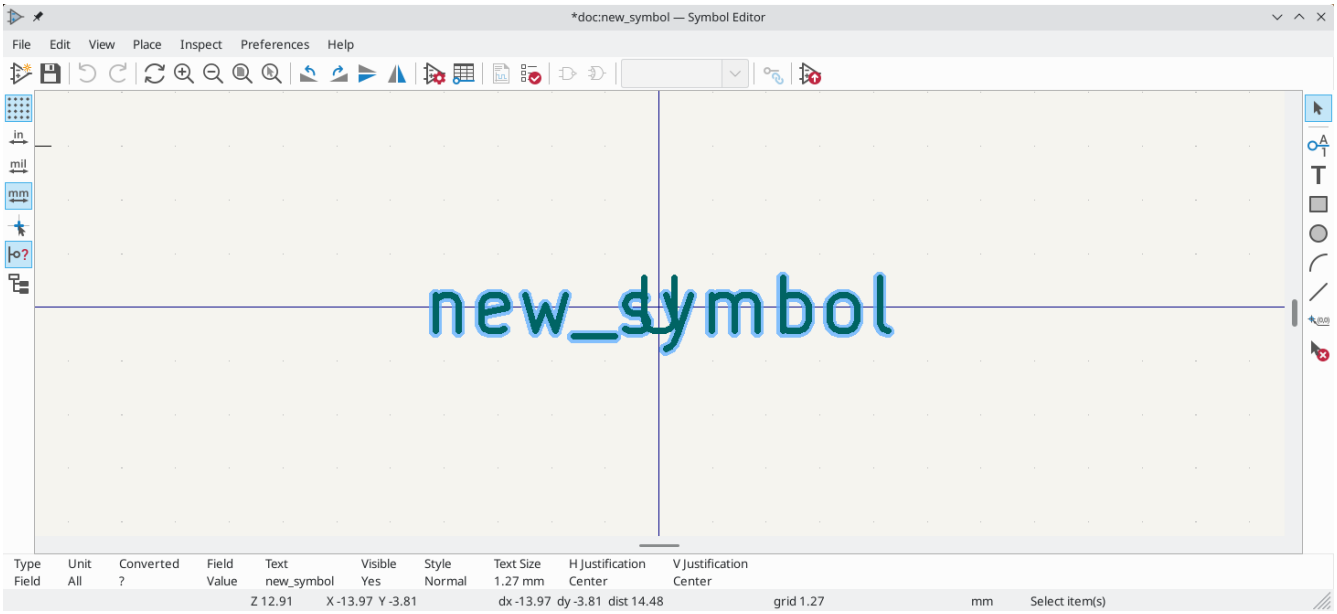
There are also several graphical options.

- 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 window.



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



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 icon and clicking on the new desired anchor position.

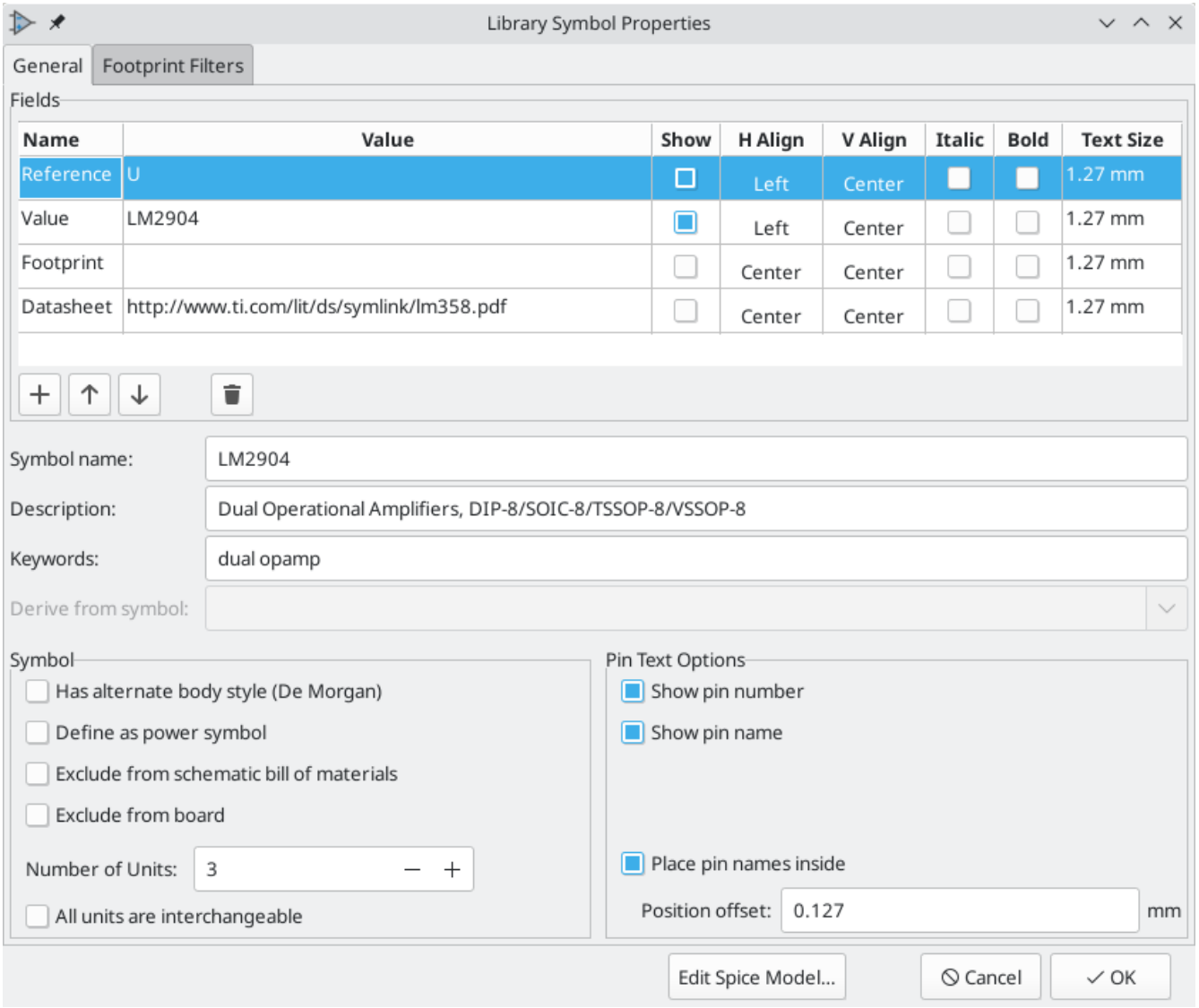
Create a Symbol from Another Symbol

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

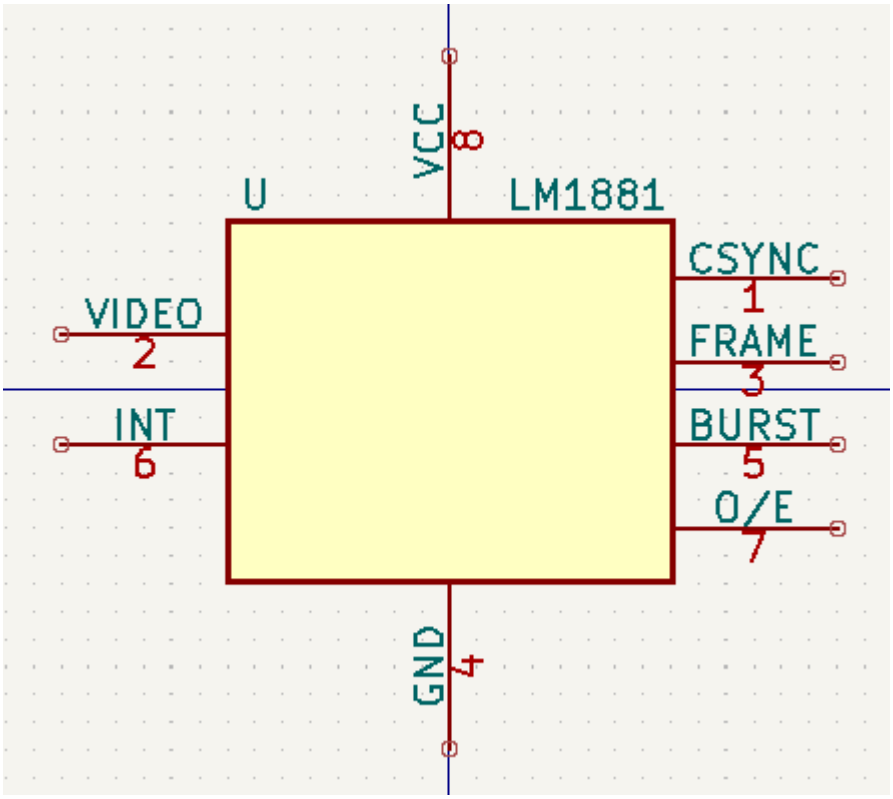
Symbol properties are set when the symbol is created but they can be modified at any point. To change the symbol properties, click on the icon to show the dialog below.



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.

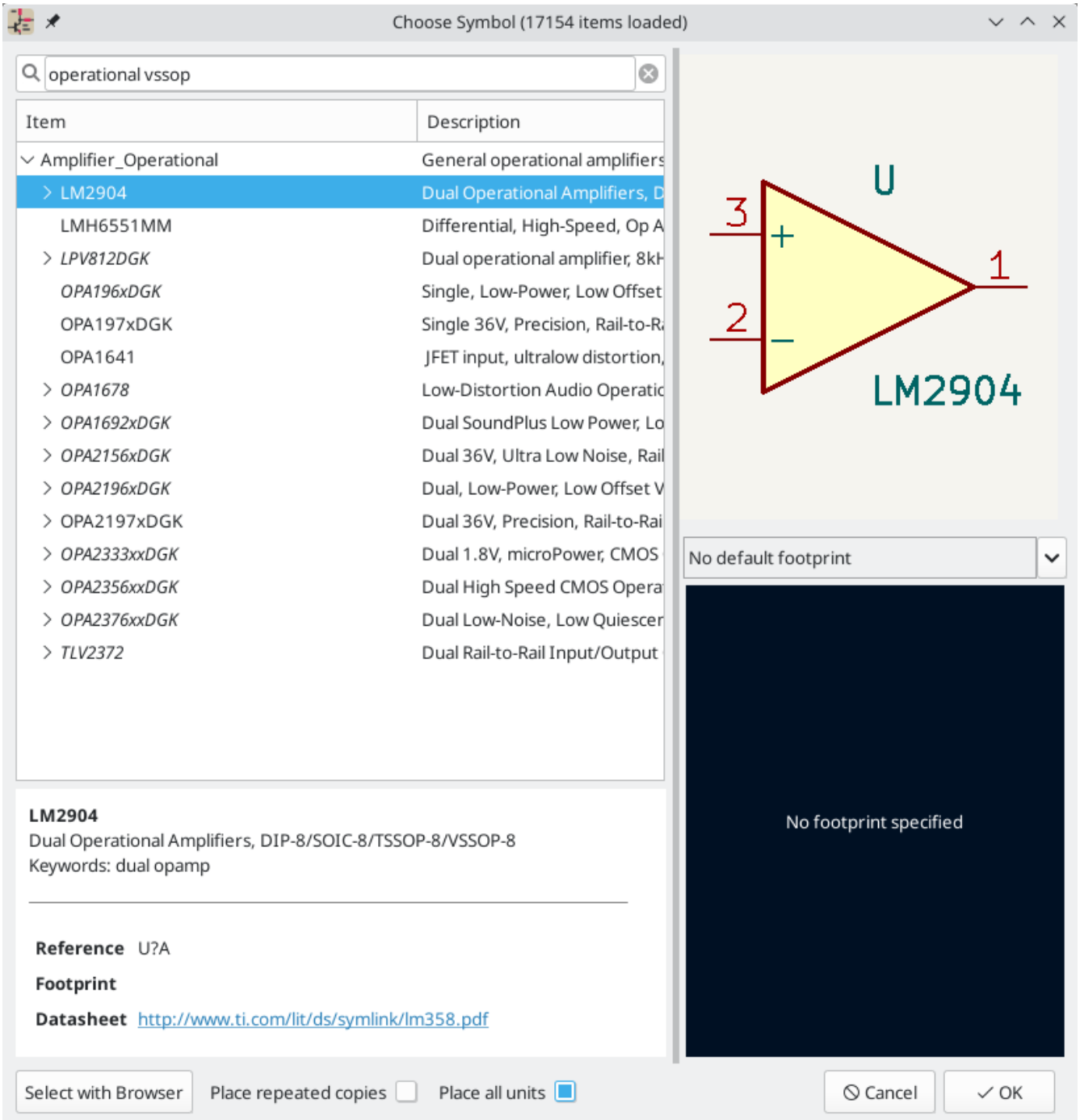
The example below shows a symbol with the "Place pin name inside" option unchecked. Notice the position of the names and pin numbers.



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.



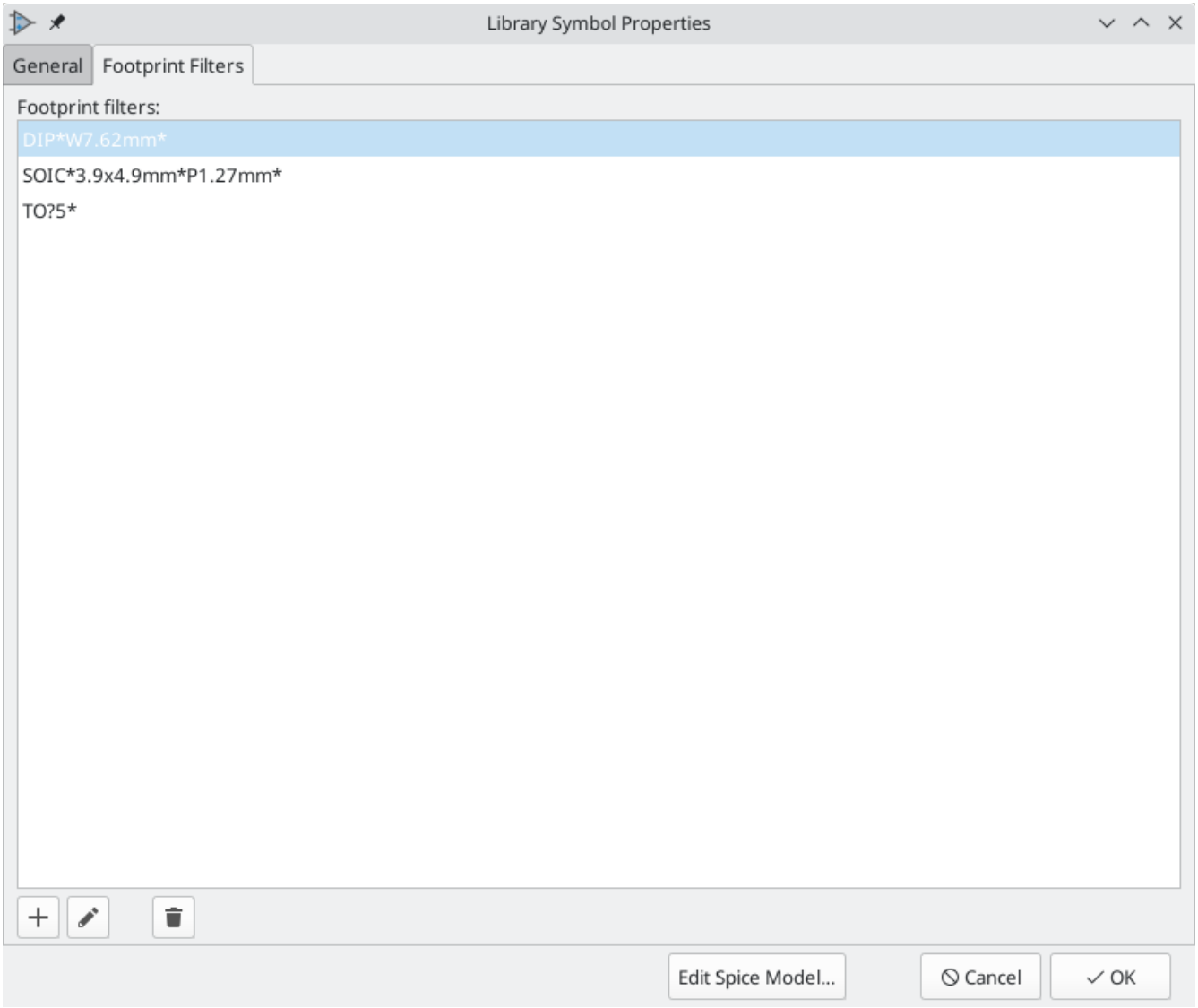
A symbol's name, description, and keywords are all used when searching for symbols in the Symbol Editor and Add a Symbol dialog. The description and keywords are displayed in the Symbol Library Browser and Add a Symbol dialog.

Footprint Filters

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.

