



Eeschema

Reference manual

Copyright

This document is Copyright © 2010–2011 by its contributors as listed below. You may distribute it and/or modify it under the terms of either the GNU General Public License (<http://www.gnu.org/licenses/gpl.html>), version 3 or later, or the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), version 3.0 or later.

All trademarks within this guide belong to their legitimate owners.

Contributors

Jean-Pierre Charras.

Feedback

Please direct any comments or suggestions about this document to the kicad mailing list:
<https://launchpad.net/~kicad-developers>

Publication date and software version

Published on October 24, 2011. Based on LibreOffice 3.4.

Note for Mac users

The Kicad support for the Apple OS X operating system is experimental.

Headings:

1 - Introduction.

1.1 - Description

1.2 - Technical overview.

2 - General commands.

2.1 - Access to the commands

2.2 - Commands with the MOUSE

2.2.1 - Basic commands

2.2.2 - Operations on blocks

2.3 - Hot keys

2.4 - Selecting the grid size.

2.5 - Selecting ZOOM.

2.6 - Displaying the coordinates of the cursor

2.7 - Menu Bars

2.8 - Upper toolbar

2.9 - Right toolbar icons

2.10 - Left toolbar icons

2.11 - Pop-up menus and fast editing of elements

3 - Main menu

3.1 - File menu

3.2 - Preferences menu

3.2.1 - Preferences!

3.2.2 - Hotkeys sub menu:

3.2.3 - Preferences menu / Libs and Dir

3.2.4 - Preferences menu / Colours

3.2.5 - Preferences / Options

3.2.6 - Preferences / Language

3.3 - Help menu

4 - General Toolbar

4.1 - Sheet Management

4.2 - Options of the schematic editor

4.2.1 - General options

4.2.2 - Template fields names:

4.3 - Research tool

4.4 - Netlist tool

4.5 - Annotation tool

4.6 - E.R.C tool

4.6.1 - Main folder/dialog

4.6.2 - Options folder/dialog

4.7 - BOM (Bill of Material) tool

4.8 - Import tool for back-annotation

5 - Schematic Creation / Editing.

5.1 - Definitions.

5.2 - General considerations.

5.3 - The development chain.

5.4 - Component placement / editing.

5.4.1 - Find and place a component.

5.4.2 - Power ports.

5.4.3 - Component Editing / Modification (already placed component).

5.4.3.1 - Component modification.

5.4.3.2 - Text fields modification.

5.5 - Wires,Buses,Labels,Power ports.

5.5.1 - Basics.

5.5.2 - Connections (Wires and Labels).

5.5.3 - Connections (Buses).

5.5.3.1 - Bus members.

5.5.3.2 - Connections between bus members.

5.5.3.3 - Global Connections between buses.

5.5.4 - Power ports connection.

5.5.5 - "No Connection" symbols.

Eeschema

- [5.6 - Complements.](#)
 - [5.6.1 - Comments.](#)
 - [5.6.2 - Title block.](#)
- [6 - Hierarchical schematics.](#)
 - [6.1 - Presentation.](#)
 - [6.2 - Navigation in the Hierarchy](#)
 - [6.3 - Local, hierarchical and global labels .](#)
 - [6.3.1 - properties:](#)
 - [6.3.2 - Notes:](#)
 - [6.4 - Hierarchy Creation. Headlines.](#)
 - [6.5 - Sheet symbol.](#)
 - [6.6 - Connections: Hierarchy pins.](#)
 - [6.7 - Connections: Hierarchical labels.](#)
 - [6.7.1 - Labels, Hierarchical Labels Global labels and Invisible Power Pins.](#)
 - [6.7.1.1 - Simple labels.](#)
 - [6.7.1.2 - Hierarchical labels.](#)
 - [6.7.1.3 - Invisible Power pins.](#)
 - [6.7.2 - Global Labels:](#)
 - [6.8 - Complex Hierarchy](#)
 - [6.9 - Flat Hierarchy](#)
- [7 - Automatic classification Annotation.](#)
 - [7.1 - Purpose:](#)
 - [7.2 - Example.](#)
 - [7.2.1 - Annotation Order:](#)
 - [7.2.2 - Annotation Choice:](#)
- [8 - Design verification \(E.R.C.\)](#)
 - [8.1 - Overview.](#)
 - [8.2 - Use.](#)
 - [8.3 - Example of ERC :](#)
 - [8.4 - Displaying diagnostics:](#)
 - [8.5 - Powers and Power flags:](#)
 - [8.6 - Configuration](#)
 - [8.7 - ERC report file.](#)
- [9 - Netlist Creation.](#)
 - [9.1 - Overview.](#)
 - [9.2 - Netlist formats.](#)
 - [9.3 - Examples.](#)
 - [9.4 - Note.](#)
 - [9.4.1 - Precautions.](#)
 - [9.4.2 - PSPICE netlists.](#)
 - [9.5 - Other formats, using « plugins »](#)
 - [9.5.1 - Init the Dialog window:](#)
 - [9.5.2 - Command line format:](#)
 - [9.5.3 - The converter or the sheet style \(plug in\)](#)
 - [9.5.4 - Format of the intermediate netlist text file:](#)
- [10 - Plot and Print](#)
 - [10.1 - Overview](#)
 - [10.2 - Common commands:](#)
 - [10.3 - Plot / Plot HPGL](#)
 - [10.3.1 - Sheet size selection](#)
 - [10.3.2 - Offset adjustments](#)
 - [10.4 - Plot / Plot Postscript](#)
 - [10.5 - Plot / Plot SVG](#)
 - [10.6 - Plot / Plot DXF](#)
 - [10.7 - Print:](#)
- [11 - LibEdit : Components Management](#)
 - [11.1 - General information about libraries](#)
 - [11.1.1 - Libraries :](#)
 - [11.1.2 - Management Menus](#)
 - [11.2 - Components overview](#)
 - [11.3 - Load a component for editing](#)
 - [11.3.1 - Main Toolbar](#)
 - [11.3.2 - Library selection and maintenance](#)

Eeschema

- [11.3.3 - Component selection and saving](#)
 - [11.3.3.1 - Selection](#)
 - [11.3.3.2 - Save](#)
 - [11.3.3.3 - Transfer into another library](#)
 - [11.3.3.4 - Cancellation of component editing](#)
- [11.4 - Component creation](#)
 - [11.4.1 - Create of a new component](#)
 - [11.4.2 - Creation based on another component](#)
 - [11.4.3 - Editing of the main characteristics](#)
 - [11.4.4 - Multi-part components](#)
- [11.5 - Component design](#)
 - [11.5.1 - Graphic elements membership options](#)
 - [11.5.2 - Geometrical graphic elements](#)
 - [11.5.3 - Graphic elements of text type](#)
- [11.6 - Pin creation and editing](#)
 - [11.6.1 - Pins overview](#)
 - [11.6.2 - Multi-part components, double representation.](#)
 - [11.6.3 - Pins: basic option](#)
 - [11.6.4 - Pins: Defining characteristics](#)
 - [11.6.5 - Pins shapes](#)
 - [11.6.6 - Pins : Electric types](#)
 - [11.6.7 - Pins : global modifications](#)
 - [11.6.8 - Pins : Multi-part components and double representations](#)
- [11.7 - Field Editing](#)
- [11.8 - Power port symbols : Creation](#)
- [12 - LibEdit : Complements](#)
 - [12.1 - Overview](#)
 - [12.2 - Positioning the anchor](#)
 - [12.3 - Alias](#)
 - [12.4 - Fields:](#)
 - [12.5 - Component documentation](#)
 - [12.5.1 - Keywords](#)
 - [12.5.2 - Component documentation \(Doc\)](#)
 - [12.5.3 - Associated documentation file \(DocFileName\)](#)
 - [12.5.4 - Footprint filtering for CvPcb](#)
 - [12.6 - Symbol library](#)
 - [12.6.1 - Export/Create a symbol](#)
 - [12.6.2 - Import a symbol](#)
- [13 - Viewlib](#)
 - [13.1 - Role](#)
 - [13.2 - Main screen](#)
 - [13.3 - Viewlib Toolbar](#)
- [14 - Customize Netlists and BOM](#)
 - [14.1 - The Intermediate Netlist:](#)
 - [14.1.1 - Schematic sample:](#)
 - [14.1.2 - The Intermediate Netlist file sample:](#)
 - [14.2 - Conversion to a new format:](#)
 - [14.3 - XSLT approach:](#)
 - [14.3.1 - Sample to create a Pads-Pcb netlist file](#)
 - [14.3.2 - Sample to create a Cadstar netlist file](#)
 - [14.3.2.1 - The Style-Sheet file:](#)
 - [14.3.2.2 - The output file:](#)
 - [14.3.3 - Sample to create a OrcadPCB2 netlist file](#)
 - [14.3.3.1 - The Style-Sheet file:](#)
 - [14.3.3.2 - The output file:](#)
 - [14.3.4 - Using Eeschema plugins interface:](#)
 - [14.3.4.1 - Init the Dialog window:](#)
 - [14.3.4.2 - Parameters:](#)
 - [14.3.4.3 - The command line:](#)
 - [14.3.4.4 - Command line format:](#)
 - [14.4 - The Intermediate Netlist structure:](#)
 - [14.4.1 - General structure:](#)
 - [14.4.2 - The header section:](#)

Eeschema

[14.4.3 - The components section:](#)

[14.4.3.1 - Note about time stamps for components:](#)

[14.4.4 - The libparts section:](#)

[14.4.5 - The libraries section:](#)

[14.4.6 - The nets section:](#)

[14.5 - More info about xsltproc:](#)

[14.5.1 - Introduction](#)

[14.5.2 - Synopsis](#)

[14.5.3 - Command Line Options](#)

[14.5.4 - Return values](#)

[14.5.5 - More Information](#)

Headings:

[1 - Introduction.](#)

[1.1 - Description](#)

[1.2 - Technical overview.](#)

1 - Introduction.

1.1 - Description

Eeschema is powerful schematic capture software, available under the following operating systems :

- LINUX
- WINDOWS XP/2000/W7

Whatever the system used, the generated files are completely compatible from one system to another.

Eeschema is "integrated" software because all of the functions of drawing, control, layout, library management and access to the PCB design software are carried out from within Eeschema.

It also allows **hierarchical** drawings, using **multi-sheets diagrams** .

Eeschema handles:

- Flat hierarchies.
- Simple hierarchies.
- Complex hierarchies.

It is intended to work with printed circuit software such as **PCBNEW**, to which it will provide the **Netlist** file, which describes the electrical connections of the PCB to design.

Eeschema also integrates a component editor which allows the creation, editing, and visualization of components, as well as the handling of the symbol libraries (Import, export, addition and deletion of library components).

Eeschema also integrates the following additional but essential functions needed for modern schematic capture software :

- Design rules check (**D.R.C.**) for the automatic control of incorrect connections, the inputs of components left unconnected...
- Generation of the layout files in format POSTSCRIPT or HPGL
- Generation of the layout files on a local printer.
- Bill of Materials generation.
- **Netlist** generation for PCB layout or simulation software.

1.2 - Technical overview.

This 32 bit software is limited only by the memory size available.

There is thus no real limitation to the component count, number of component pins, connections, sheets...

Eeschema allows simple or multi sheet diagrams.

In the case of multi sheets diagrams, the representation is hierarchical, and the access to each sheet is then immediate.

Eeschema can uses for multi sheet diagrams:

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

The maximum size of the drawings is always adjustable from A4 format to A0 and from A to E format.

Headings:

[2 - General commands.](#)

[2.1 - Access to the commands](#)

[2.2 - Commands with the MOUSE](#)

[2.2.1 - Basic commands](#)

[2.2.2 - Operations on blocks](#)

[2.3 - Hot keys](#)

[2.4 - Selecting the grid size.](#)

[2.5 - Selecting ZOOM.](#)

[2.6 - Displaying the coordinates of the cursor](#)

[2.7 - Menu Bars](#)

[2.8 - Upper toolbar](#)

[2.9 - Right toolbar icons](#)

[2.10 - Left toolbar icons](#)

[2.11 - Pop-up menus and fast editing of elements](#)

2 - General commands.

2.1 - Access to the commands

You can reach the various commands 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).
- Clicking on the mouse buttons (important complementary commands); In particular a right click opens a contextual menu, depending on the element under the cursor (Zoom, grid and edition of the elements).
- Function keys of the keyboard (**F1**, **F2**, **F3**, **F4**, **Insert** and **space** keys).
Particularly :
The “Escape” key often allows the canceling of a command in progress.
The “Insert” key allows the duplication of the last element created.

Here are the various possible accesses to the commands.

2.2.2 - Operations on blocks

You can move, drag, copy and delete selected areas in all Eeschema menus. Areas are selected with the left mouse button. The command is completed with the release of the button. By holding one of the keys “Shift”, “Ctrl”, or the 2 keys “Shift and Ctrl”, during selection this results in the copying, dragging or deletion of the selected area.

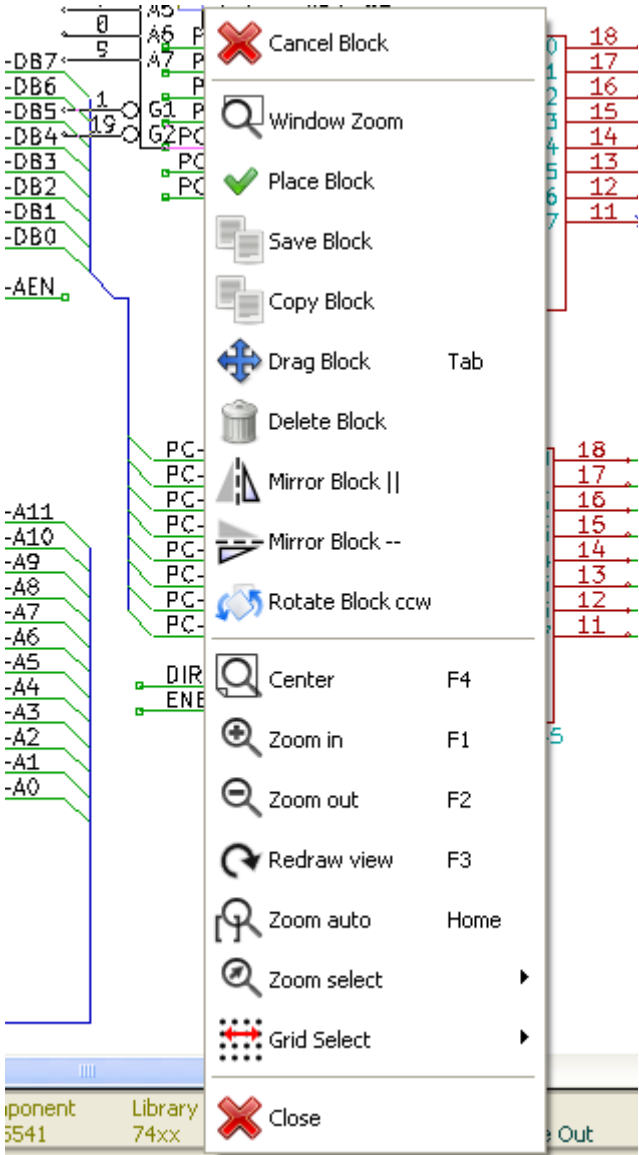
Commands summary:

left mouse button	Move selection.
Shift + left mouse button	Copy selection.
Ctrl + left mouse button	Drag selection.
Control + Shift + left mouse button	Delete selection.

The command is executed at button release.
During selection move :

- Click again to place back the elements.
- Click right button to cancel.

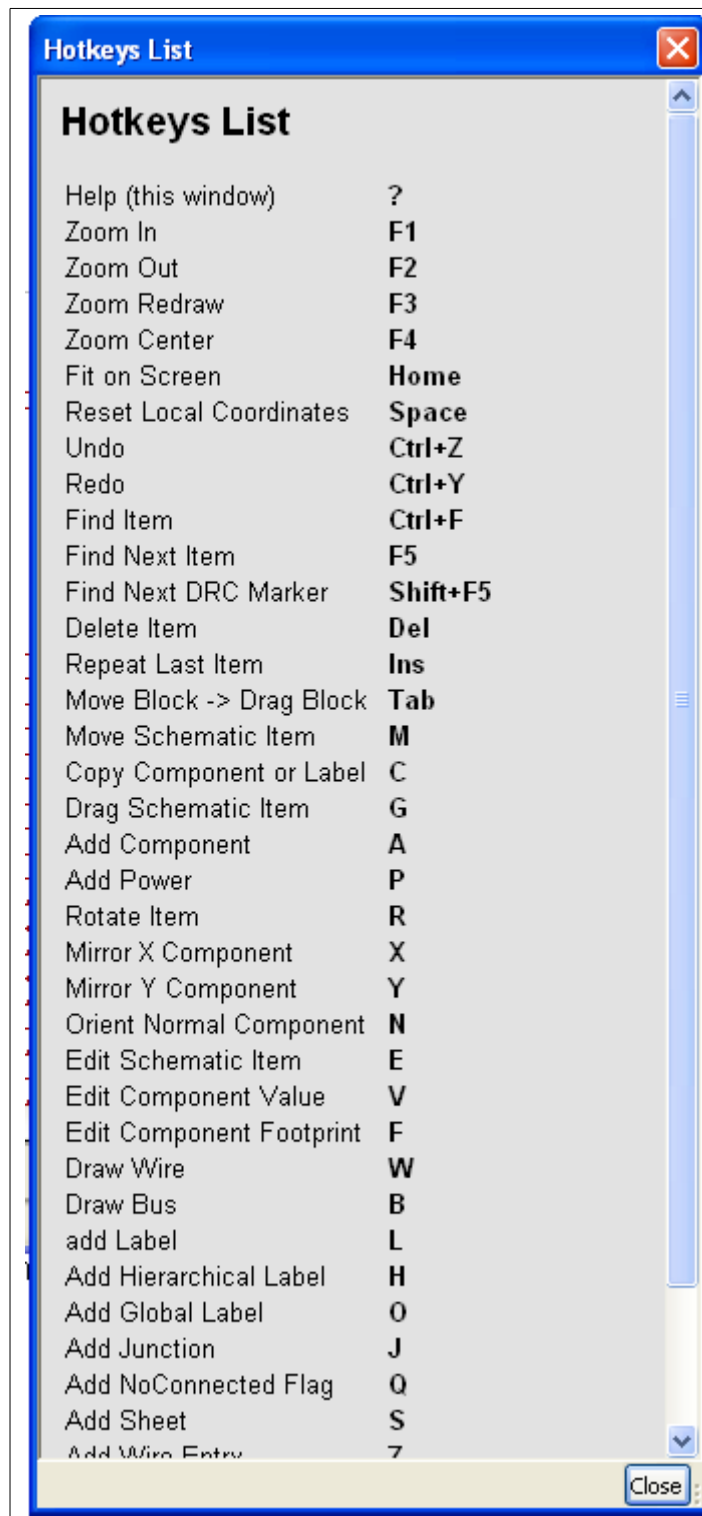
If a move bloc command has started, an other command bloc can be slected by Pop Up menu (mouse, right button):