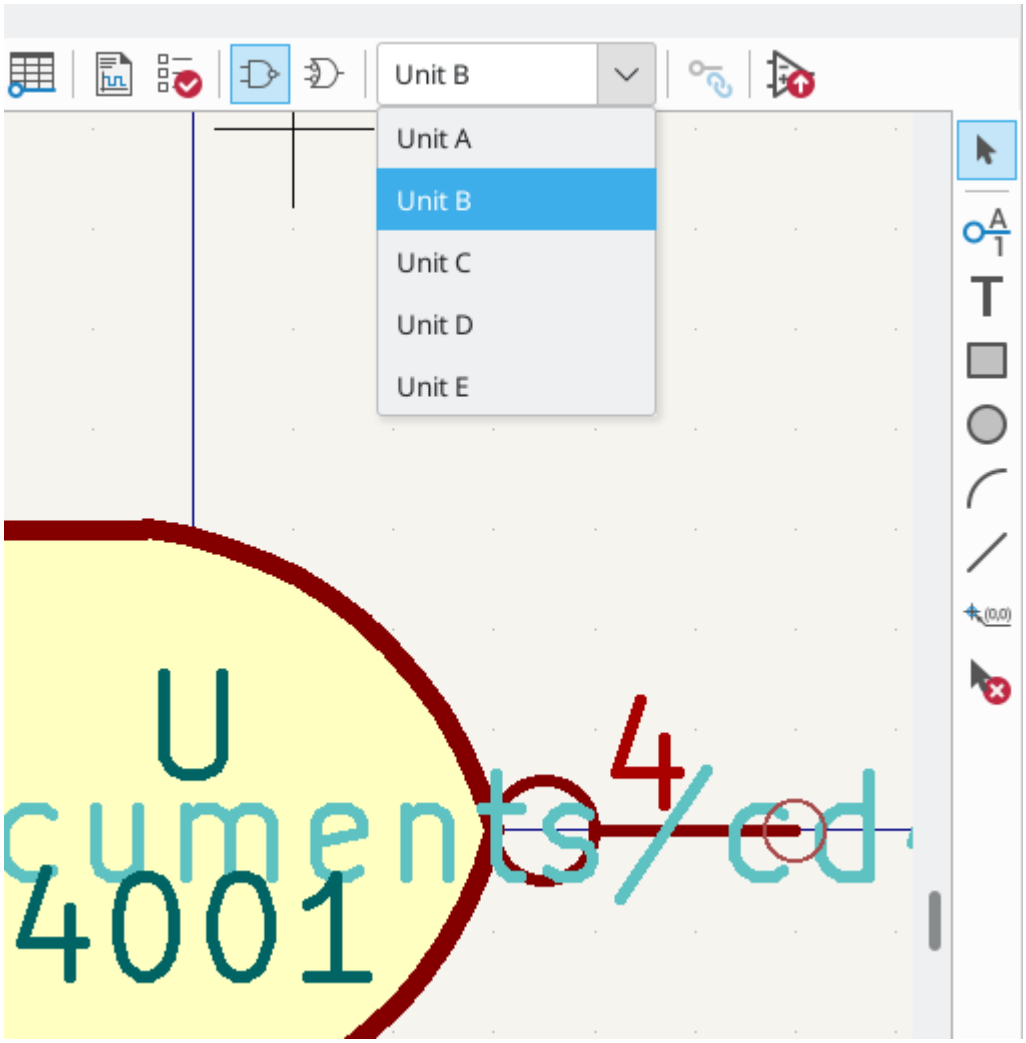
Filters can use wildcards: `*` matches any number of characters, including zero, and `?` matches zero or one characters. For example, `SOIC-*` would match the `SOIC-8_3.9x4.9mm_P1.27mm` footprint as well as any other footprint beginning with `SOIC-`. The filter `SOT?23` matches `SOT23` as well as `SOT-23`.



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 icon.

To edit the alternate representation, click on the icon. Use the 74LS00 dropdown shown below to select the unit you wish to edit.



Graphical Elements

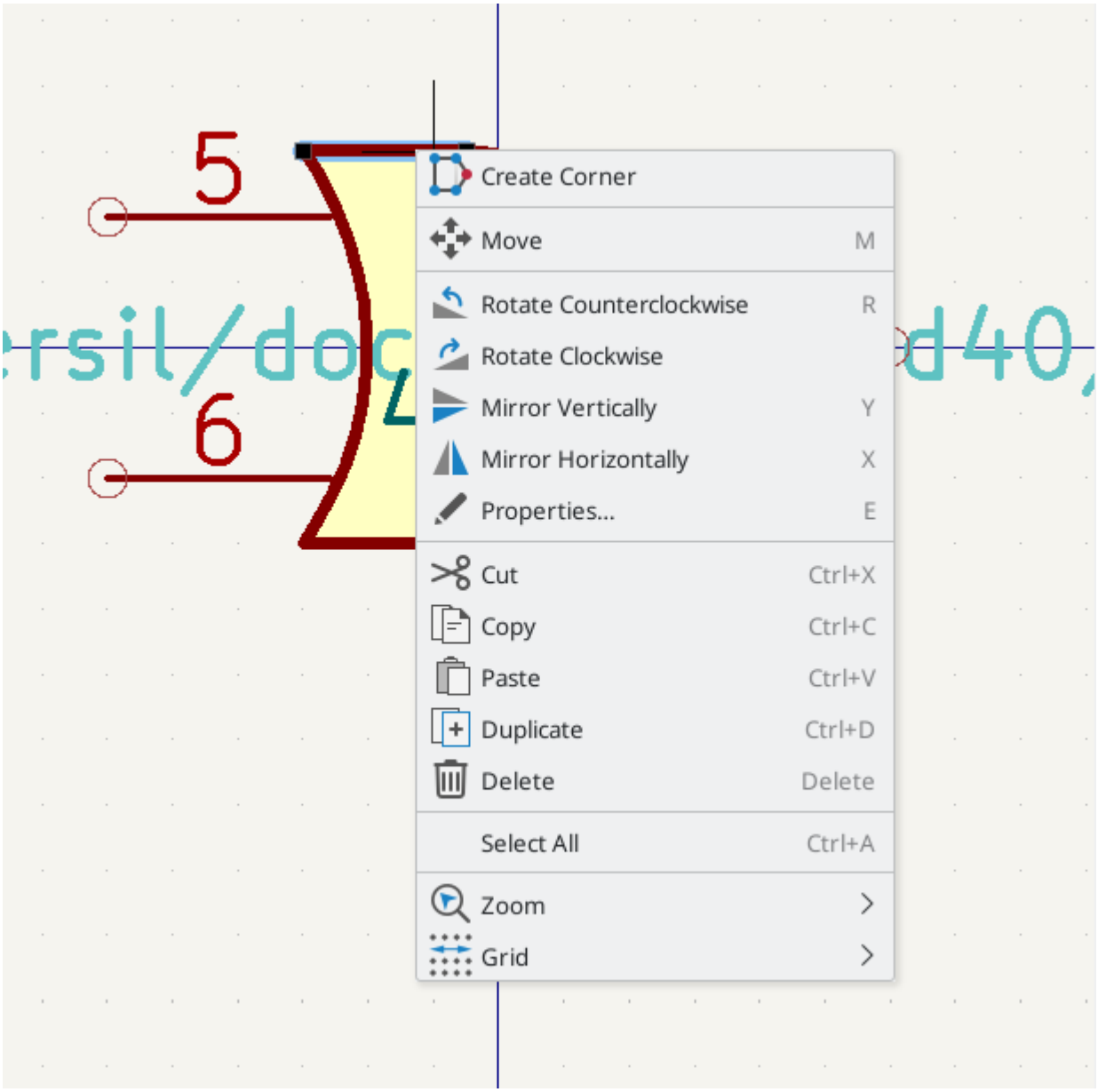
Graphical elements create the visual representation of a symbol and contain no electrical connection information. Graphical elements are created with the following tools:

- Lines and polygons defined by start and end points.
- Rectangles defined by two diagonal corners.
- Circles defined by the center and radius.
- Arcs defined by the starting and ending point of the arc and its center. An arc goes from 0° to 180°.

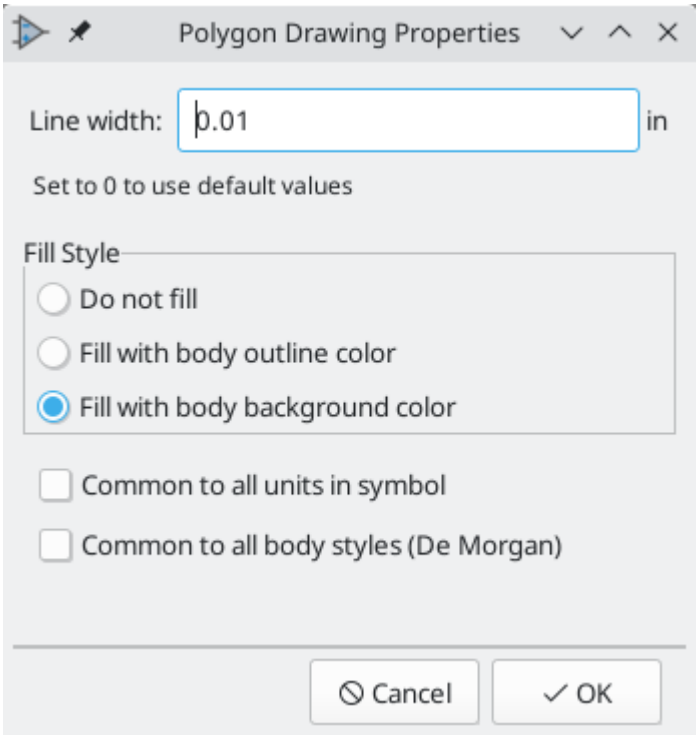
The vertical toolbar on the right hand side of the main window allows you to place all of the graphical elements required to design the representation of a symbol.

Graphical Element Membership

Each graphic element (line, arc, circle, etc.) can be defined as common to all units and/or body styles or specific to a given unit and/or body style. Element options can be quickly accessed by right-clicking on the element to display the context menu for the selected element. Below is the context menu for a line element.



You can also double-left-click on an element to modify its properties. Below is the properties dialog for a polygon element.



The properties of a graphic element are:

- "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

The T 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

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.

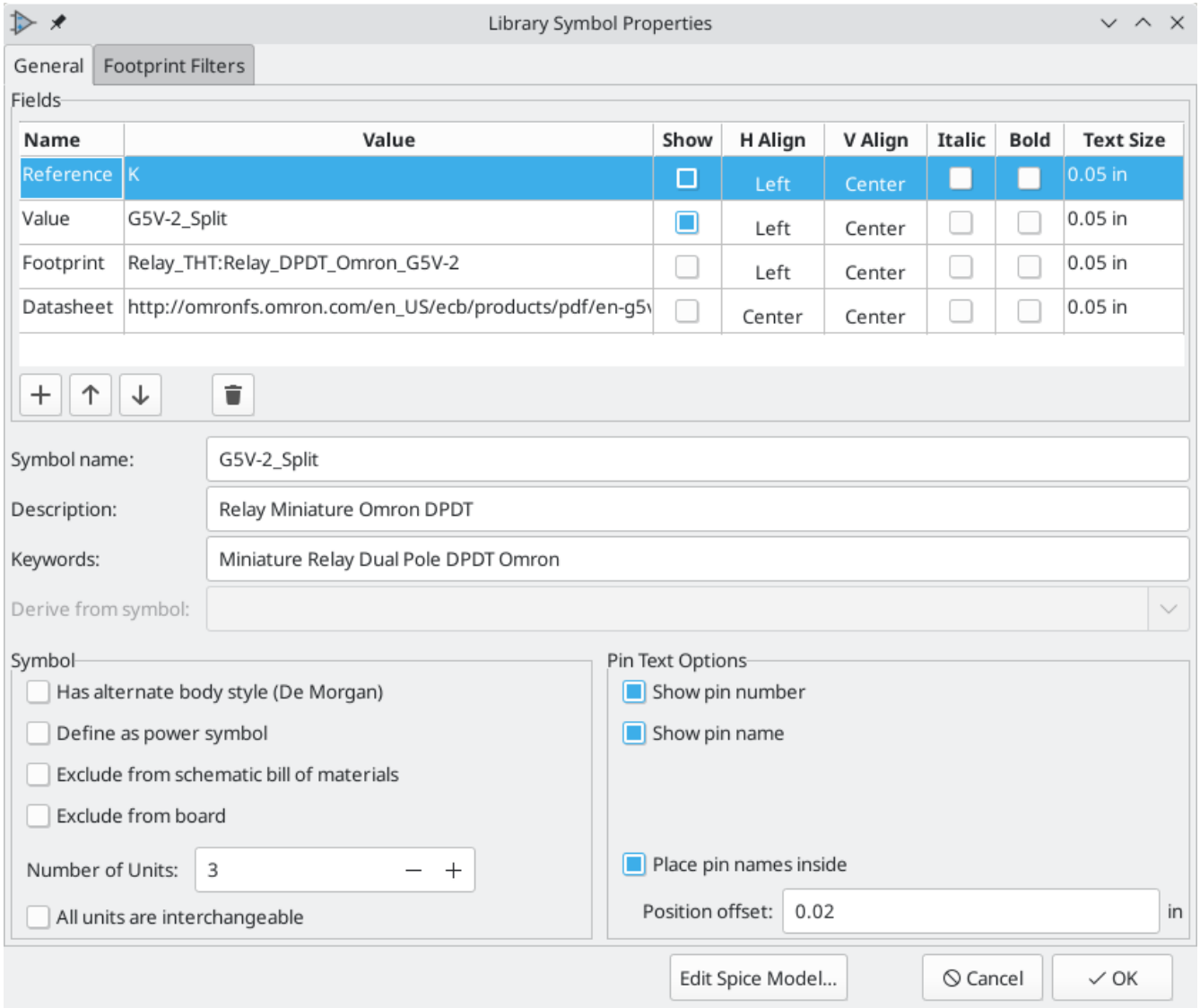
By default, pins are specific to a unit and body style. When a pin is common to all units or all body styles, it only needs to be created once. This is also the case for the body style graphic shapes and text, which may be

common to each unit, but typically are specific to each body style).

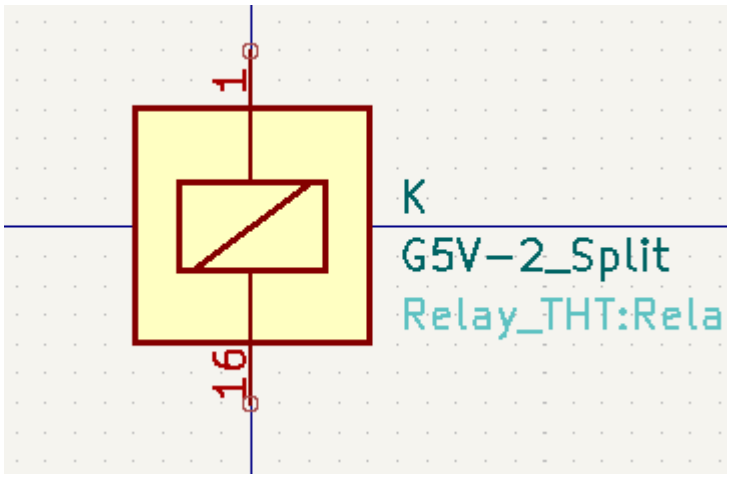
Example of a Symbol With Multiple Noninterchangeable Units

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.

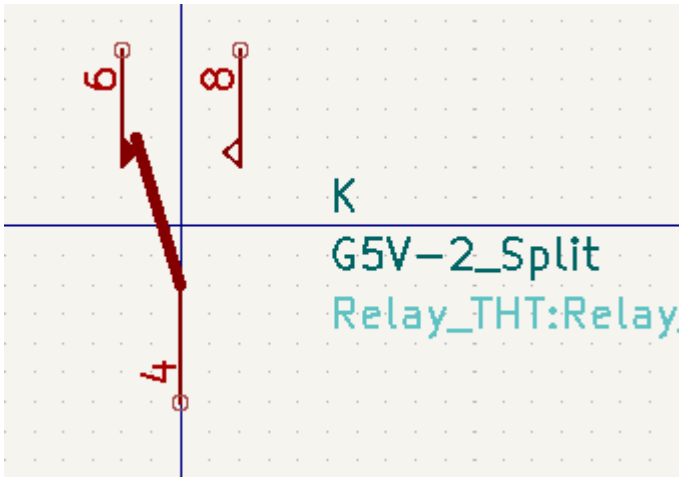
The three units are not all the same, so "All units are interchangeable" should be deselected in the Symbol Properties dialog. Alternatively, this option could have been specified when the symbol was initially created.



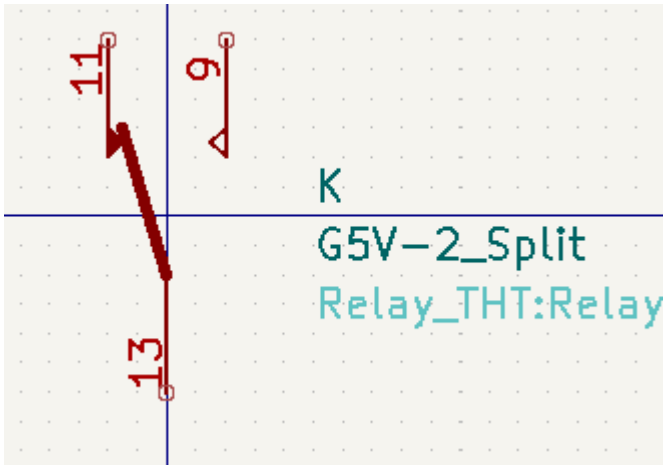
Unit A



Unit B



Unit C



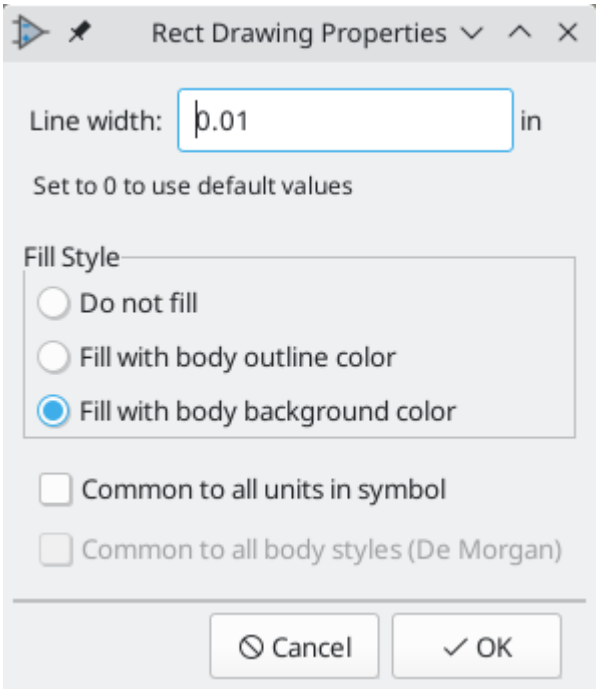
Unit A does not have the same symbol and pin layout as Units B and C, so the units are not interchangeable.