2.3 - Hot keys

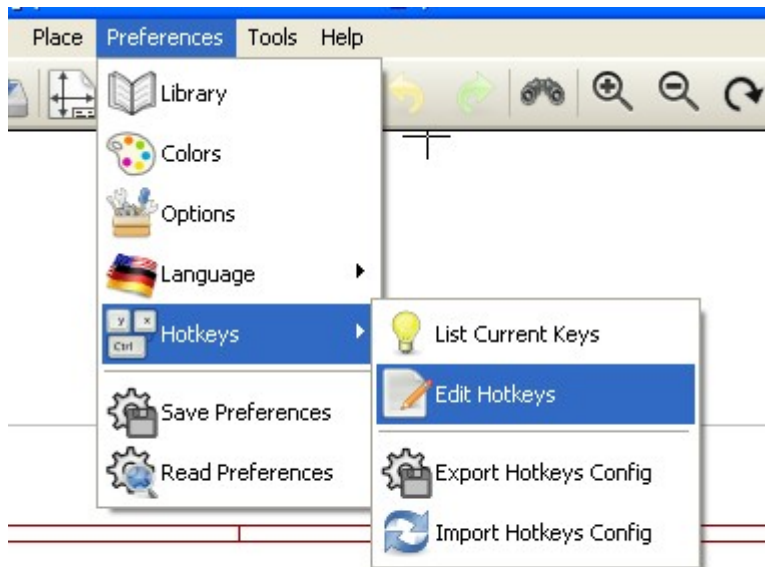
The hot keys are not case sensitive.

- The ? key displays the current hot keys list.
- The Preference menu manage the hot keys



Here is an example

The hot keys can be programmed by users
To do it: call the hotkey editor:



And edit hotkey list (it is commented).

2.4 - Selecting the grid size.

The cursor moves on a grid, which can be displayed or not (this grid is always displayed in the library management menus).

You can change the grid size via the pop-up menu or the **Preferences/Options** menu.

The default grid size is 50 mil (0.050 ") or 1,27 millimeters.

One can also work with the average (20 mil) or fine grid(10 mil).

This is not recommended for usual work.

This average or fine grid is especially intended to design or handle components with large numbers of pins (several hundreds).

2.5 - Selecting ZOOM.

To change "ZOOM":

- Right click to open the Pop-up menu and select the desired zoom.
- Or use the function keys:
 - **F1**: Zoom in.
 - **F2**: Zoom out.
 - **F3**: Redraw.
 - **F4**: Center around the cursor
Or simple click on the mouse middle button (without moving the mouse)
 - **Window Zoom**: Mouse drag, with the middle button.
 - **Mouse wheel**: Zoom in / Zoom out
 - **SHIFT+Mouse wheel**: Up/down panning
 - **CTRL+Mouse wheel**: Left/Right panning

2.6 - Displaying the coordinates of the cursor

The display units are in inches (inch or ") or millimetres.

However, Eeschema always works internally with 1/1000 of an inch.

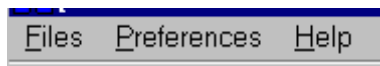
The following information is displayed at the bottom right hand side of the window :

- The zoom factor.
- The absolute position of the cursor.
- The relative position of the cursor.
- The relative co-ordinates (x, y) can be reset with the space bar.
- The coordinates posted will then relate to this point.



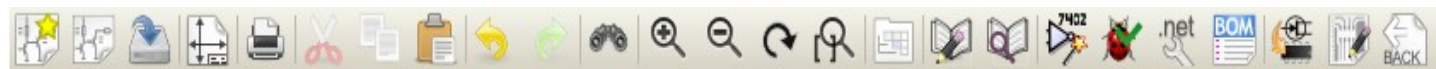
2.7 - Menu Bars
















This menu allows the opening and saving of schematics, the program configuration, and it also contains the help menu.











2.8 - Upper toolbar

This toolbar gives access to the main functions of EESchema.

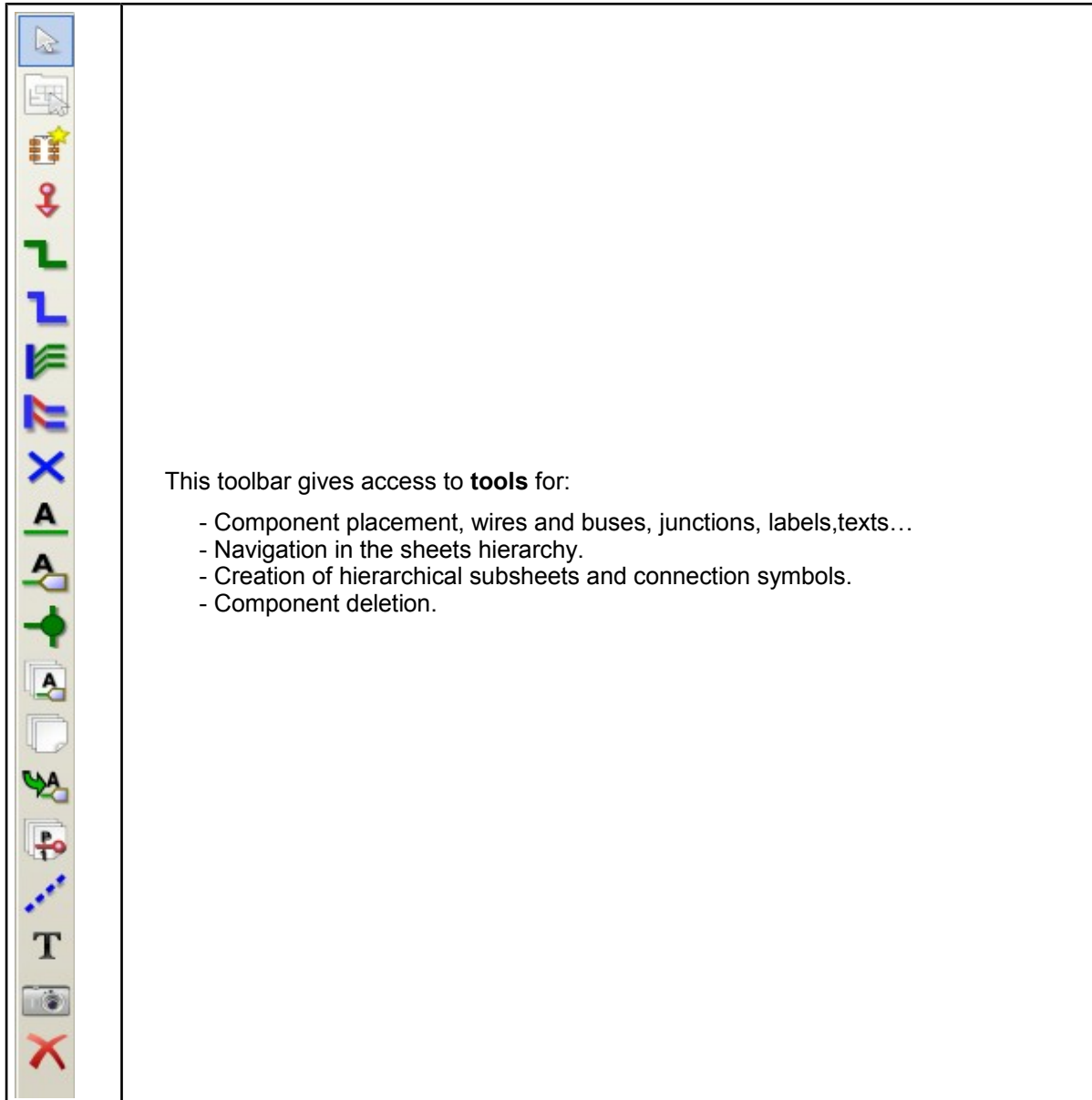


	Create a new schematic.
	Open a schematic.
	Save complete schematic (with the whole hierarchy).
	Select the sheet size and title block editing.
	Open print menu.
	Remove the selected elements during a move block .
	Copy selected elements in the clipboard during a move block .
	Copy last selected element or block in the current sheet.
	Undo: Cancel the last change (up to 10 levels).
	Redo (up to 10 levels).
	Call the menu of components localization and texts.
	Zoom in and out, around the center of screen.
	Redraw of the screen and optimal Zoom.
	Call the “navigator”, to display the tree structure of the diagram hierarchy (if it contains sub sheets) and the immediate selection of any sheet of the hierarchy.
	Call component editor Libedit (Examination, modification, and editing of library components).





