NOTE

"Synchronized Pins Edit Mode" can be enabled by clicking the 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

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.



Pin Creation and Editing

You can click on the 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

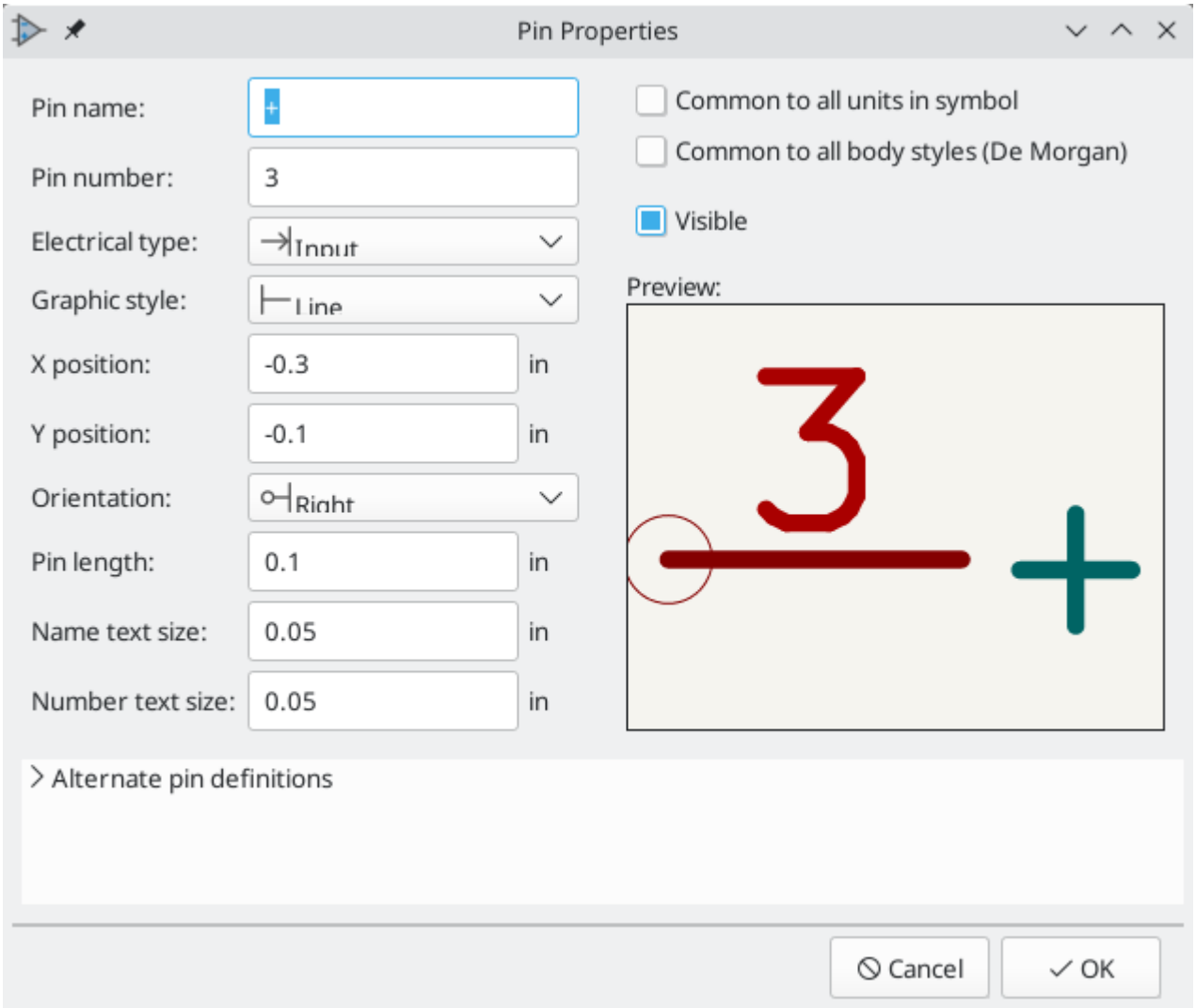
A pin is defined by its graphical representation, its name and its number. The pin's name and number can contain letters, numbers, and symbols, but not spaces. For the Electrical Rules Check (ERC) tool to be useful, the pin's electrical type (input, output, tri-state...) must also be defined correctly.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

Important notes:

- 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 $\overline{\text{FO}}$ 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

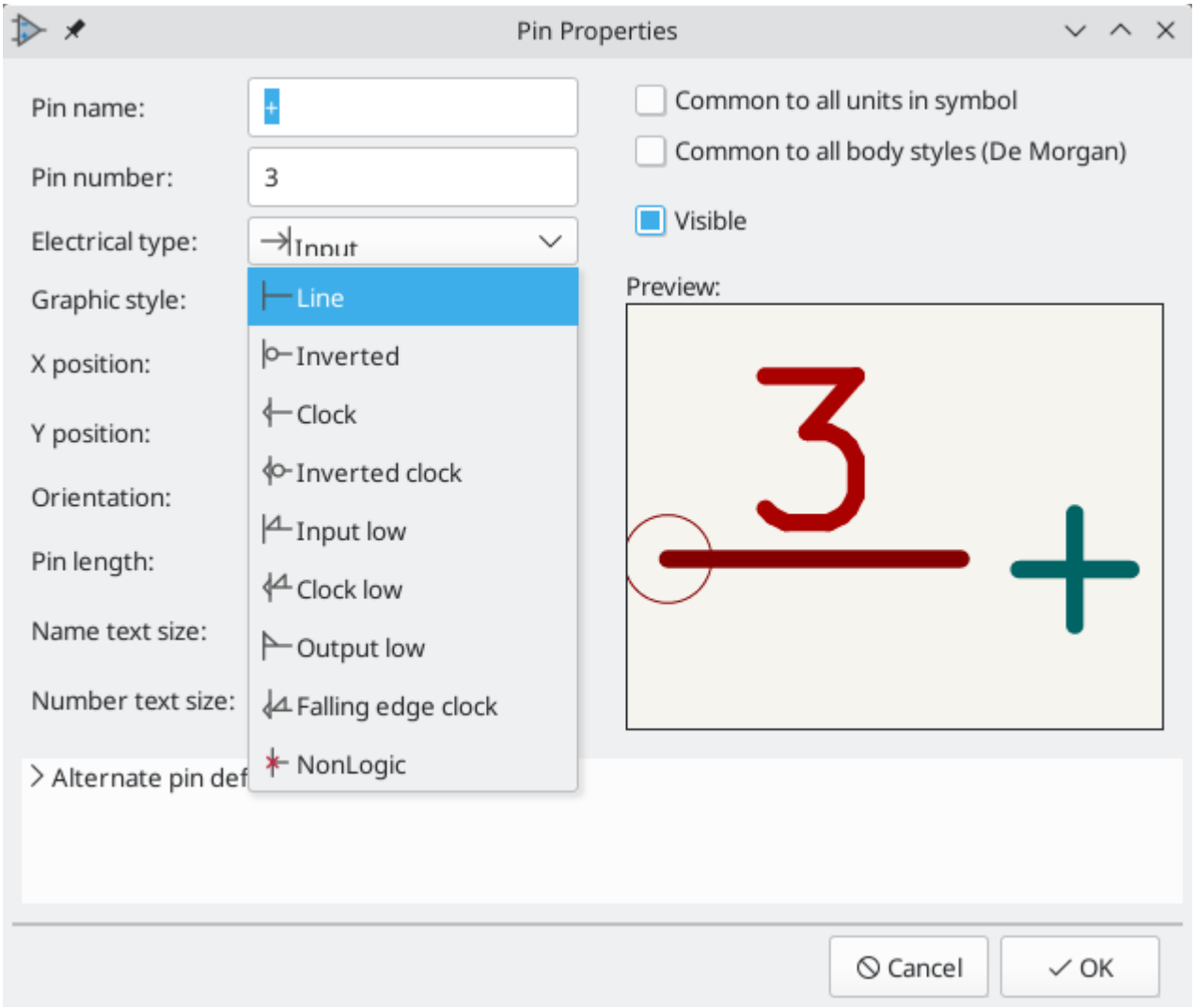


The pin properties dialog allows you to edit all of the characteristics of a pin. This dialog pops up automatically when you create a pin or when double-clicking on an existing pin. This dialog allows you to modify:

- The pin name and text size.
- The pin number and text size.
- The pin length.
- The pin electrical type and graphical style.
- Unit and alternate representation membership.
- Pin visibility.
- Alternate pin definitions.

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.



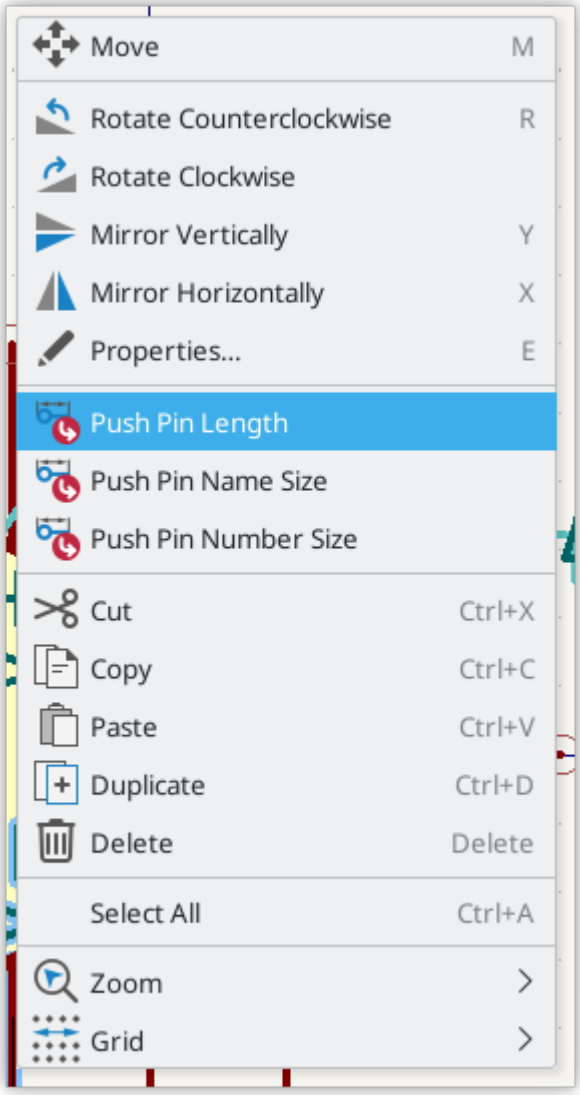
Pin Electrical Types

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.

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 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

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.



Defining Pins for Multiple Units and Alternate Symbolic Representations

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 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 button on the tool bar. To edit the pin number for each unit, select the appropriate unit using the 74LS00 drop down control.

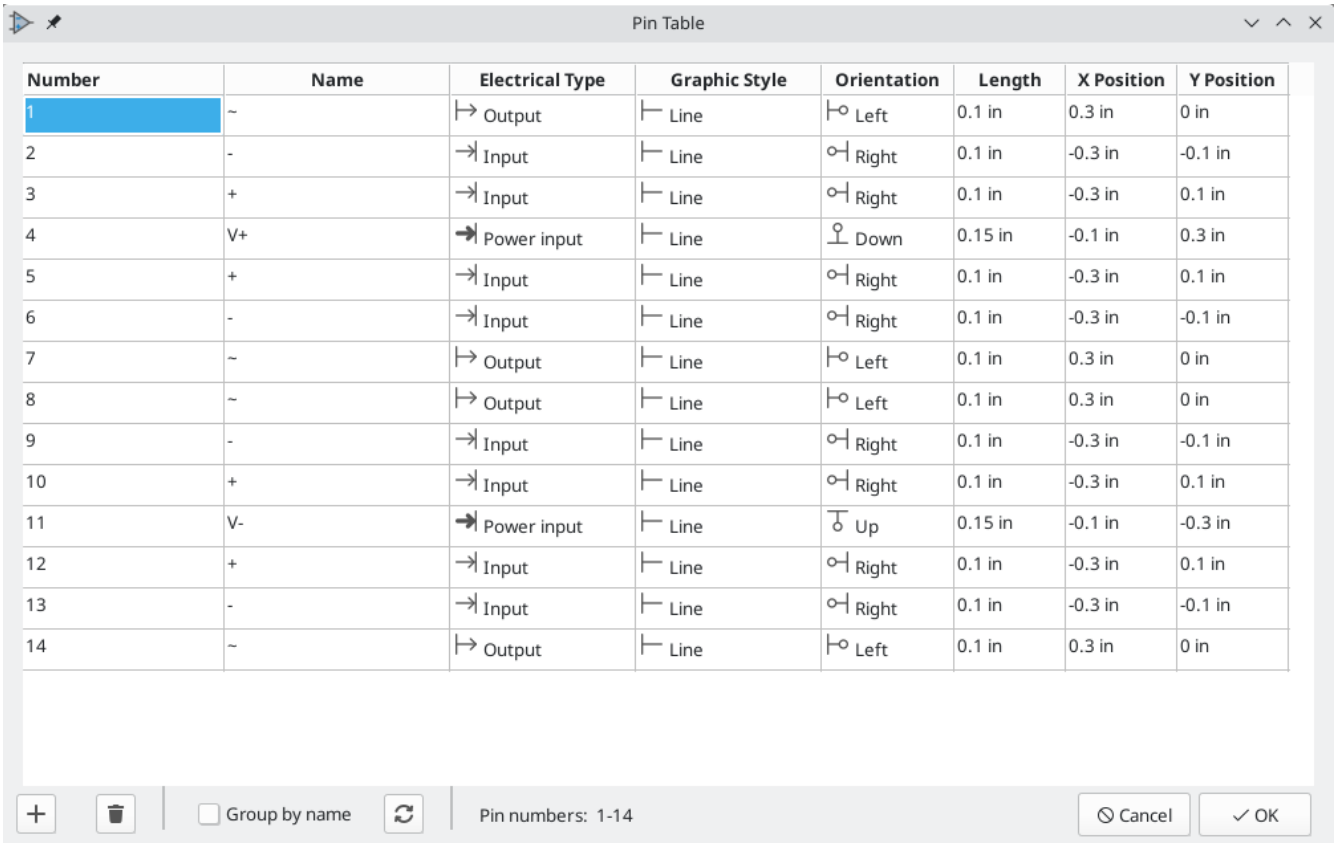
Pin Table

Another way to edit pins is to use the Pin Table, which is accessible via the 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 + and trash 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.

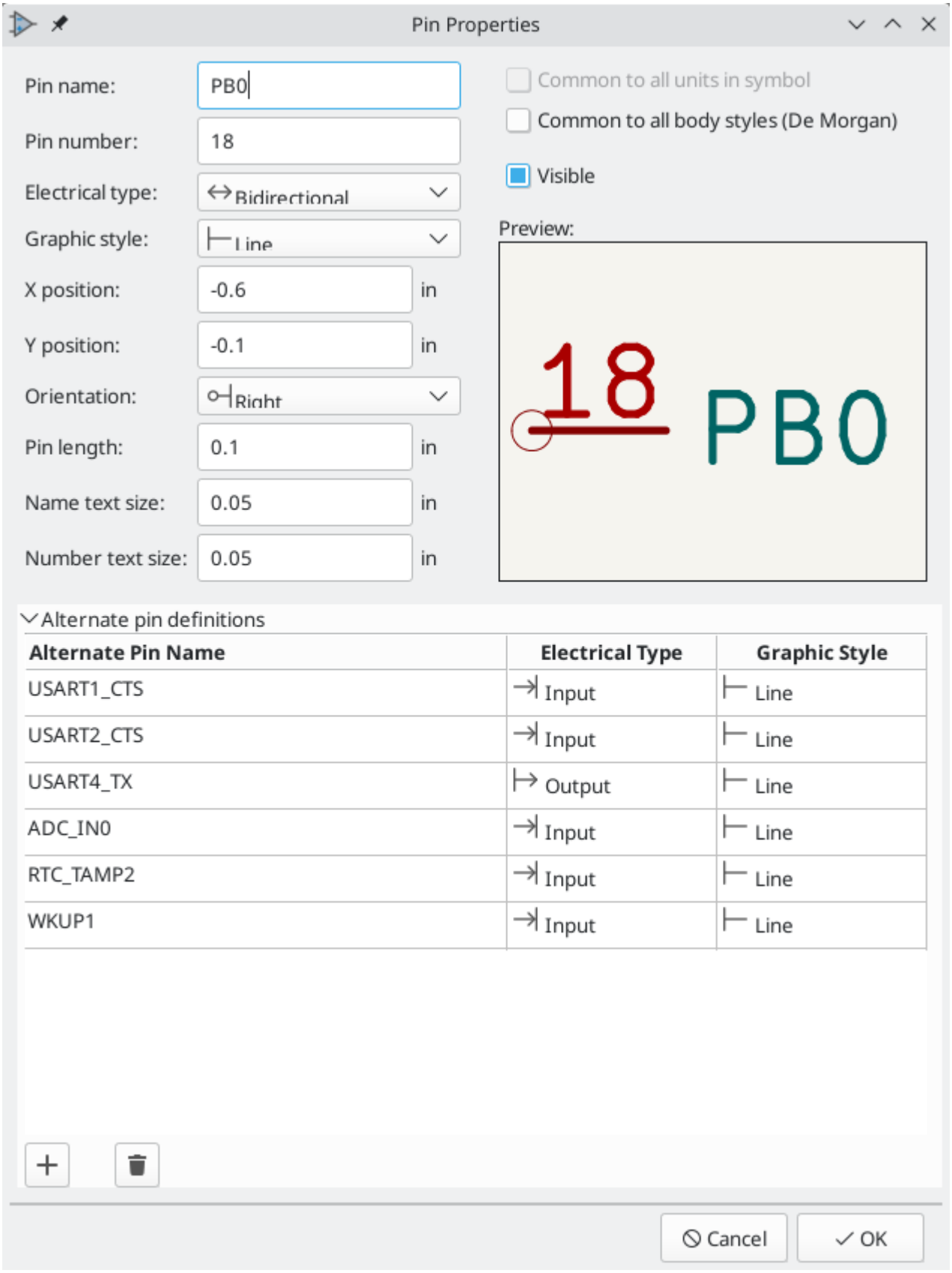


Number	Name	Electrical Type	Graphic Style	Orientation	Length	X Position	Y Position
1	~	Output	Line	Left	0.1 in	0.3 in	0 in
2	-	Input	Line	Right	0.1 in	-0.3 in	-0.1 in
3	+	Input	Line	Right	0.1 in	-0.3 in	0.1 in
4	V+	Power input	Line	Down	0.15 in	-0.1 in	0.3 in
5	+	Input	Line	Right	0.1 in	-0.3 in	0.1 in
6	-	Input	Line	Right	0.1 in	-0.3 in	-0.1 in
7	~	Output	Line	Left	0.1 in	0.3 in	0 in
8	~	Output	Line	Left	0.1 in	0.3 in	0 in
9	-	Input	Line	Right	0.1 in	-0.3 in	-0.1 in
10	+	Input	Line	Right	0.1 in	-0.3 in	0.1 in
11	V-	Power input	Line	Up	0.15 in	-0.1 in	-0.3 in
12	+	Input	Line	Right	0.1 in	-0.3 in	0.1 in
13	-	Input	Line	Right	0.1 in	-0.3 in	-0.1 in
14	~	Output	Line	Left	0.1 in	0.3 in	0 in

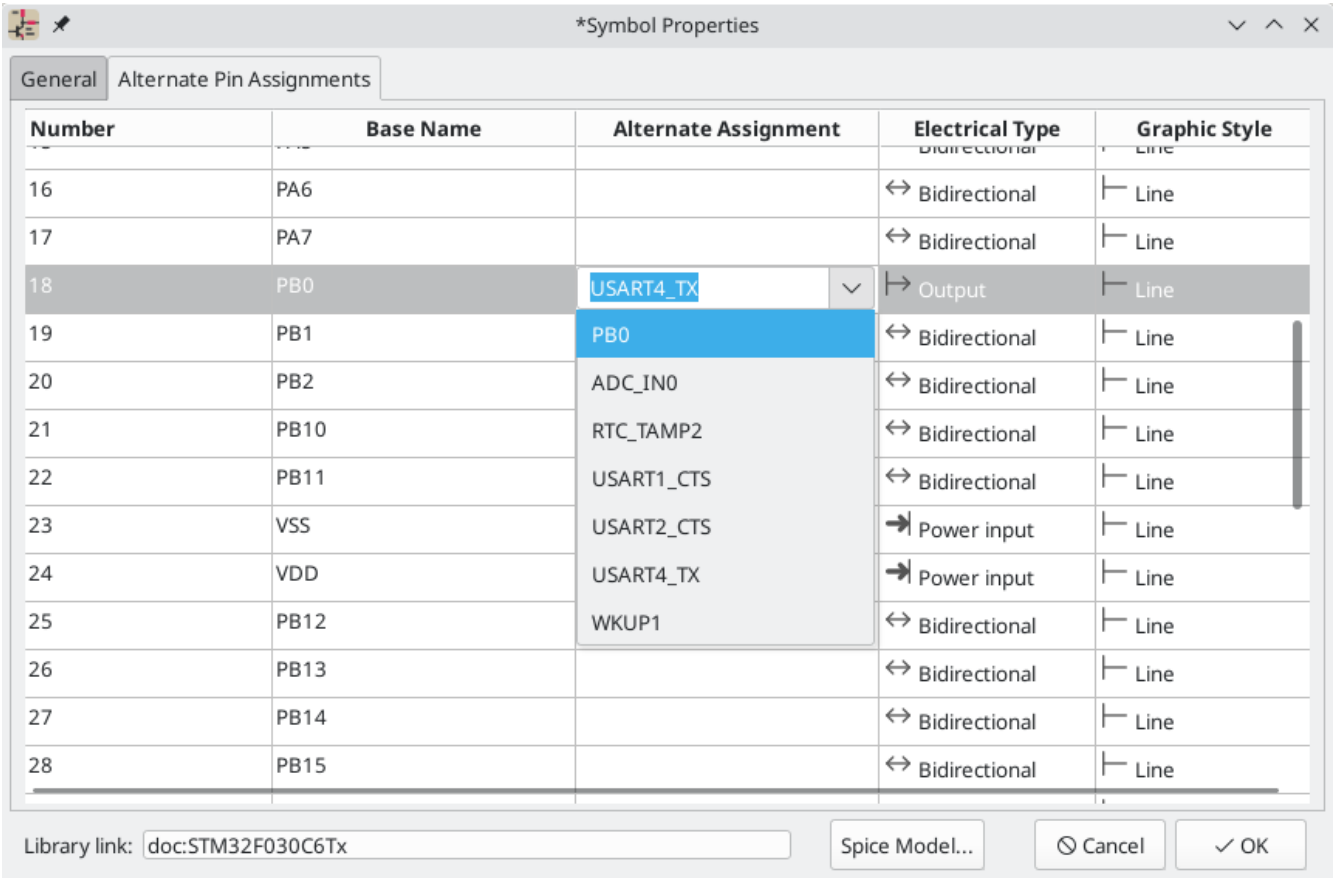
Alternate Pin Definitions

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.



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.



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.

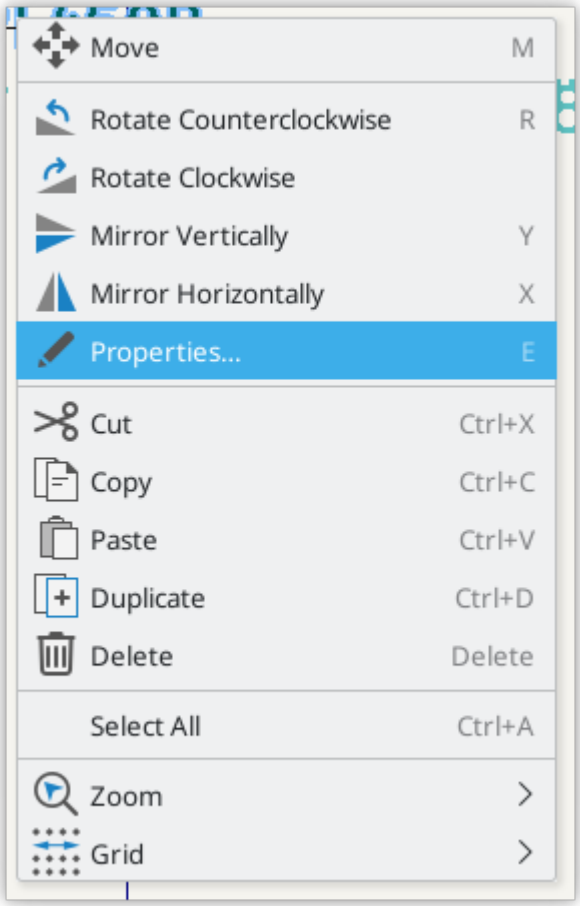
Symbols defined in libraries are typically defined with only these four default fields. Additional fields such as vendor, part number, unit cost, etc. can be added to library symbols but generally this is done in the schematic editor so the additional fields can be applied to all of the symbols in the 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

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



To add new fields, delete optional fields, or edit existing fields, use the  icon on the main tool bar to open the Symbol Properties dialog.

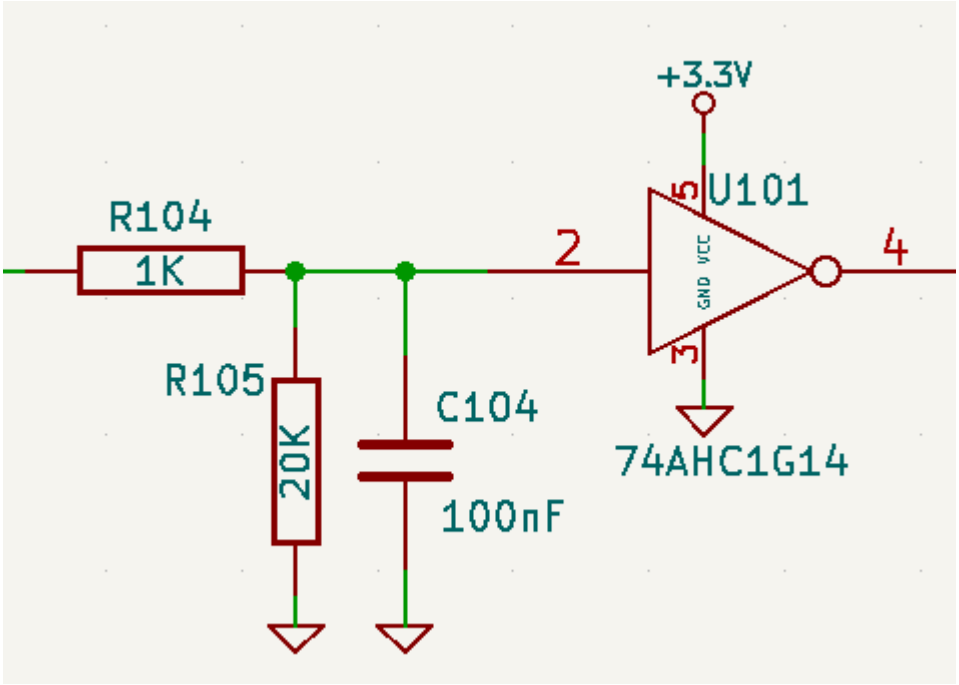
Fields are text information associated a the symbol. Do not confuse them with text in the graphic representation of a symbol.

Important notes:

- 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`.

Power Ports

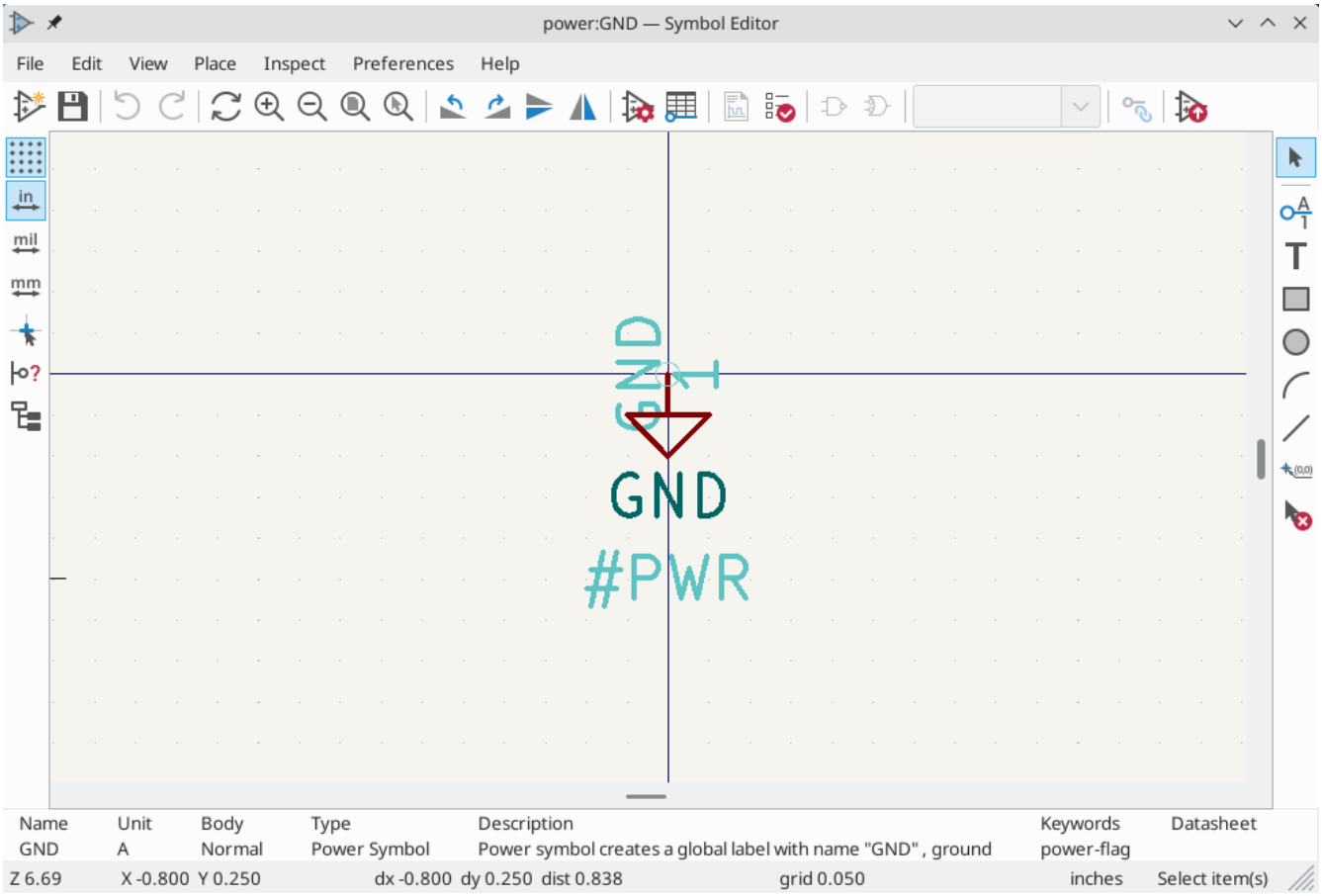
Power ports, or power symbols, are conventionally used to label a wire as part of a power net, like `VCC`, `+5V`, or `GND`. In the schematic below, the `+3.3V` and `GND` symbols are power ports. In addition to acting as a visual indicator that a net is a power rail, a power port will determine the name of the net it is attached to. This is true even if there is another net label attached to the net; the net name determined by the power symbol overrides any other net names.



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.

Power symbols are handled and created the same way as normal symbols, but there are several additional considerations described below. They consist of a graphical symbol and a pin of the type "Power input" that is marked hidden.

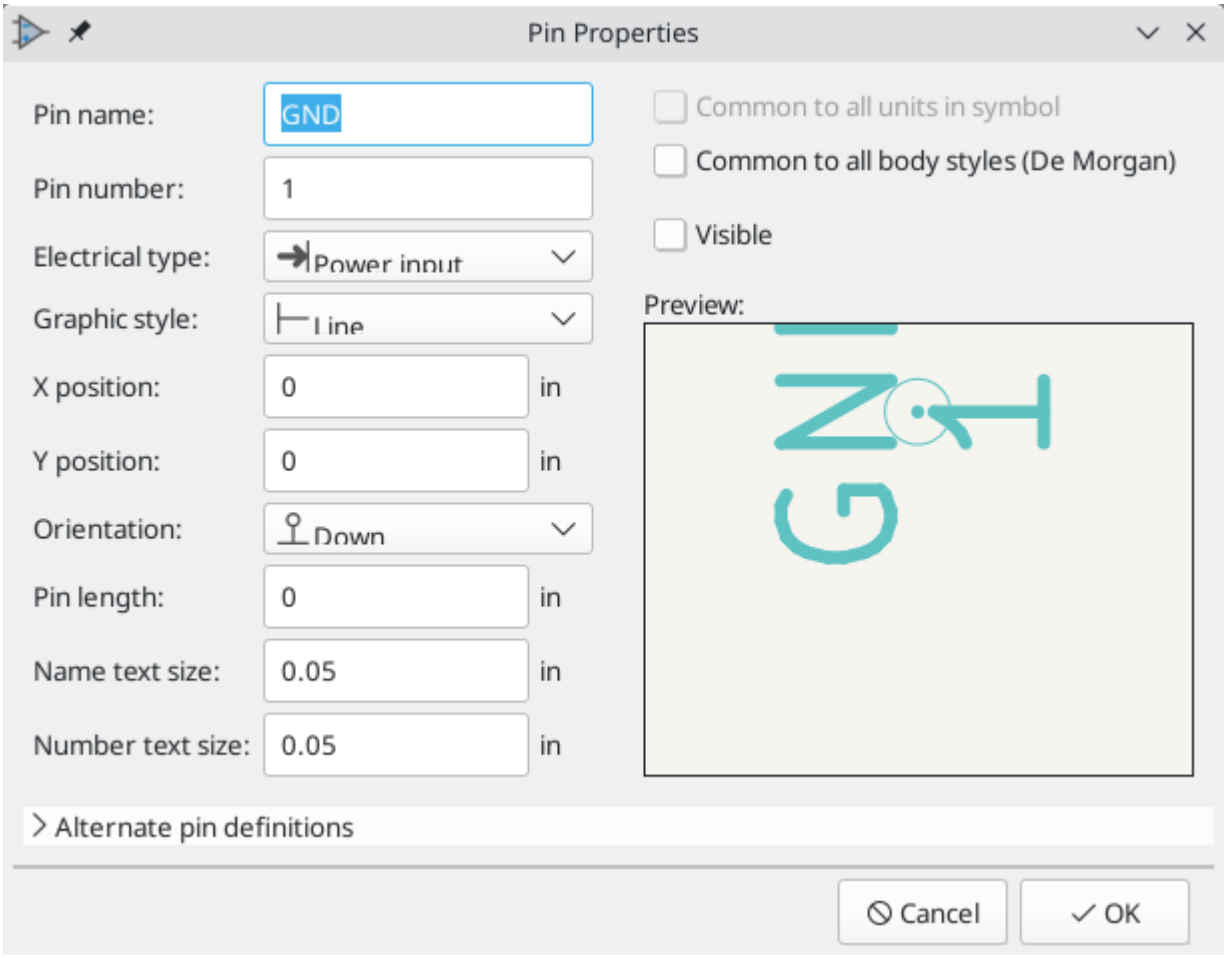
Below is an example of a `GND` power symbol.