Eeschema

	Display libraries (Viewlib).
	Component annotation.
	ERC (Electrical Rules Check) : automatic checking of electrical connections.
	Creation of the netlist (Pcbnew , Spice format).
	Generate the BOM (Bill of materials) and/or hierarchical labels.
	Call CVPCB .
	Call PCBNEW .
	Import a stuff file from Cvpcb (fill the footprint field of components)

2.9 - Right toolbar icons



The detailed use of these tools is described in the chapter “ **Diagram Creation/Editing** ”.
An outline of their use is given below.

	Stop the order or tool in progress.
	Navigation in the hierarchy: this tool makes it possible to open the subsheet of the displayed schematic (click in the symbol of this subsheet), or to go back up in the hierarchy (click in a free area of the subsheet)
	Call the component placement menu.
	"Powers" placement menu.
	Wire placement.
	Bus placement.
	Wire to bus connections. These elements have only a decorative role and do not allow connection; thus they should not be used for connections between wires.
	Bus to bus connections. They can only connect two buses between themselves.
	"No connection" symbols. These are to be placed on component pins which are not to be connected. This is useful in the ERC function to check if pins are intentionally left not connected or are missed.
	Local label placement. Two wires may be connected with identical labels in the same sheet . For connections between two different sheets, you have to use global symbols.
	Global label placement. All global labels are connected (even between different sheets).
	Junction placement. To connect two crossing wires, or a wire and a pin, when it can be ambiguous. (i.e. if an end of the wire or pin is not connected to one of the ends of the other wire).
	Hierarchical label placement. This makes it possible to place a connection between a sheet and the root sheet which contains this sheet symbol.
	Hierarchical subsheet symbol placement (resizable rectangle). You have to specify the file name to save the data of this "subsheet".
	Global label importation from subsheet, in order to create a connection on a subsheet symbol. Global labels are supposed to be already placed in this subsheet. For this hierarchy symbol, the created connection points are equivalent to a traditional component pin, and must be wired.
	Global label creation in subsheets to create connection points. This function is similar to the previous one which does not require already defined global symbols.
	Lines for framings... Only decorative, and does not perform a connection.
	Placement of comment text. Only decorative.
	Insert a bitmap image.
	Delete selected element. if several superimposed elements are selected, the priority is given to the smallest (in the decreasing priorities : junction, NoConnect, wire, bus, text, component). This also applies to hierarchical sheets. Note: the "Undelete" function of the general toolbar allows you to cancel last deletions.

2.10 - Left toolbar icons

This toolbar manages the display options :

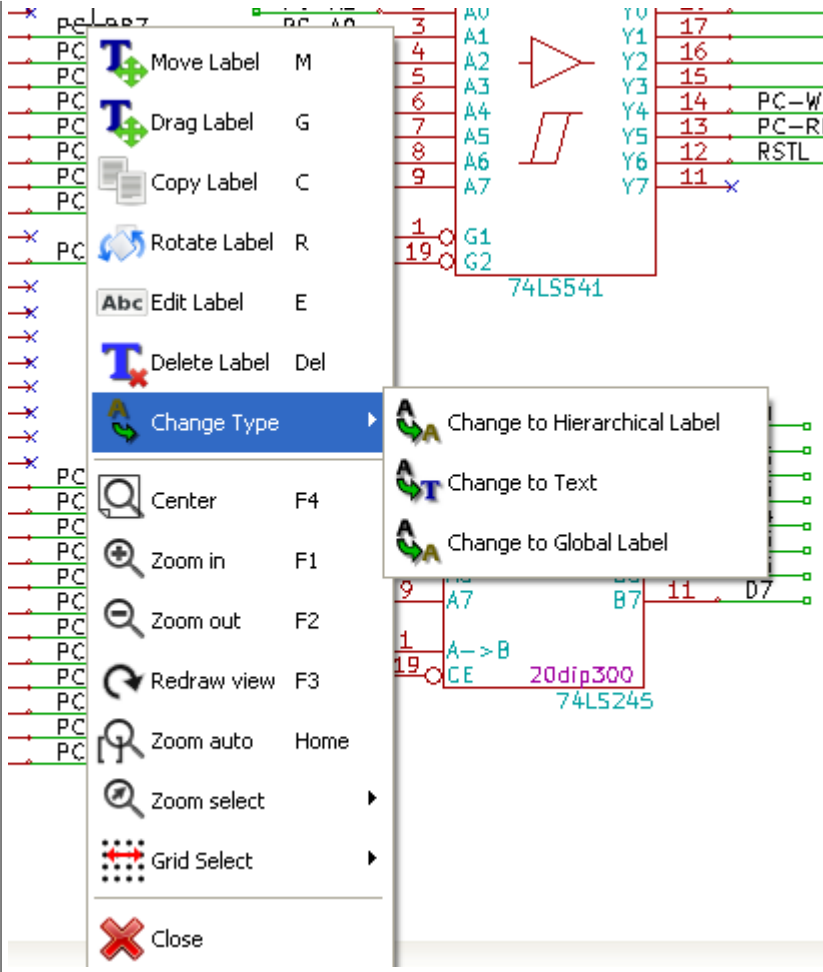
- Grid.
- Units.
- Cursor.
- “Invisible” pins.
- And allowed directions of wires and buses.

2.11 - Pop-up menus and fast editing of elements

A right click opens a pop-up menu whose content depends on the element selected (if there is one).
You have immediate access to:

- Zoom factor.
- Grid adjustment.
- And according to the case, editing of the most usually modified parameters.

Pop-up without selected element.



Editing of a label.

The screenshot displays the 'Edit Component' menu in a PCB design application. The main menu is open, with 'Edit Component' highlighted. A secondary menu is open for 'Edit Component', showing options: Edit, Value, Reference, Footprint, and Edit with Library Editor. The background shows a PCB layout with various components and their pin connections.

Editing of a component.

Headings:

[3 - Main menu](#)

[3.1 - File menu](#)

[3.2 - Preferences menu](#)

[3.2.1 - Preferences!](#)

[3.2.2 - Hotkeys sub menu:](#)

[3.2.3 - Preferences menu / Libs and Dir](#)

[3.2.4 - Preferences menu / Colours](#)

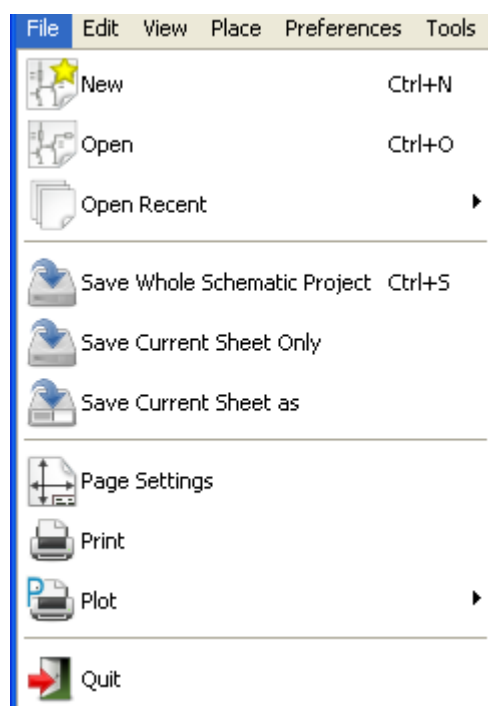
[3.2.5 - Preferences / Options](#)

[3.2.6 - Preferences / Language](#)

[3.3 - Help menu](#)

3 - Main menu

3.1 - File menu

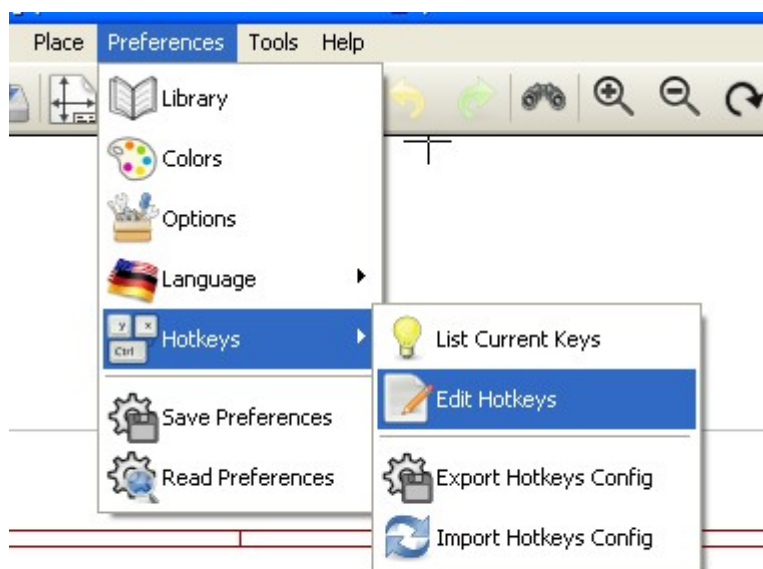


New	Clear current schematic and initialize a new one
Open	Load a schematic hierarchy
Open Recent	Open a list of recent opened files for loading
Save Whole Schematic project	Save current sheet and all its hierarchy.
Save Current Sheet Only	Save current sheet, but not others in a hierarchy.
Save Current sheet as...	Save current sheet with a new name.
Print	Access to print menu (See also chap "Print and Plot").

Plot	Plot in Postscript HPGL or SVF format (See chap "Print and Plot").
Quit	Quit without saving.

3.2 - Preferences menu

3.2.1 - Preferences!

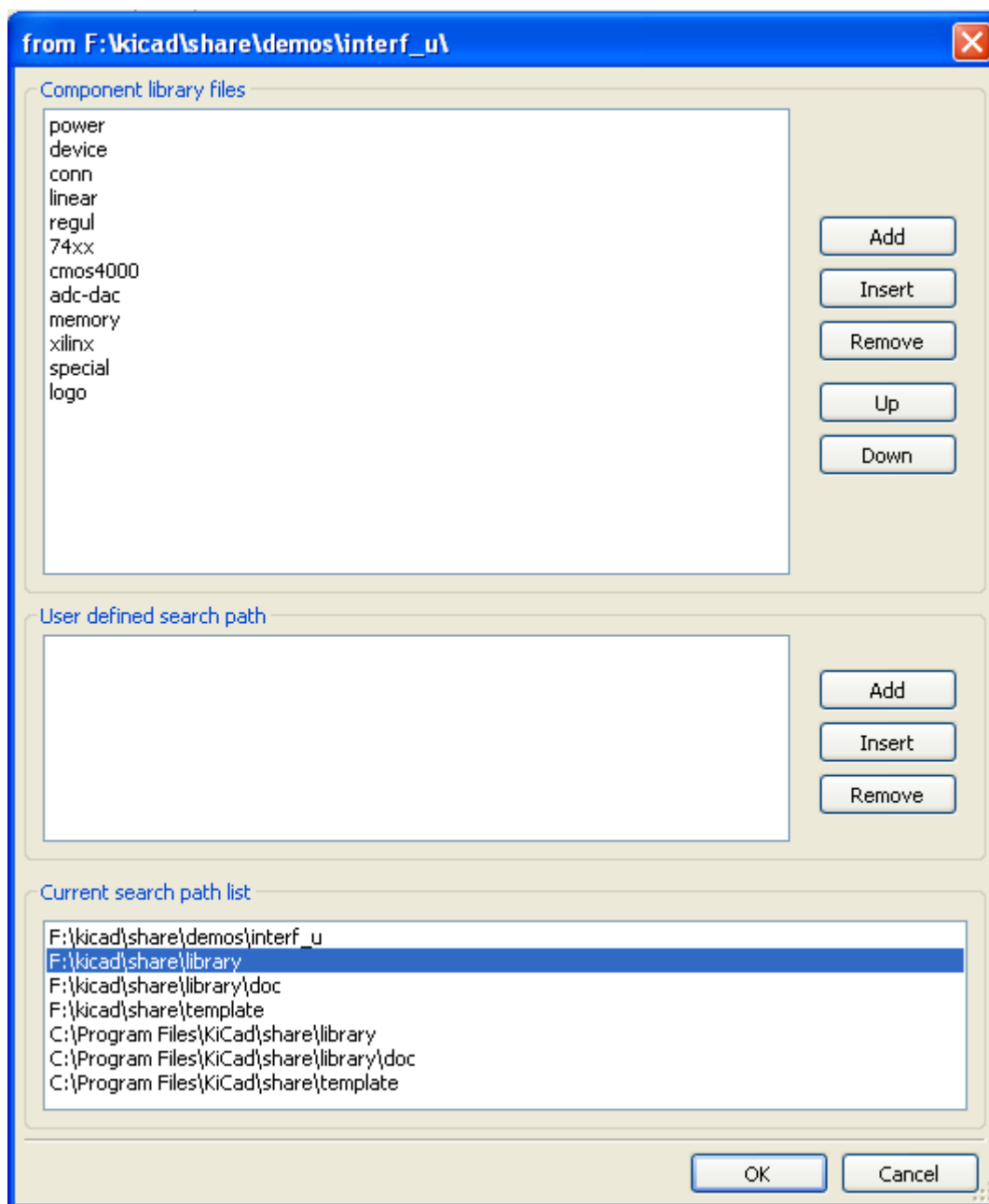


Library	Select libraries and the library's path
Colors	Select colours.
Options	Display options (Units, Grid size.).
Language	Access to the current list of translations. Use default. Mainly for translators and developers
Read preferences Save preferences	Read and Save configuration file.
Hotkeys	Access to the hot keys menu

3.2.2 - Hotkeys sub menu:

List Current Keys	Shows the current hotkeys, Same as ?
Edit Hokeys	Launch the hotkeys editor
Export Hotkeys Config	Create a hotkeys config file.
Import Hotkeys Config	Read a previously exported hotkeys config file.

3.2.3 - Preferences menu / Libs and Dir



Eeschema configuration is essentially :

- Library's path.
- Library's list.

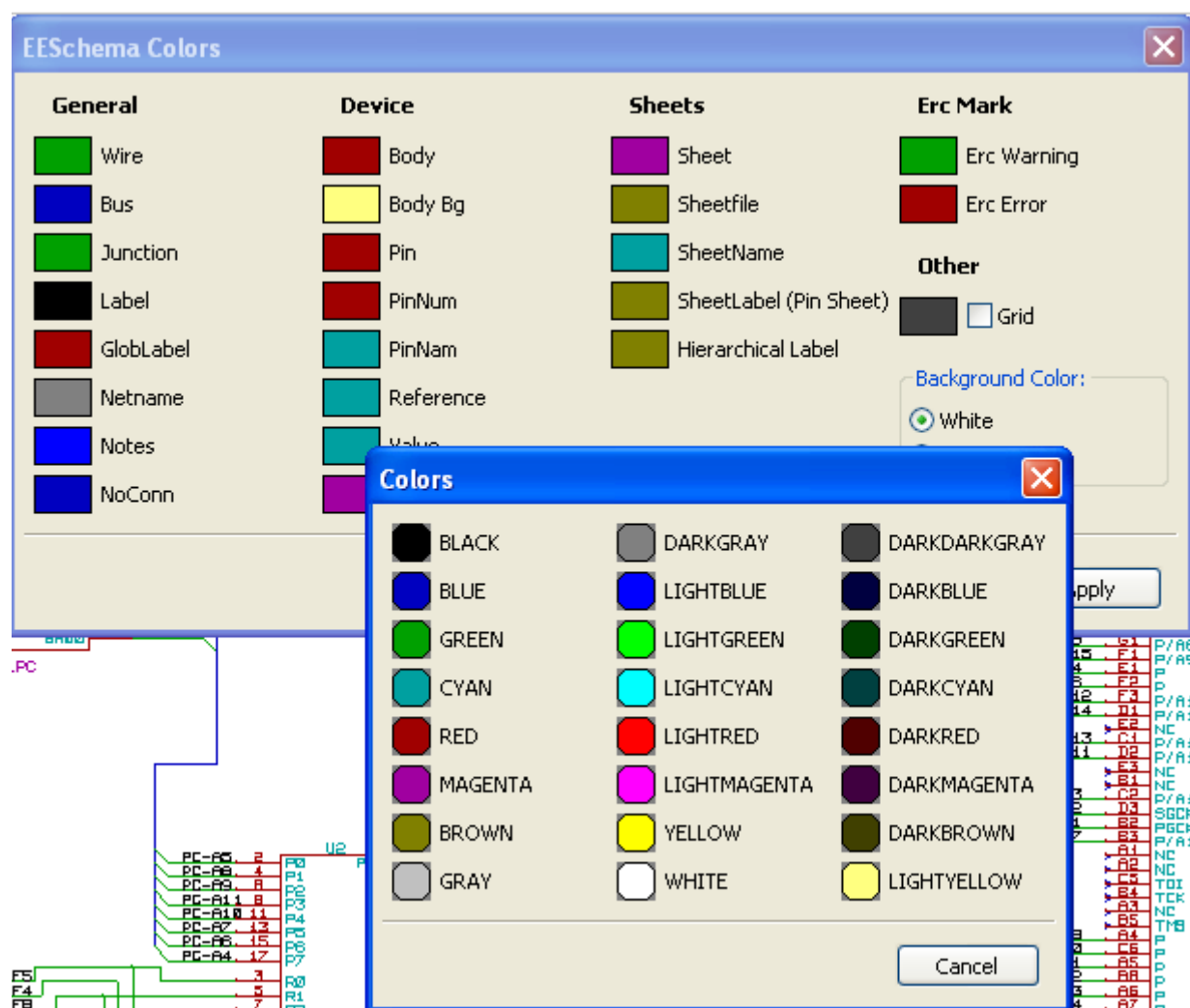
The configuration parameters are saved in the .pro file.