Creating a Power Port Symbol

Power Port symbols consist of a pin of type "Power input" that is marked invisible. Invisible power input pins have a special property of automatically connecting to a net with the same name as the pin name. A net that is wired to an invisible power input pin will therefore be named after the pin, even if there are other net labels on the net. This connection is global.

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.



To create a power symbol, use the following steps:

- 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.

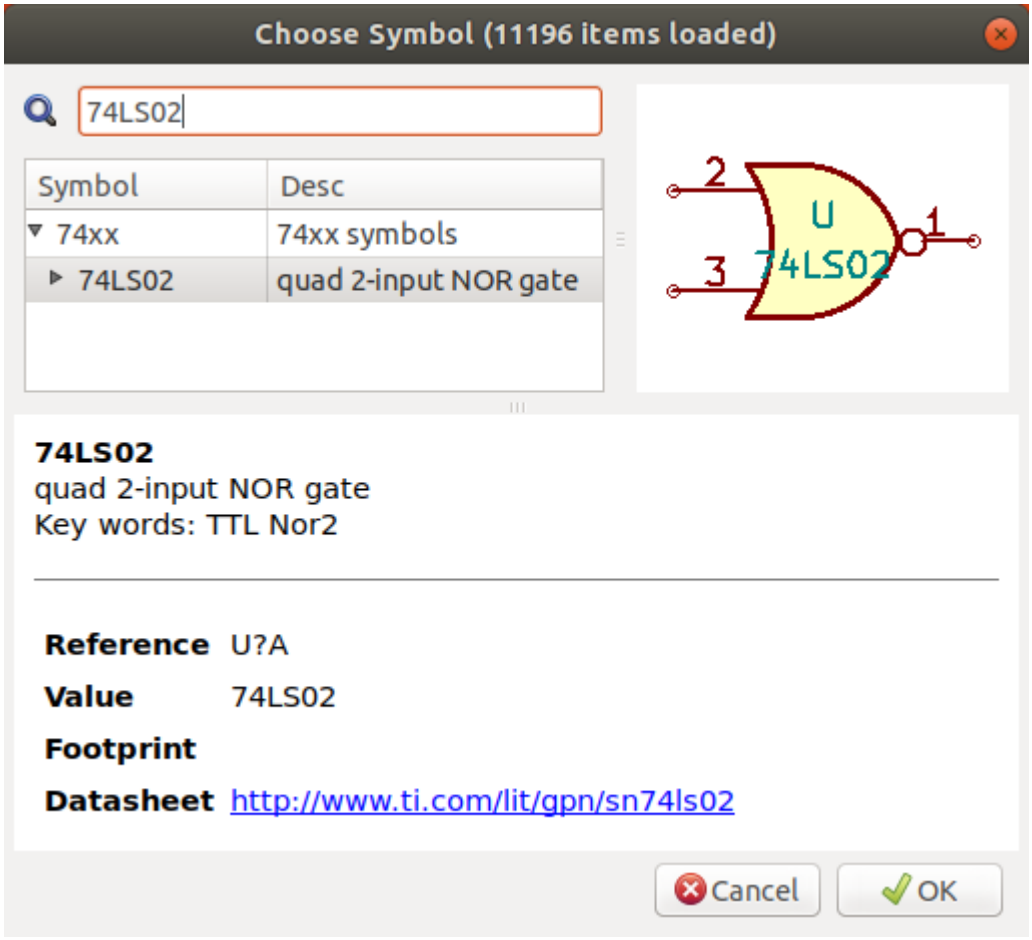
An easier method to create a new power port symbol is to use another symbol as a starting point, as described earlier.

NOTE When modifying an existing power port symbol, make sure to rename the pin name so that the new symbol connects to the appropriate power net.

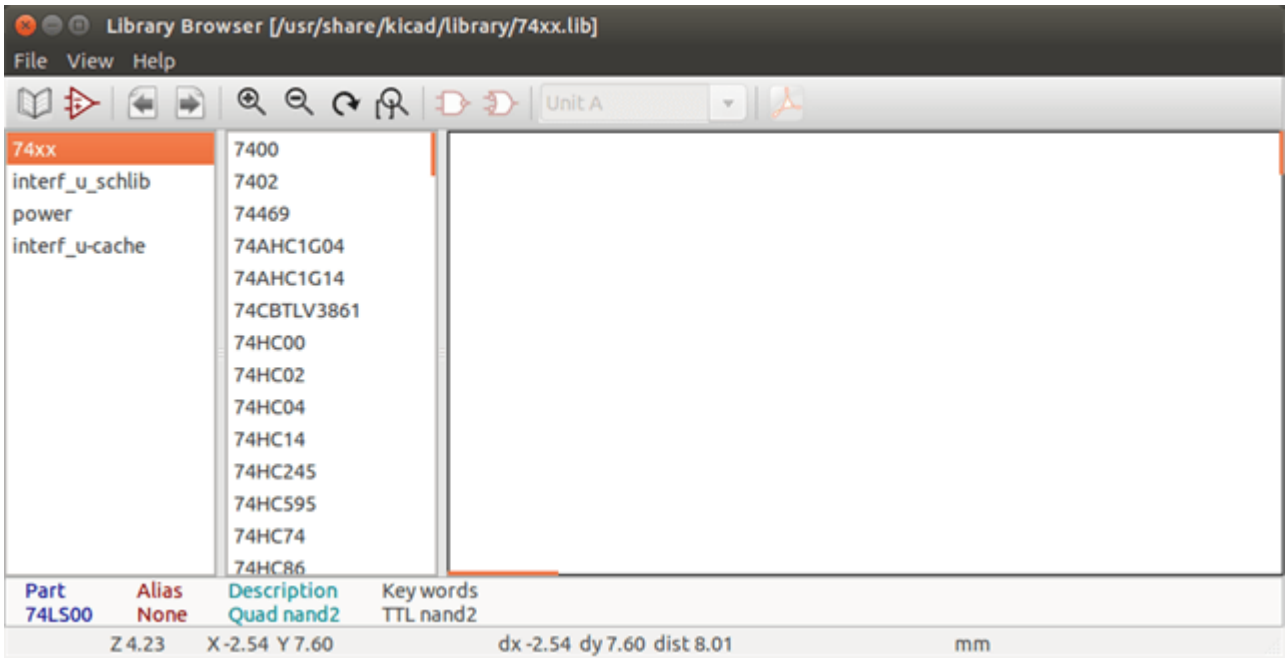
Symbol Library Browser

Introduction

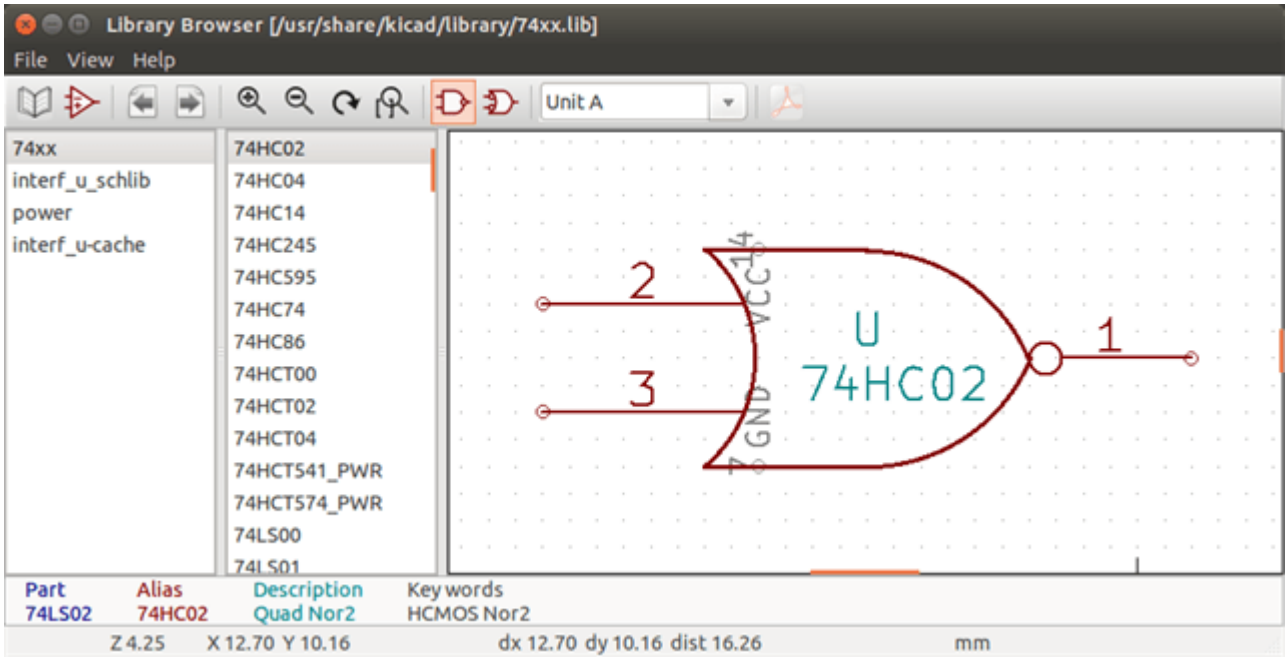
The Symbol Library Browser allows you to quickly examine the content of symbol libraries. The Symbol Library Viewer can be accessed by clicking  icon on the main toolbar, **View → Symbol Library Browser...**, or clicking **Select With Browser** in the "Choose Symbol" window.



Viewlib - main screen



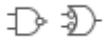
To examine the contents of a library, select a library from the list on the left hand pane. All symbols in the selected library will appear in the second pane. Select a symbol name to view the symbol.



Symbol Library Browser Top Toolbar

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

The available commands are:

	Selection of the symbol which can be also selected in the displayed list.
	Display previous symbol.
	Display next symbol.
	Zoom tools.
	Selection of the representation (normal or alternate) if an alternate representation exists.
	Selection of the unit for symbols that contain multiple units.
	If they exist, display the associated documents.
	Close the browser and place the selected symbol in the schematic.

Create a Netlist

Overview

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

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

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

- PCB layout software.
- Schematic and electrical signal simulators.
- Programmable logic (FPGA, CPLD, etc.) compilers.

KiCad supports several netlist formats:

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

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

Netlist formats

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

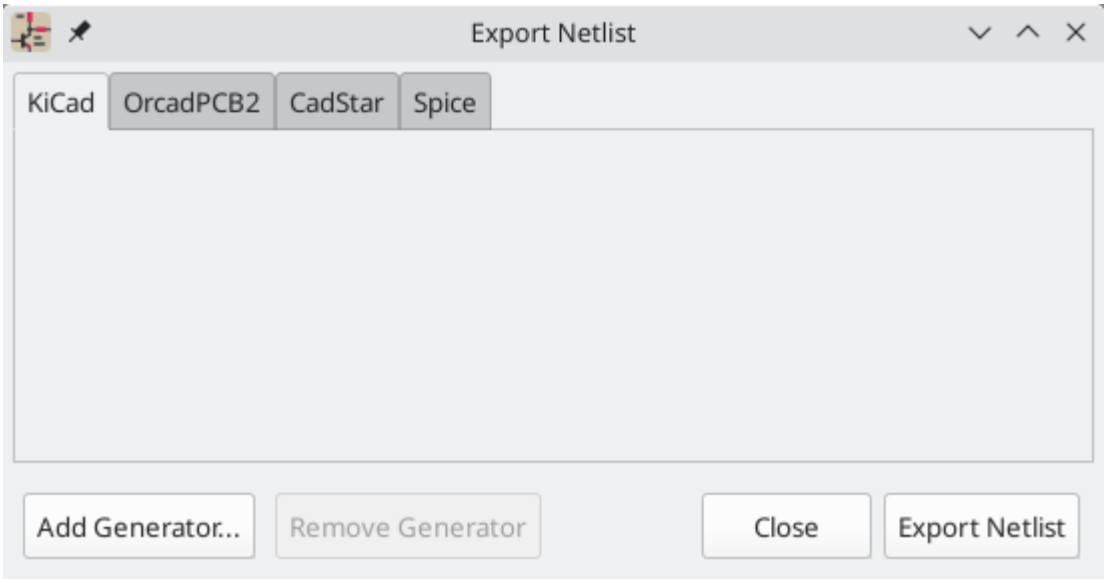
Several netlist formats are available, and are selectable with the tabs at the top of the window. Some netlist formats have options.

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

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

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

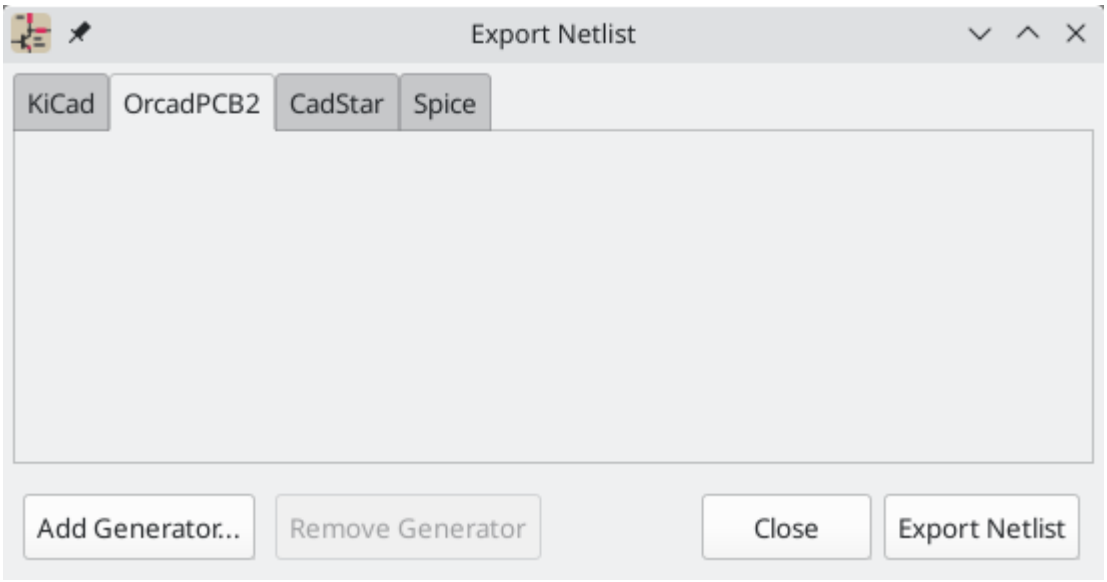
KiCad Netlist Format



The KiCad netlist exporter does not have any options.

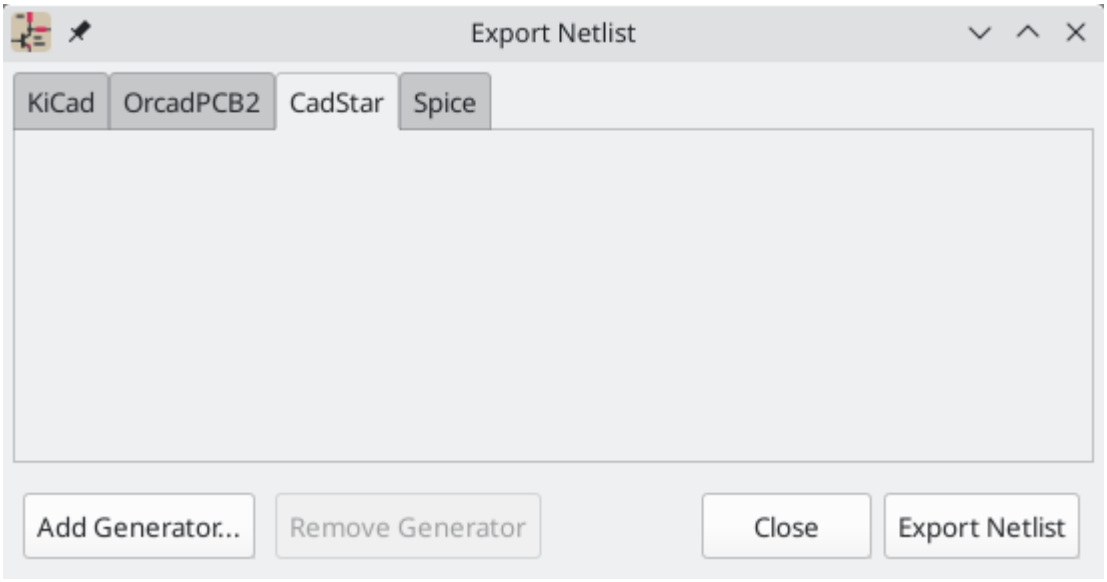
NOTE In KiCad version 5.0 and later, it is not necessary to create a netlist for transfering a design from the schematic editor to the PCB editor. Instead, use the "Update PCB from Schematic" tool.