Different configuration files in different directories are possible.

Eeschema seeks and uses by decreasing priorities :

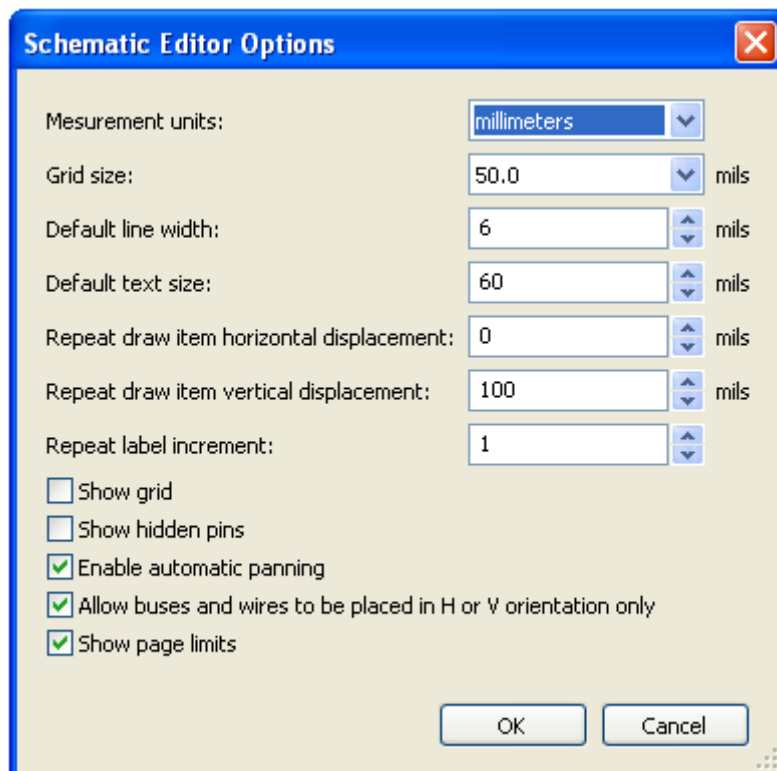
1. The configuration file (**project>.pro**) in the current directory.
2. The **kicad.pro** configuration file in the **kicad** directory. This file can thus be the **default configuration**.
3. Default values if no file is found. It will at least then be necessary to fill out the list of libraries to load, and then save the configuration.

3.2.4 - Preferences menu / Colours



Various drawing elements, colour selection and background colour (black or white only).

3.2.5 - Preferences / Options



Measurement units:	Select the display and the cursor coordinate units (Inches or Millimetres).
Grid Size:	Grid size selection. One must work with normal grid (0,050 inches or 1,27 mm). <u>Smaller grids are used for component building.</u>
Default line width:	Pen size used to draw objects that do not have a specified pen size.
Default text size:	Value used when creating new texts or labels
Repeat draw item horizontal displacement	shift value on X axis during element duplication (usual value 0) (after placing an item like a component, label or wire, a duplication is made by the <i>Inser</i> key)
Repeat draw item horizontal displacement	shift value on Y axis during element duplication (usual value is 0,100 inches or 2,54 mm)
Repeat label increment:	Increment during duplication of texts ending in a number, such as bus members (usual value 1 or - 1).
Show Grid :	If checked : display grid.
Show hidden pins:	Display invisible (or <i>hidden</i>) pins . If checked, allows the display of power pins.
Enable automatic panning	If checked, automatically shifts the window if the cursor leaves the window, during wire drawing, or element moving.
Allow buses and wires to be placed in H or V orientation only	If checked buses and wires can be only vertical or horizontal. Else buses and wires can be placed in any direction.
Show page limit	If checked, shows the page limits on screen.

3.2.6 - Preferences / Language

Use default mode.

Other languages are available mainly for maintenance purpose.

3.3 - Help menu

Access to on-line help (this document) and also for checking the current version of Eeschema (Eeschema about).

Headings:

[4 - General Toolbar](#)

[4.1 - Sheet Management](#)

[4.2 - Options of the schematic editor](#)

[4.2.1 - General options](#)

[4.2.2 - Template fields names:](#)

[4.3 - Research tool](#)

[4.4 - Netlist tool](#)

[4.5 - Annotation tool](#)

[4.6 - E.R.C tool](#)

[4.6.1 - Main folder/dialog](#)

[4.6.2 - Options folder/dialog](#)

[4.7 - BOM \(Bill of Material\) tool](#)

[4.8 - Import tool for back-annotation](#)

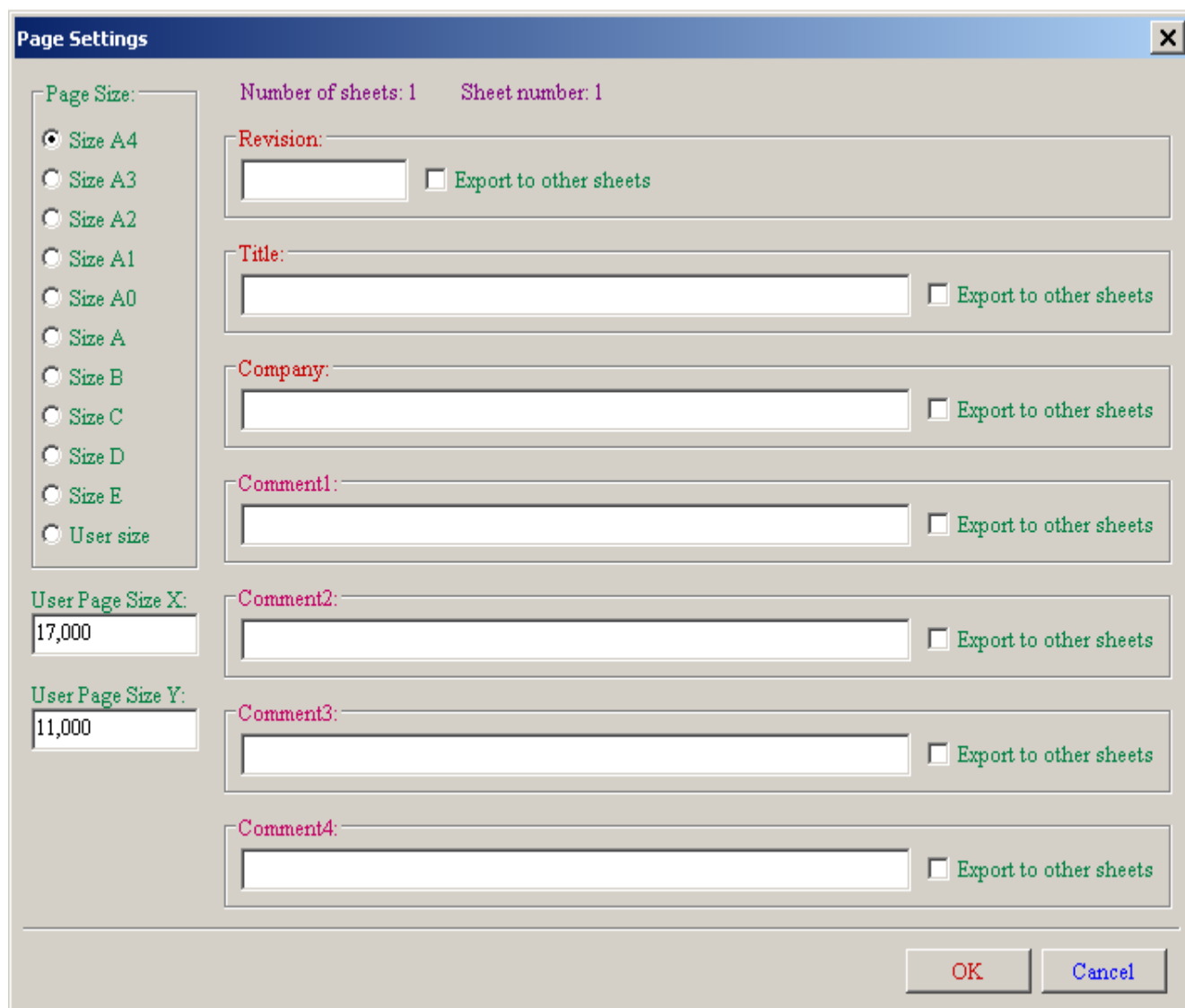
4 - General Toolbar

4.1 - Sheet Management



Access to page settings:

You can define here the sheet size and some texts in title block.



The image shows the 'Page Settings' dialog box in Eeschema. It has a title bar with a close button (X). The dialog is divided into several sections. On the left, there is a 'Page Size:' section with a list of radio buttons: Size A4 (selected), Size A3, Size A2, Size A1, Size A0, Size A, Size B, Size C, Size D, Size E, and User size. Below this are two text boxes: 'User Page Size X:' with the value '17,000' and 'User Page Size Y:' with the value '11,000'. On the right, there are two status labels: 'Number of sheets: 1' and 'Sheet number: 1'. Below these are four groups of input fields, each with a label in red text and a checkbox: 'Revision:' with a text box and 'Export to other sheets' checkbox; 'Title:' with a text box and 'Export to other sheets' checkbox; 'Company:' with a text box and 'Export to other sheets' checkbox; and 'Comment1:', 'Comment2:', 'Comment3:', and 'Comment4:' each with a text box and 'Export to other sheets' checkbox. At the bottom right are 'OK' and 'Cancel' buttons.

Note:

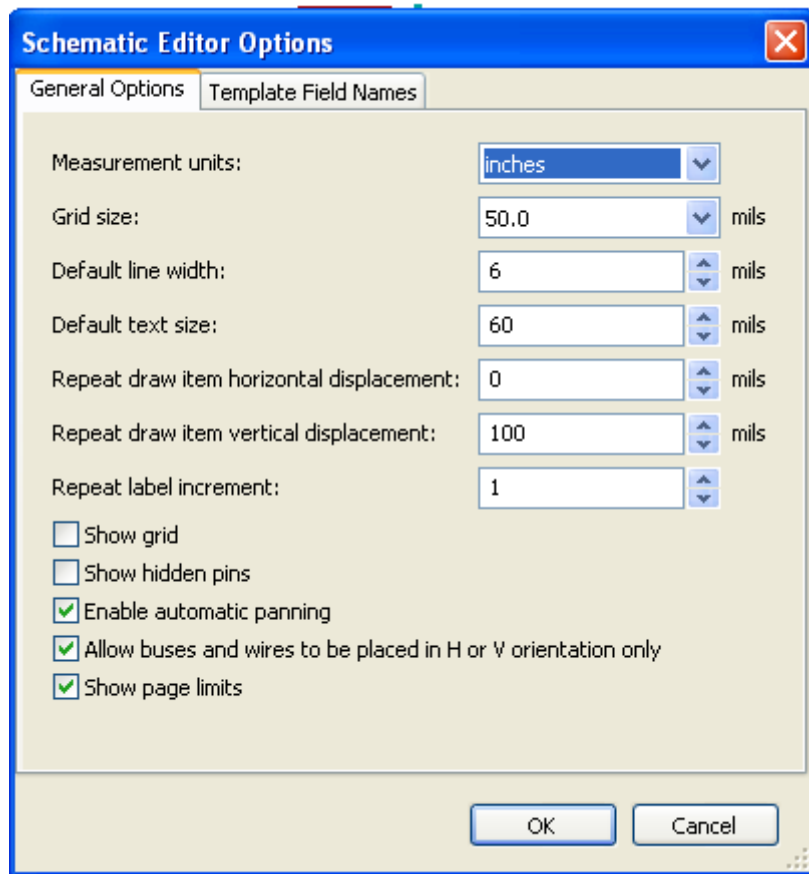
The date is automatically updated.

Total number of sheets and sheet number are automatically updated.

4.2 - Options of the schematic editor

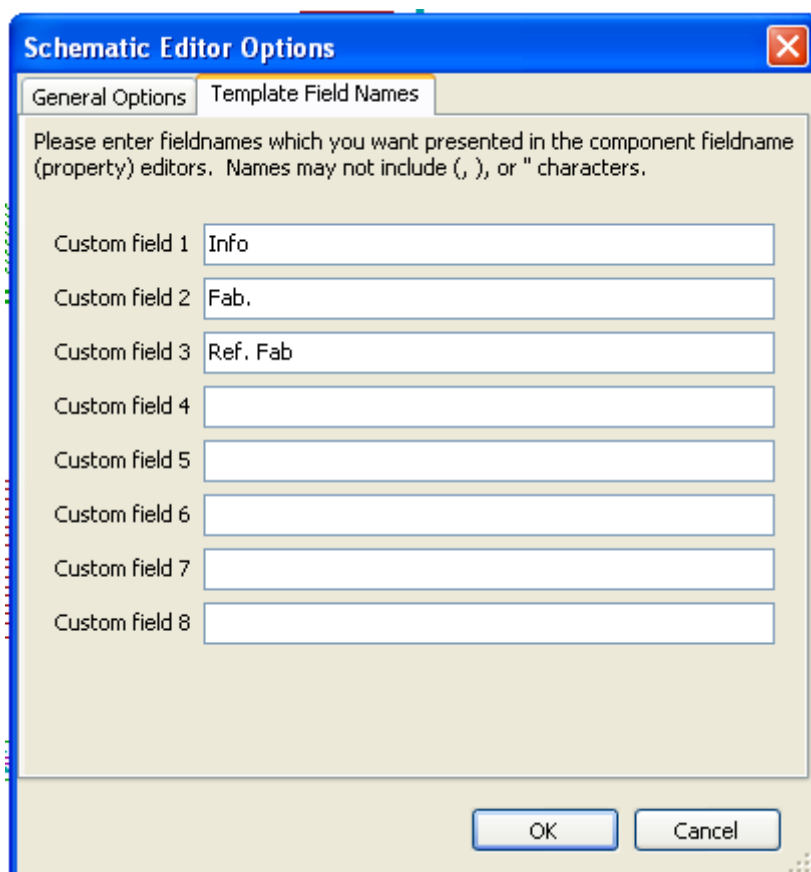
4.2.1 - General options

They are options relative to drawing are the repeat command (**Insert** key):



4.2.2 - Template fields names:

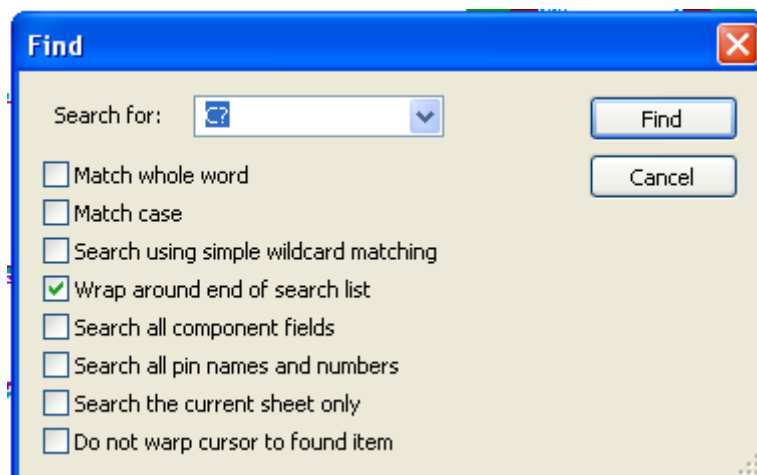
You can define here custom fields that are always existing in each component (even if the field is left empty in a component)



4.3 - Research tool



To access this tool.



You can search a component, a value, or a text string in the current sheet or in the whole hierarchy. The cursor will be positioned (on demand) on the found element, in the concerned sub-sheet.

4.4 - Netlist tool



Gives access to the netlist tool.

This netlist file can apply to the whole hierarchy (usual option), or only to the current sheet (the netlist is then partial, but this option can be useful for some software). In a multisheet hierarchy, any local label is known only inside the sheet to which it belongs.

Eeschema

Thus the label TOTO of sheet 3 is different from the label TOTO of sheet 5 (if no connection has been intentionally introduced to connect them).

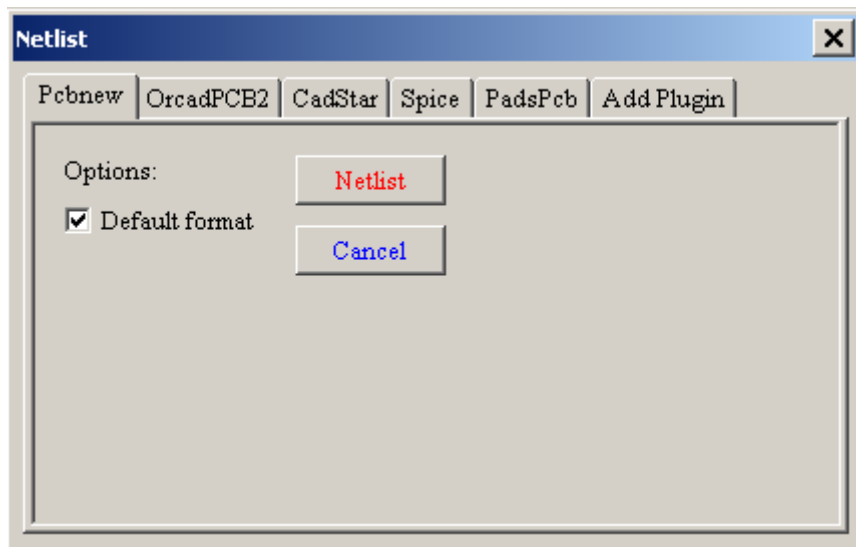
This is due to the fact that the sheet number (updated by the annotate command) is associated with the local label. In the previous example, the first label TOTO is actually TOTO_3, and the second label TOTO is actually TOTO_5. This association can be inhibited if it is wished, but be aware of possible undesired connections.