OrCAD PCB2 Netlist Format



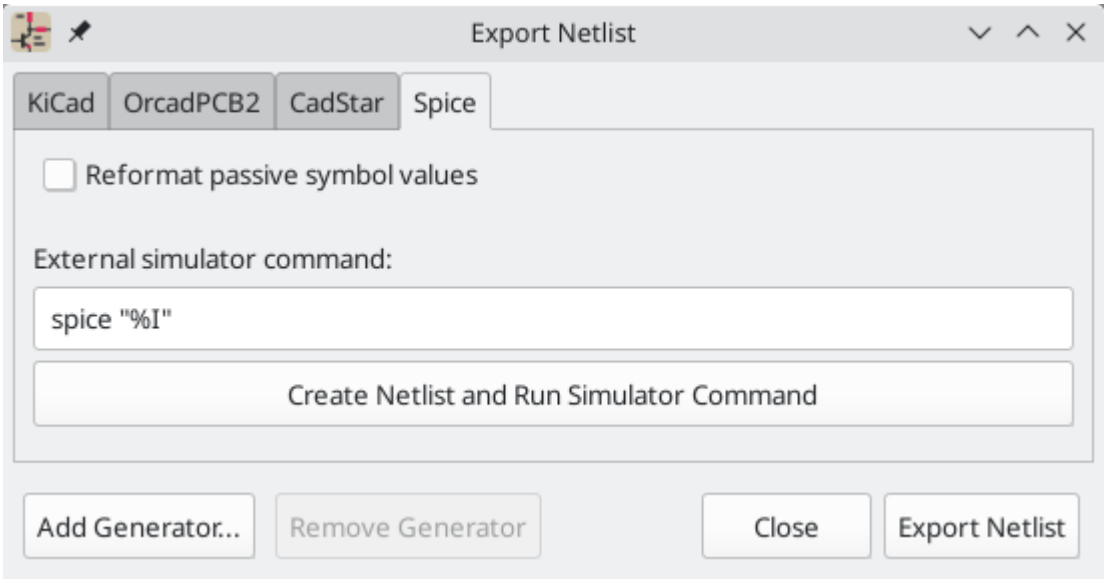
The OrCAD netlist exporter does not have any options.

CADSTAR Netlist Format



The CADSTAR netlist exporter does not have any options.

Spice Netlist Format



The Spice netlist format offers several options.

When the **Reformat passive symbol values** box is checked, passive symbol values will be adjusted to be compatible with Spice. Specifically:

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

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

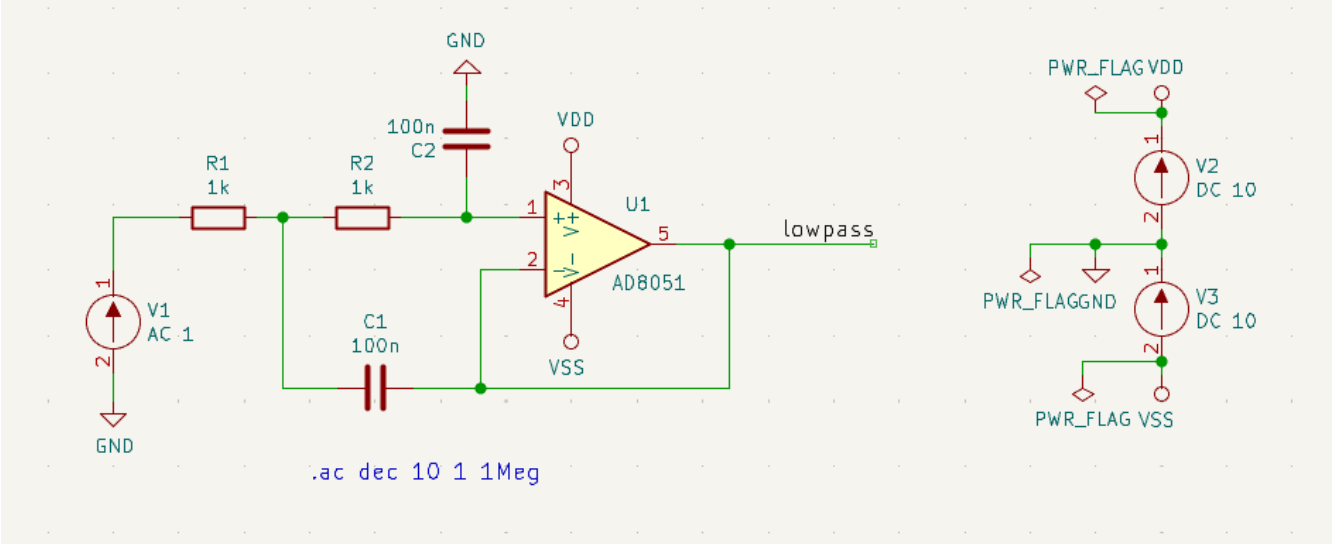
Enter the path to the external simulator in the text box, with `%I` representing the generated netlist. Click the **Create Netlist and Run Simulator Command** button to generate the netlist and automatically run the simulator.

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

For more information on the contents of Spice netlists, see the Spice netlist section.

Netlist examples

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



The KiCad format netlist for this schematic is as follows:

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

In Spice format, the netlist is as follows:

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

Notes on Netlists

Netlist name precautions

Many software tools that use netlists do not accept spaces in component names, pins, nets, or other fields. Avoid using spaces in pins, labels, names, and value fields of components to ensure maximum compatibility.

In the same way, special characters other than letters and numbers can cause problems. Note that this limitation is not related to KiCad, but to the netlist formats that can then become untranslatable by other software that reads those netlist files.

Spice netlists

Spice simulators expect simulation commands (`.PROBE` , `.AC` , `.TRAN` , etc.) to be included in the netlist.

Any text line included in the schematic diagram starting with a period (`.`) will be included in the netlist. If a text object contains multiple lines, only the lines beginning with a period will be included.

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

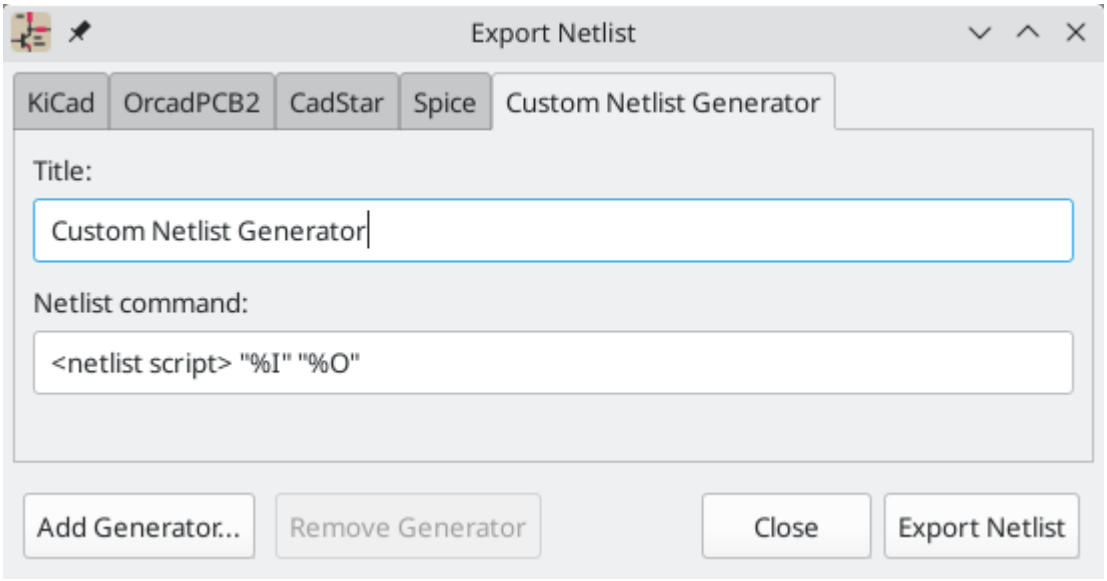
Other formats

KiCad supports custom netlist generators for exporting netlists in other formats. Some examples of netlist generators are given in the custom netlist generators section.

A netlist generator is a script or program that converts the intermediate netlist file created by KiCad into the desired netlist format. The intermediate netlist file contains all of the netlist information required to create an arbitrary netlist for the schematic. Python and XSLT are commonly used tools to create custom netlist generators.

Adding new netlist generators

New netlist generators are added by clicking the **Add Generator…** button.



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 `%I` argument represents the input intermediate netlist filename and the `%O` argument represents the output netlist filename. The exact netlist command will depend on the generator script used.

Command line format

Consider the following example which uses `xsltproc` to generate a netlist in PADS ASC format. `xsltproc` converts the intermediate netlist using the `netlist_form_pads-pcb.asc.xsl` stylesheet to define the output format:

```
xsltproc -o %O.net /usr/share/kicad/plugins/netlist_form_pads-pcb.asc.xsl %I
```

The purpose of each part of the command is as follows:

`xsltproc`	A tool to convert an XML file (the intermediate netlist) according to an XSLT stylesheet.
`-o %O.net`	Output filename. `%O` is replaced with the name of the intermediate netlist file, which is `<schematic name>.xml`. Therefore in this example the complete output filename is `<schematic name>.xml.net`. An arbitrary output filename can be specified if desired with `-o <filename>`.
`/usr/share/kicad/plugins/netlist_form_pads-pcb.asc.xsl`	XSLT stylesheet which determines how the output is formatted. This particular stylesheet is included with KiCad, but custom stylesheets can also be created.
`%I`	Input (intermediate netlist) filename. `%I` is replaced with the name of the intermediate netlist file, which is `<schematic name>.xml`.

For netlist generators that do not use `xsltproc`, the generator command will differ.

Intermediate netlist file format

See the custom netlist generators section for more information about netlist generators, a description of the intermediate netlist format, and some examples of netlist generators.

Creating Customized Netlists and BOM Files

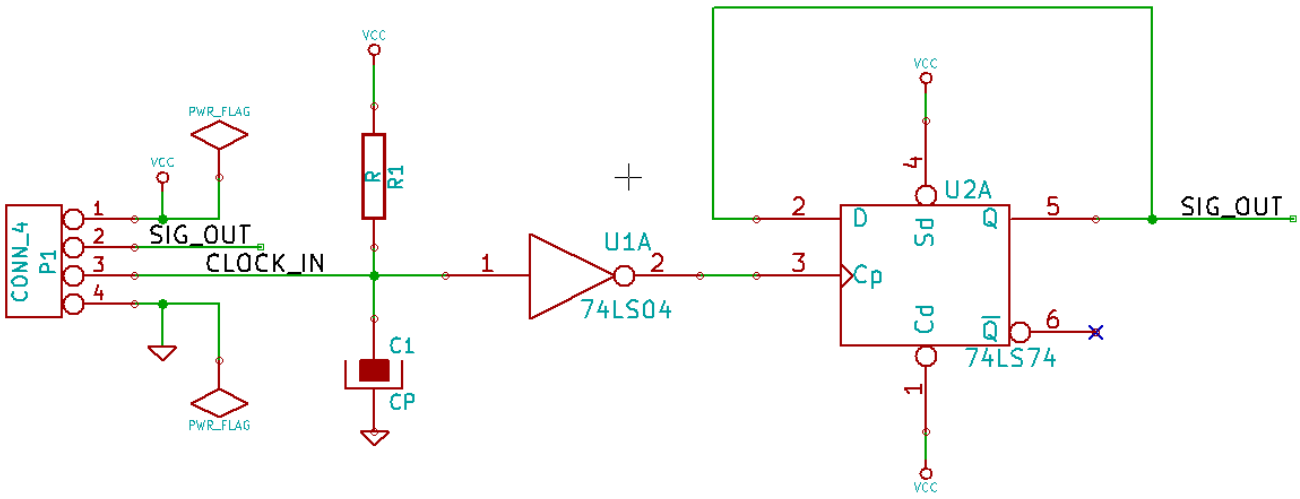
Intermediate Netlist File

BOM files and netlist files can be converted from an Intermediate netlist file created by KiCad.

This file uses XML syntax and is called the intermediate netlist. The intermediate netlist includes a large amount of data about your board and because of this, it can be used with post-processing to create a BOM or other reports.

Depending on the output (BOM or netlist), different subsets of the complete Intermediate Netlist file will be used in the post-processing.

Schematic sample



The Intermediate Netlist file sample

The corresponding intermediate netlist (using XML syntax) of the circuit above is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 20:35:21</date>
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2141</tstamp>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20BA</tstamp>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20A6</tstamp>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2094</tstamp>
    </comp>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E208A</tstamp>
    </comp>
  </components>
  <libparts>
    <libpart lib="device" part="C">
      <description>Condensateur non polarise</description>
      <footprints>
        <fp>SM*</fp>
        <fp>C?</fp>
        <fp>C1-1</fp>
      </footprints>
      <fields>
        <field name="Reference">C</field>
        <field name="Value">C</field>
      </fields>
      <pins>
        <pin num="1" name="~" type="passive"/>
        <pin num="2" name="~" type="passive"/>
      </pins>
    </libpart>
    <libpart lib="device" part="R">
      <description>Resistance</description>
      <footprints>
        <fp>R?</fp>
        <fp>SM0603</fp>
        <fp>SM0805</fp>
```

Conversion to a new netlist format

By applying a post-processing filter to the Intermediate netlist file you can generate foreign netlist files as well as BOM files. Because this conversion is a text to text transformation, this post-processing filter can be written using Python, XSLT, or any other tool capable of taking XML as input.

XSLT itself is an XML language very suitable for XML transformations. There is a free program called *xsltproc* that you can download and install. 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.

XSLT approach

The document that describes XSL Transformations (XSLT) is available here:

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

Create a Pads-Pcb netlist file

The pads-pcb format is comprised of two sections.

- The footprint list.
- The Nets list: grouping pads references by nets.

Immediately below is a style-sheet which converts the Intermediate Netlist file to a pads-pcb 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 -->
<xsl:template match="node">
    <xsl:text> </xsl:text>
```

And here is the pads-pcb output file after running xsltproc:

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

*END*
```

The command line to make this conversion is:

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

Create a Cadstar netlist file

The Cadstar format is comprised of two sections.

- The footprint list.
- The Nets list: grouping pads references by nets.

Here is the style-sheet file to make this specific conversion:

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

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

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

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

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

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

<!-- for each component -->
<xsl:template match="comp">
    <xsl:text>.ADD_COM </xsl:text>
    <xsl:value-of select="@ref"/>
    <xsl:text> </xsl:text>
    <xsl:choose>
        <xsl:when test = "value != '' ">
            <xsl:text>"</xsl:text> <xsl:apply-templates select="value"/> <xsl:text>"
</xsl:text>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>""</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

<!-- for each net -->
<xsl:template match="net">
    <!-- nets are output only if there is more than one pin in net -->
    <xsl:if test="count(node)>1">
    <xsl:variable name="netname">
```

Here is the Cadstar output file.

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


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

.END
```

Create an OrcadPCB2 netlist file

This format has only one section which is the footprint list. Each footprint includes its list of pads with reference to a net.

Here is the style-sheet for this specific conversion:

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

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

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

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

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

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

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

<!--
    Generate id in header like "eeschema (2010-08-17 BZR 2450)-unstable"
-->
<xsl:template match="tool">
    <xsl:apply-templates/>
</xsl:template>

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

<!--
    This template read each component
    (path = /export/components/comp)
    creates lines:
     ( 3EBF7DBD $noname U1 74LS125
       ... pin list ...
       )
    and calls "create_pin_list" template to build the pin list
-->
<xsl:template match="comp">
    <xsl:text> ( </xsl:text>
```

Here is the OrcadPCB2 output file.

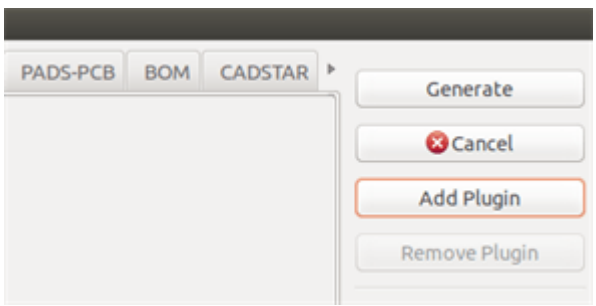
```
( { Eeschema Netlist Version 1.1  29/08/2010 21:07:51
eeschema (2010-08-28 BZR 2458)-unstable}
 ( 4C6E2141 $noname P1 CONN_4
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 /CLOCK_IN )
  (  4 GND )
 )
 ( 4C6E20BA $noname U2 74LS74
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 N-04 )
  (  4 VCC )
  (  5 /SIG_OUT )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

Netlist plugins interface

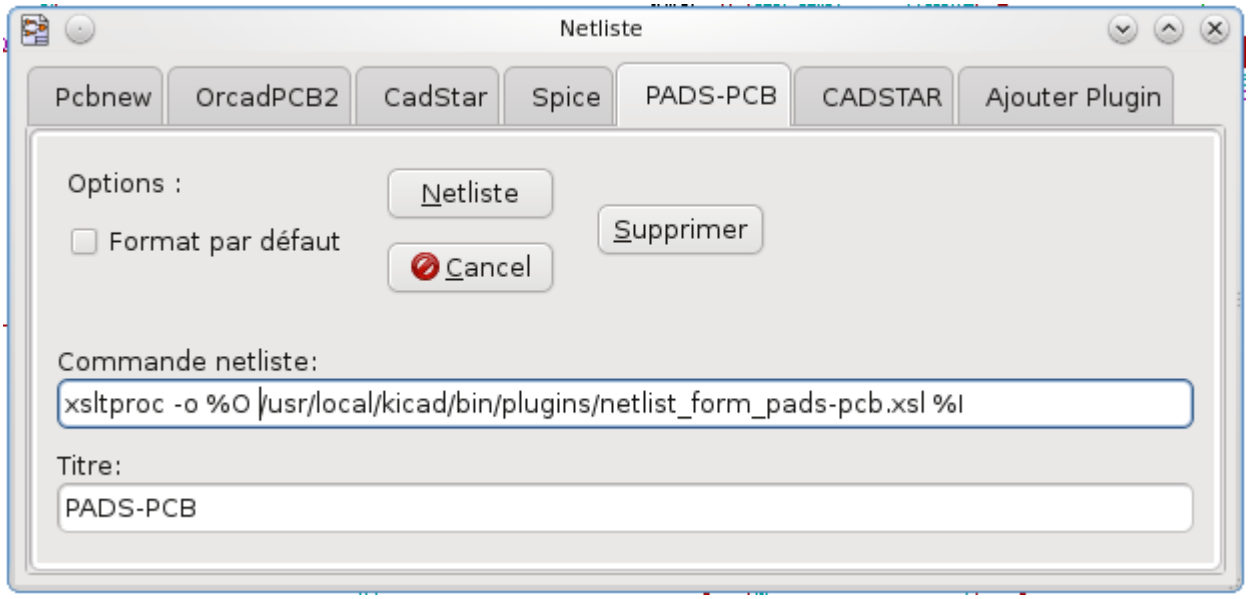
Intermediate Netlist converters can be automatically launched within the Schematic Editor.

Init the Dialog window

One can add a new netlist plug-in user interface tab by clicking on the Add Plugin button.



Here is what the configuration data for the PadsPcb tab looks like:



Plugin Configuration Parameters

The netlist plug-in configuration dialog requires the following information:

- The title: for instance, the name of the netlist format.
- The command line to launch the converter.

Once you click on the netlist button the following will happen:

1. KiCad creates an intermediate netlist file *.xml, for instance test.xml.
2. KiCad runs the plug-in by reading test.xml and creates test.net.

Generate netlist files with the command line

Assuming we are using the program *xsltproc.exe* to apply the sheet style to the intermediate file, *xsltproc.exe* is executed with the following command:

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

In KiCad under Windows the command line is the following:

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

Under Linux the command becomes as follows:

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

Where *netlist_form_pads-pcb.xsl* is the style-sheet that you are applying. Do not forget the double quotes around the file names, this allows them to have spaces after the substitution by KiCad.

The command line format accepts parameters for filenames:

The supported formatting parameters are.

- %B ⇒ base filename and path of selected output file, minus path and extension.
- %I ⇒ complete filename and path of the temporary input file (the intermediate net file).

%O ⇒ complete filename and path of the user chosen output file.

%I will be replaced by the actual intermediate file name

%O will be replaced by the actual output file name.

Command line format: example for xsltproc

The command line format for xsltproc is the following:

<path of xsltproc> xsltproc <xsltproc parameters>

under Windows:

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

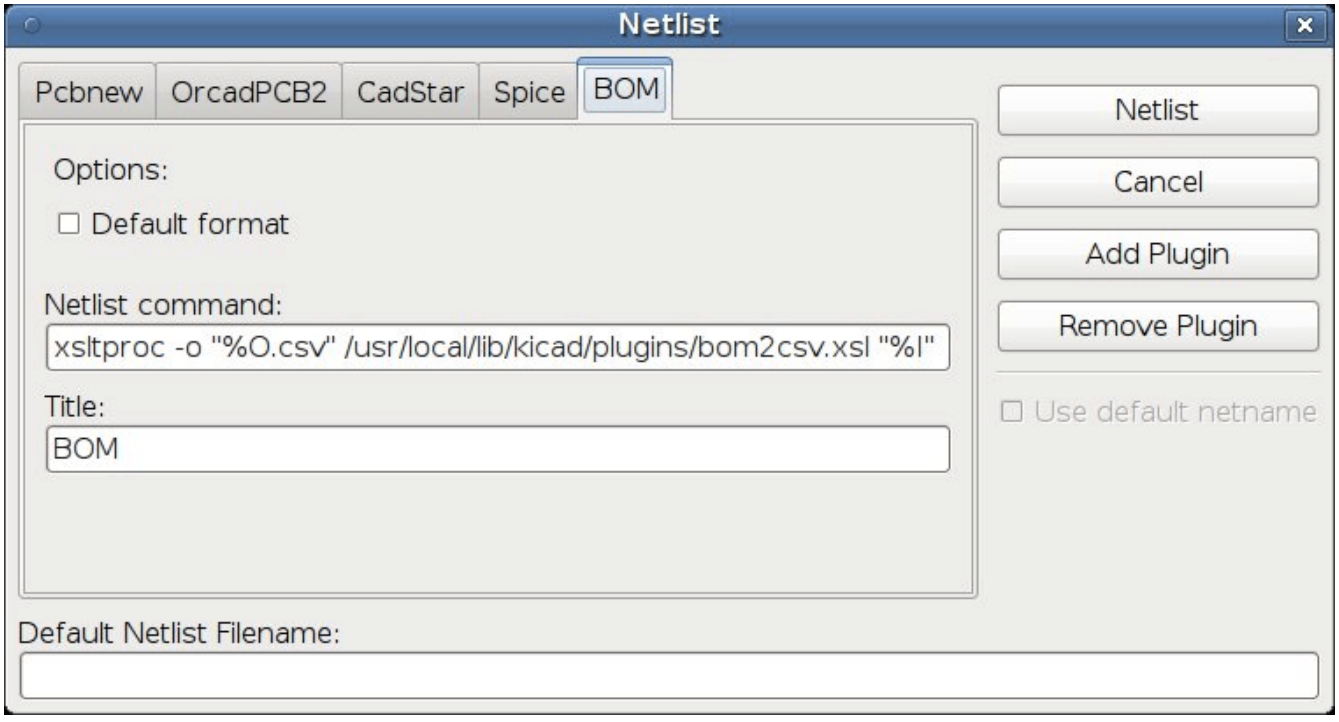
under Linux:

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

The above examples assume xsltproc is installed on your PC under Windows and all files located in kicad/bin.

Bill of Materials Generation

Because the intermediate netlist file contains all information about used components, a BOM can be extracted from it. Here is the plug-in setup window (on Linux) to create a customized Bill Of Materials (BOM) file:



The path to the style sheet bom2csv.xsl is system dependent. The currently best XSLT style-sheet for BOM generation at this time is called *bom2csv.xsl*. You are free to modify it according to your needs, and if you develop something generally useful, ask that it become part of the KiCad project.

Command line format: example for python scripts

The command line format for python is something like:

python <script file name> <input filename> <output filename>

under Windows:

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

under Linux:

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

Assuming python is installed on your PC.

Intermediate Netlist structure

This sample gives an idea of the netlist file format.

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 21:07:51</date>
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2141</tstamp>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20BA</tstamp>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20A6</tstamp>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2094</tstamp>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E208A</tstamp>
    </comp>
  </components>
  <libparts/>
  <libraries/>
  <nets>
    <net code="1" name="GND">
      <node ref="U1" pin="7"/>
      <node ref="C1" pin="2"/>
      <node ref="U2" pin="7"/>
      <node ref="P1" pin="4"/>
    </net>
    <net code="2" name="VCC">
      <node ref="R1" pin="1"/>
      <node ref="U1" pin="14"/>
      <node ref="U2" pin="4"/>
      <node ref="U2" pin="1"/>
      <node ref="U2" pin="14"/>
      <node ref="P1" pin="1"/>
    </net>
    <net code="3" name="">
      <node ref="U2" pin="6"/>
    </net>
    <net code="4" name="">
      <node ref="U1" pin="2"/>
      <node ref="U2" pin="3"/>
    </net>
```

General netlist file structure

The intermediate Netlist accounts for five sections.

- The header section.
- The components section.
- The lib parts section.
- The libraries section.
- The nets section.

The file content has the delimiter <export>

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

The header section

The header has the delimiter <design>

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

This section can be considered a comment section.

The components section

The component section has the delimiter <components>

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

This section contains the list of components in your schematic. Each component is described like this:

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

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 (time stamps)	time stamp of the schematic file.
tstamp (time stamp)	time stamp of the component.

Note about time stamps for components

To identify a component in a netlist and therefore on a board, the timestamp reference is used as unique for each component. However KiCad provides an auxiliary way to identify a component which is the corresponding footprint on the board. This allows the re-annotation of components in a schematic project and does not loose the link between the component and its footprint.

A time stamp is an unique identifier for each component or sheet in a schematic project. However, in complex hierarchies, the same sheet is used more than once, so this sheet contains components having the same time stamp.

A given sheet inside a complex hierarchy has an unique identifier: its sheetpath. A given component (inside a complex hierarchy) has an unique identifier: the sheetpath + its tstamp

The libparts section

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

- The allowed footprints names (names use wildcards) delimiter <fp>.
- The fields defined in the library delimiter <fields>.
- The list of pins delimiter <pins>.

```
<libparts>
<libpart lib="device" part="CP">
  <description>Condensateur polarise</description>
  <footprints>
    <fp>CP*</fp>
    <fp>SM*</fp>
  </footprints>
  <fields>
    <field name="Reference">C</field>
    <field name="Valeur">CP</field>
  </fields>
  <pins>
    <pin num="1" name="1" type="passive"/>
    <pin num="2" name="2" type="passive"/>
  </pins>
</libpart>
</libparts>
```

Lines like <pin num="1" type="passive"/> give also the electrical pin type. Possible electrical pin types are

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

The libraries section

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>
```

The nets section

The nets section has the delimiter <nets>. This section contains the "connectivity" of the schematic.

```
<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>
```

This section lists all nets in the schematic.

A possible net contains the following.

```
<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	is an internal identifier for this net
name	is a name for this net
node	give a pin reference connected to this net

More about xsltproc

Refer to the page: *http://xmlsoft.org/XSLT/xsltproc.html*

Introduction

xsltproc is a command line tool for applying XSLT style-sheets to XML documents. While it was developed as part of the GNOME project, it can operate independently of the GNOME desktop.

xsltproc is invoked from the command line with the name of the style-sheet to be used followed by the name of the file or files to which the style-sheet is to be applied. It will use the standard input if a filename provided is - .

If a style-sheet is included in an XML document with a Style-sheet Processing Instruction, no style-sheet needs to be named in the command line. xsltproc will automatically detect the included style-sheet and use it. By default, the output is to *stdout*. You can specify a file for output using the -o option.

Synopsis

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

Command line options

-V or *--version*

Show the version of libxml and libxslt used.

-v or *--verbose*

Output each step taken by xsltproc in processing the stylesheet and the document.

-o or *--output file*

Direct output to the file named *file*. For multiple outputs, also known as ``chunking'', -o directory/ directs the output files to a specified directory. The directory must already exist.

--timing

Display the time used for parsing the stylesheet, parsing the document and applying the stylesheet and saving the result. Displayed in milliseconds.

--repeat

Run the transformation 20 times. Used for timing tests.

--debug

Output an XML tree of the transformed document for debugging purposes.

--novalid

Skip loading the document's DTD.

--noout

Do not output the result.

--maxdepth value

Adjust the maximum depth of the template stack before libxslt concludes it is in an infinite loop. The default is 500.

--html

The input document is an HTML file.

--param name value

Pass a parameter of name *name* and value *value* to the stylesheet. You may pass multiple name/value pairs up to a maximum of 32. If the value being passed is a string rather than a node identifier, use --stringparam instead.

--stringparam name value

Pass a paramenter of name *name* and value *value* where *value* is a string rather than a node identifier. (Note: The string must be utf-8.)

--nonet

Do not use the Internet to fetch DTD's, entities or documents.

--path paths

Use the list (separated by space or column) of filesystem paths specified by *paths* to load DTDs, entities or documents.

--load-trace

Display to stderr all the documents loaded during the processing.

--catalogs

Use the SGML catalog specified in SGML_CATALOG_FILES to resolve the location of external entities. By default, xsltproc looks for the catalog specified in XML_CATALOG_FILES. If that is not specified, it uses /etc/xml/catalog.

--xinclude

Process the input document using the Xinclude specification. More details on this can be found in the Xinclude specification: http://www.w3.org/TR/xinclude/

--profile --norman

Output profiling information detailing the amount of time spent in each part of the stylesheet. This is useful in optimizing stylesheet performance.

--dumpextensions

Dumps the list of all registered extensions to stdout.

--nowrite

Refuses to write to any file or resource.

--nomkdir

Refuses to create directories.

--writesubtree path

Allow file write only within the *path* subtree.

--nodtdattr

Do not apply default attributes from the document's DTD.

Xsltproc return values

xsltproc returns a status number that can be quite useful when calling it within a script.

0: normal

1: no argument

2: too many parameters

3: unknown option

4: failed to parse the stylesheet

5: error in the stylesheet

6: error in one of the documents

7: unsupported xsl:output method

8: string parameter contains both quote and double-quotes

9: internal processing error

10: processing was stopped by a terminating message

11: could not write the result to the output file

More Information about xsltproc

libxml web page: http://www.xmlsoft.org/

W3C XSLT page: http://www.w3.org/TR/xslt

Simulator

KiCad provides an embedded electrical circuit simulator using 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.

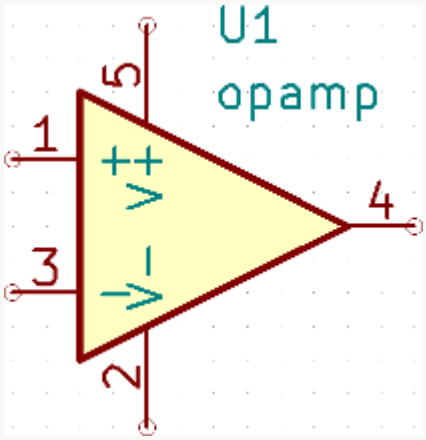
Each component can have only one model assigned, even if component consists of multiple units. In such case, the first unit should have the model specified.

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 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.

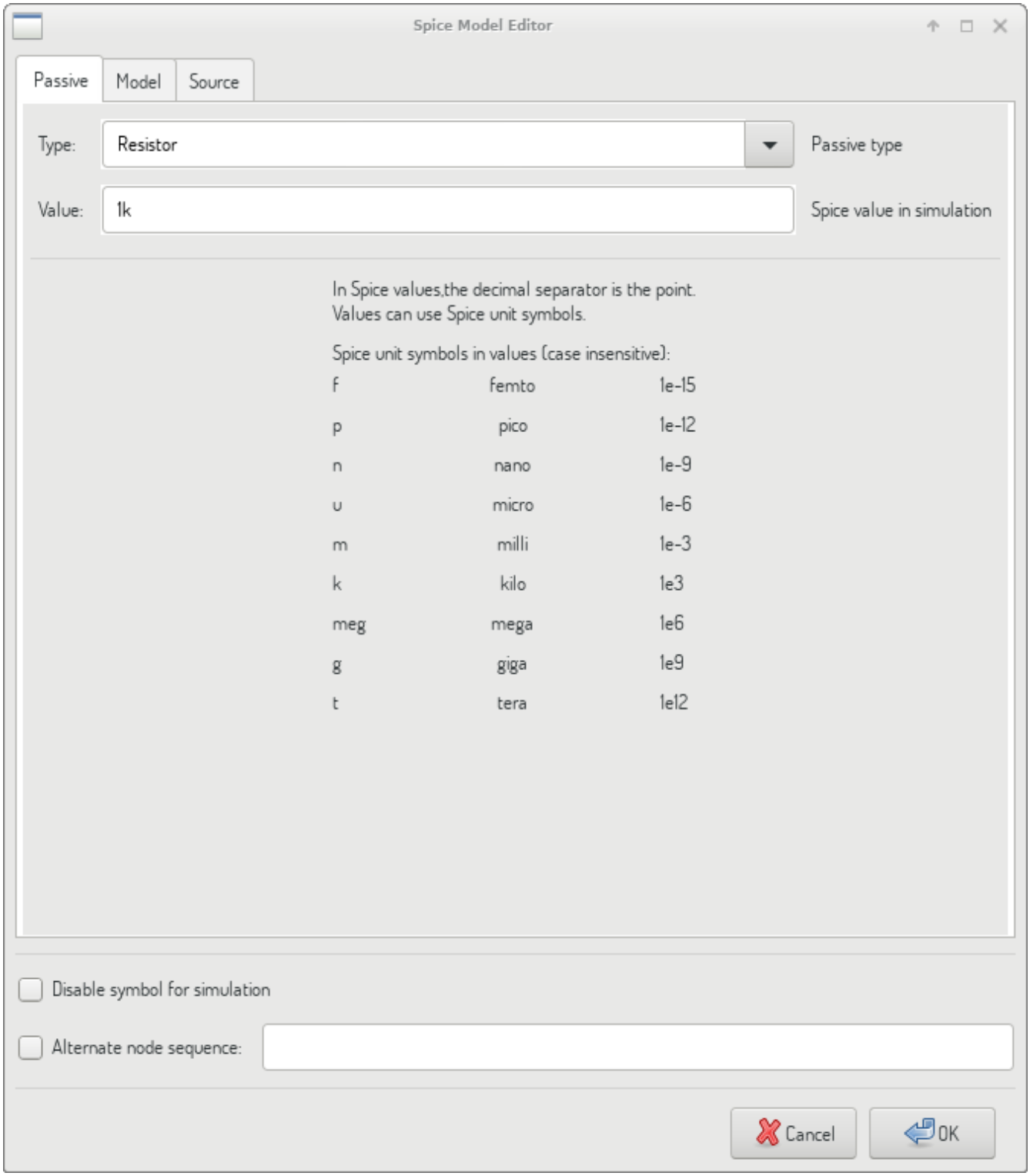
Spice Model Editor dialog has three tabs corresponding to different model types. There are two options common to all model types:

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 ```  To match the symbol pins to the Spice model nodes shown above, one needs to use an alternate node sequence option with value: "1 3 5 2 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 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.

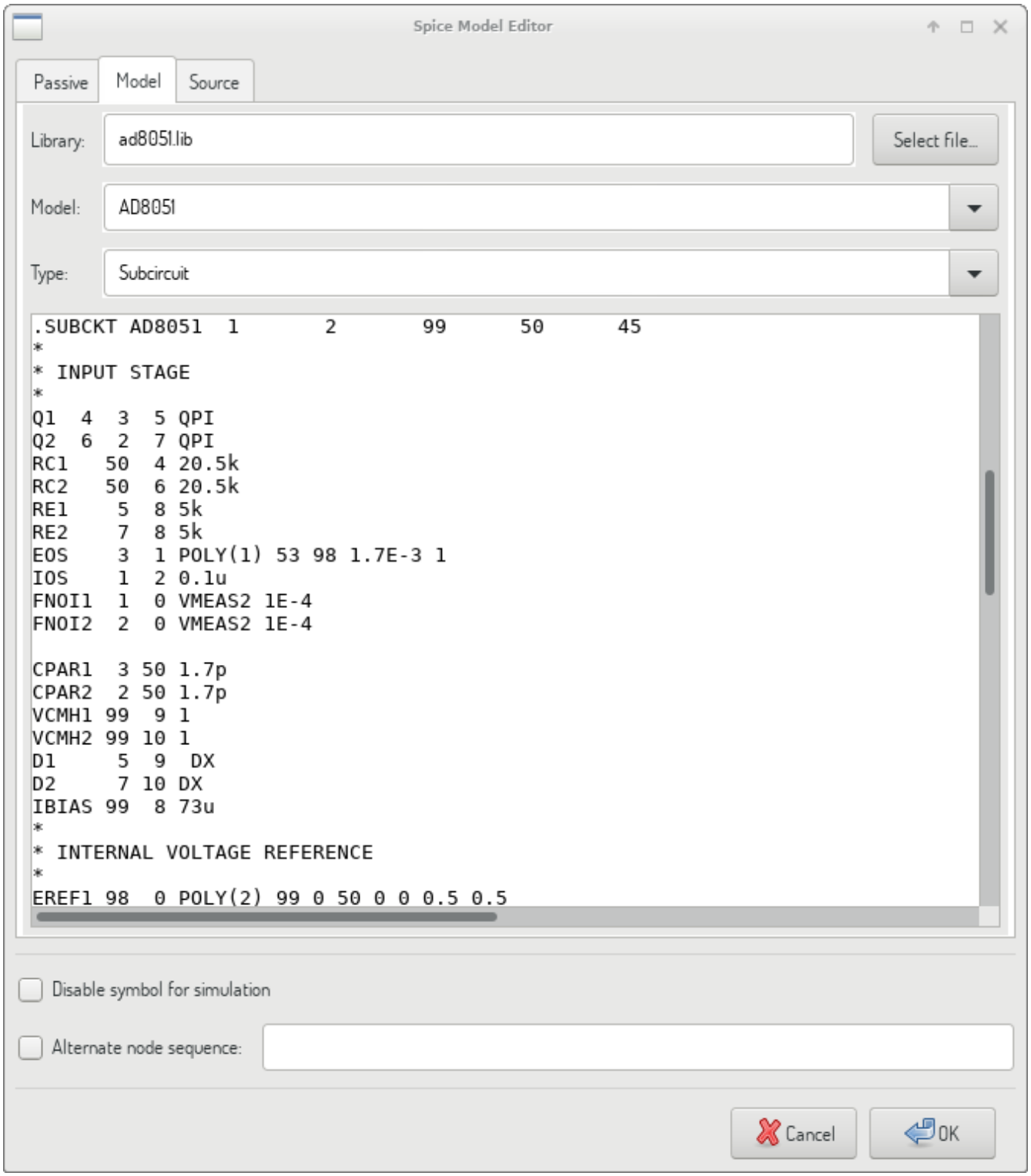


Type	Selects the device type (resistor, capacitor or inductor).
Value	Defines the device property (resistance, capacitance or inductance). The value may use common Spice unit prefixes (as listed below the text input field) and should use point as the decimal separator. Note that Spice does not correctly 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.

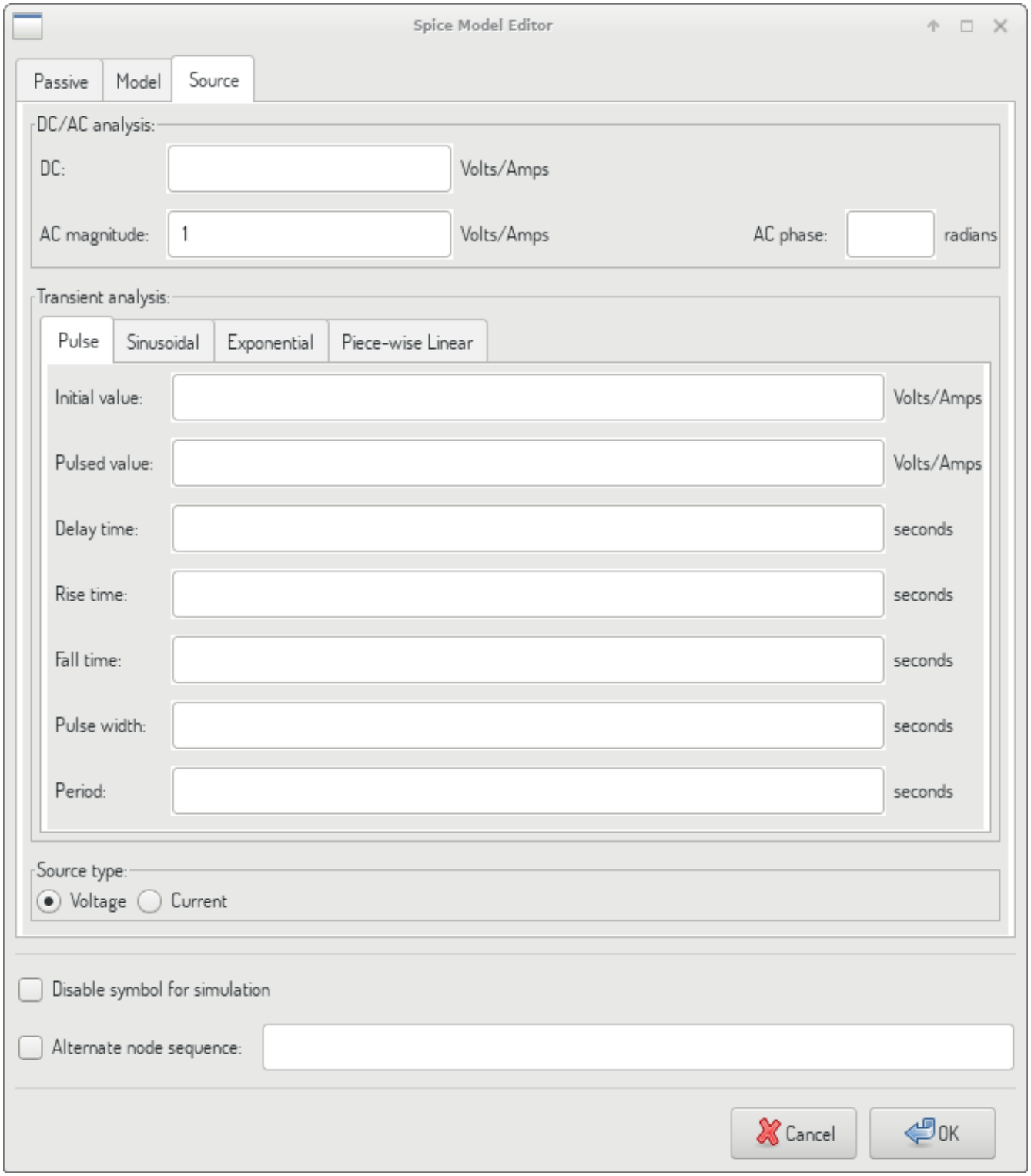


File	Path to a Spice library file. This file is going to be used by the simulator, as it is added using *.include* directive.
Model	Selected device model. When a file is selected, the list is filled with available models to choose from.
Type	Selects model type (subcircuit, BJT, MOSFET or diode). Normally 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.



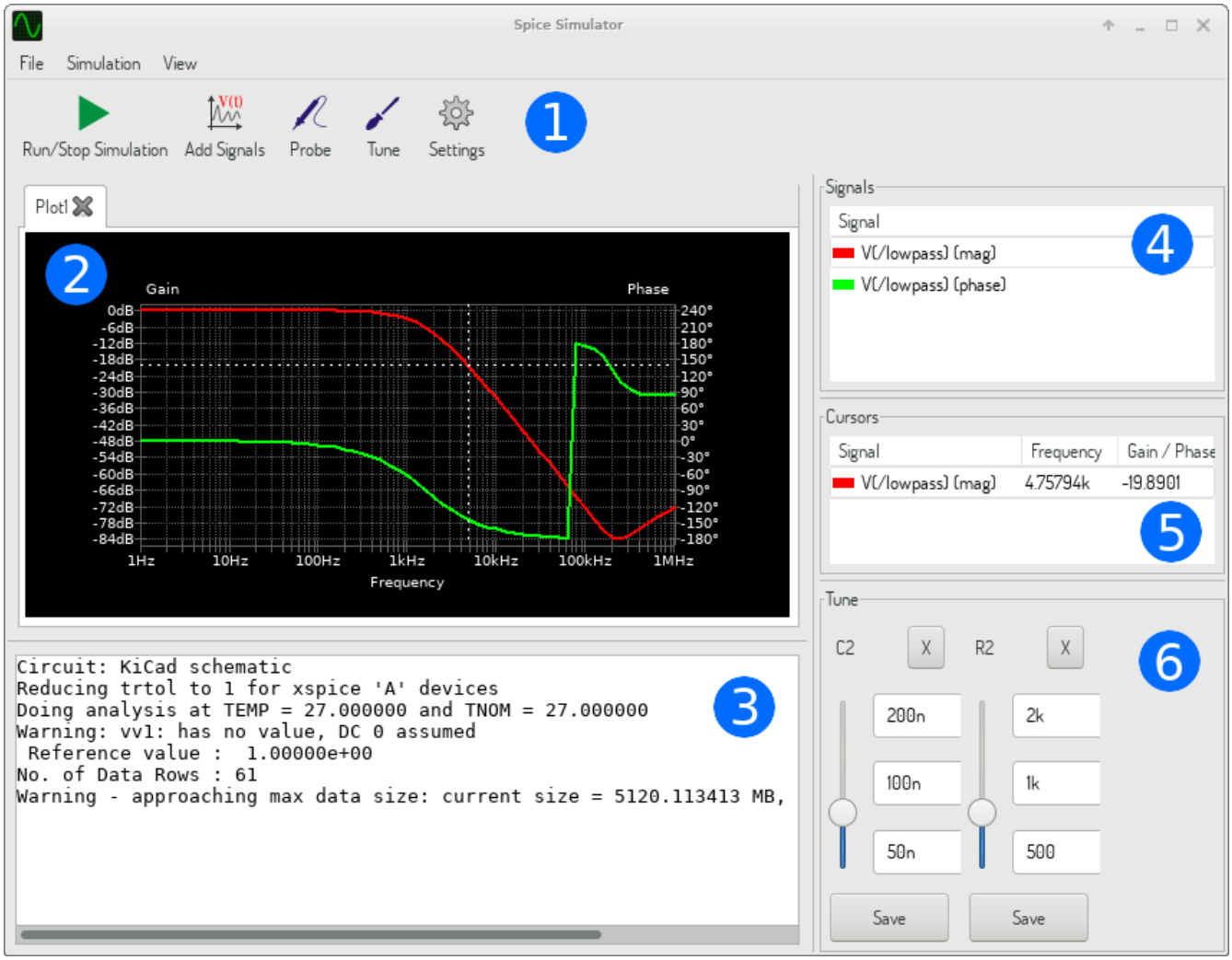
Refer to the ngspice documentation, chapter 4 (Voltage and Current Sources) for more details about sources.

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.



The dialog is divided into several sections:

- Toolbar
- Plot panel
- Output console
- Signals list
- Cursors list
- Tune panel

Menu

File

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.

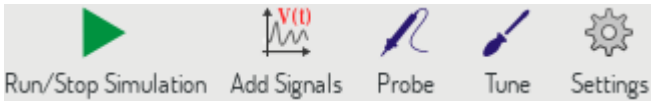
Simulation

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 Probe tool.
Tune component value	Start the Tuner tool.
Show SPICE Netlist...	Open a dialog showing the generated netlist for the simulated circuit.
Settings...	Open the simulation settings dialog.

View

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.

Toolbar



The top toolbar provides access to the most frequently performed actions.

Run/Stop Simulation	Start or stop the simulation.
Add Signals	Open a dialog to select signals to be plotted.
Probe	Start the schematics Probe tool.
Tune	Start the Tuner tool.
Settings	Open the simulation settings dialog.

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 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

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.

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

Cursors list

Shows the list of cursors and their coordinates. Each signal may have one cursor displayed. Cursors visibility is set using the Signals list.

Tune panel

Displays components picked with the 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.

Tuner tool

Tuner tool lets the user pick components for tuning.

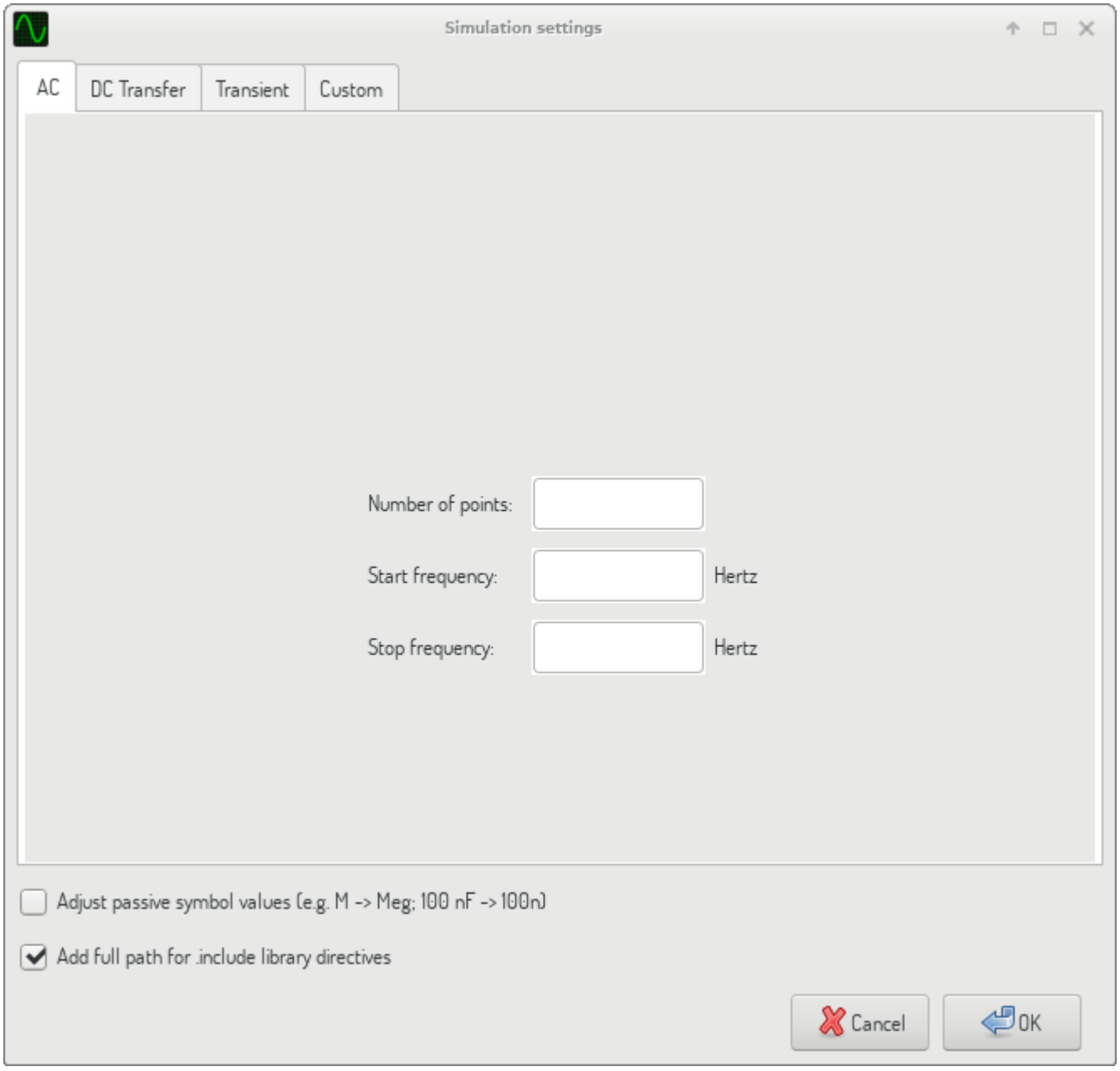
To select a component for tuning, click on one in the schematics editor when the tool is active. Selected components will appear in the Tune panel. Only passive components might be tuned.

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.

Simulation settings



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 ngspice documentation, chapter 1.2.

An alternative way to configure a simulation is to type 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:

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.