Notice 1:

Label lengths have no limitations in Eeschema, but the software exploiting the generated netlist can be limited on this point.

Notice 2:

Avoid spaces in the labels, because they then appear as separated words. It is not a limitation of Eeschema, but of many netlist formats, which often suppose 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, for the Spice simulator.

External plugins can be launch to extend netlist formats list (a PadsPcb Plugin was added here)

4.5 - Annotation tool

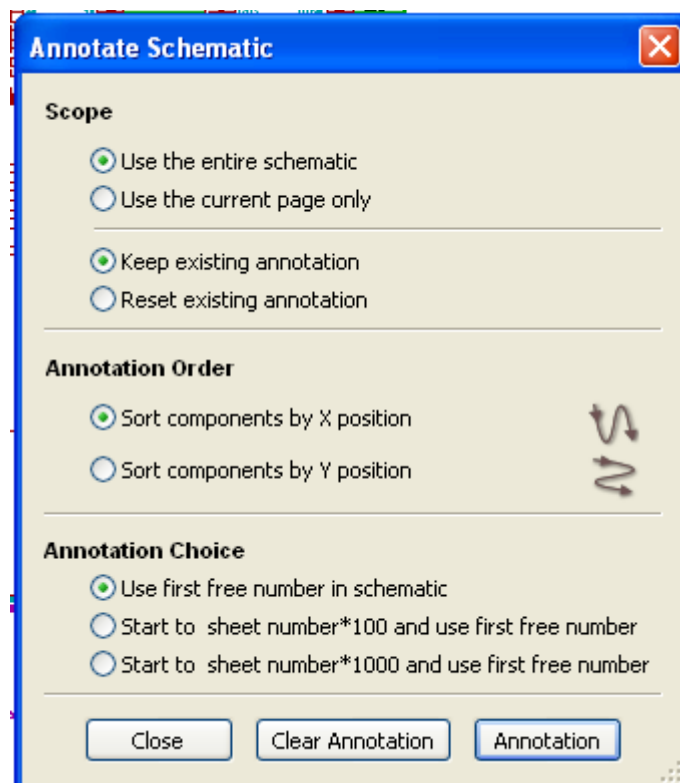


To access the annotation tool.

This tool performs an auto incrementing designation of the 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 the sheets are re-annotated (usual Option).

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 (usual option).

Reset existing annotation:

Unconditional annotation, all the components will be re-annotated (this option is to be used when there are duplicated references).

Order

Sorting option to set the annotation numbers to components

4.6 - E.R.C tool



To access E.R.C tool.

This tool performs a design verification (known as **Electrical Rules Check**).

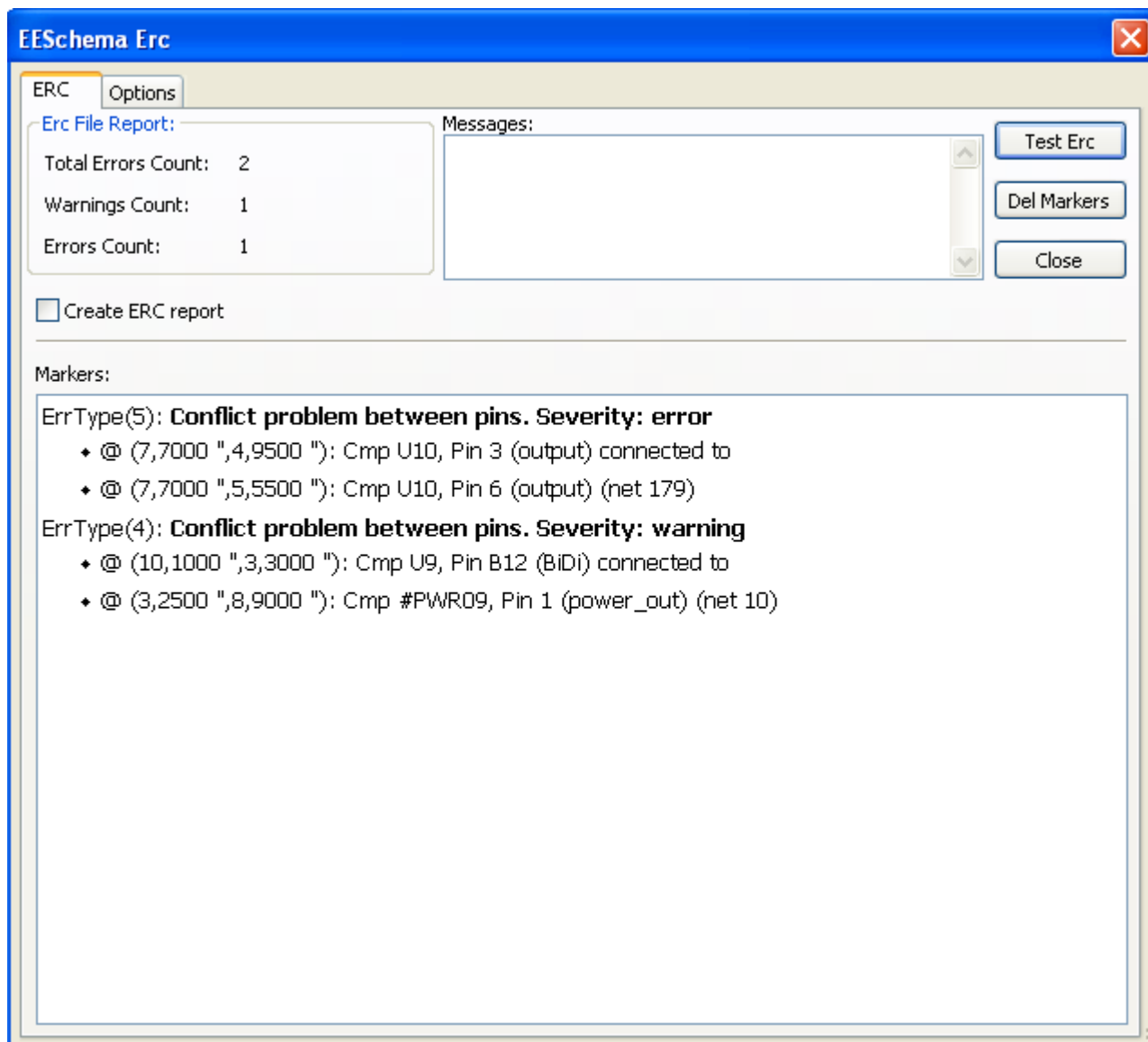
This function is particularly useful to detect forgotten connections, and inconsistencies.

Eeschema places markers on the pins or labels able to pose a problem.

The diagnosis can then be given by left clicking on the marker.

An errors file can also be generated.

4.6.1 - Main folder/dialog



Errors are displayed in the **Erc Diags** dialog box :

- Errors and warnings count.
- Errors count.
- Warnings count.

Option:

- Create ERC report : check this option to generate an ERC report file.

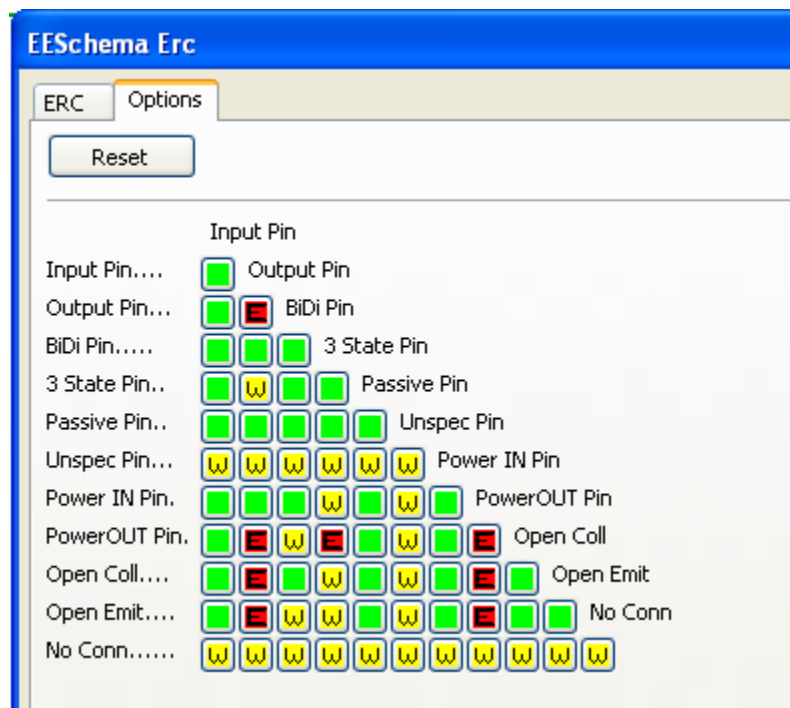
Commands:

- *Test Erc*: to perform an Electrical Rules Check.
- *Del Markers* : to remove all ERC markers.
- *Close* : to exit this dialog box.

Note:

- When clicking on an error message, jump to the corresponding marker in schematic.

4.6.2 - Options folder/dialog



This Setup ERC dialog box allows you to establish connectivity rules between pins; you can choose between 3 options for each case :

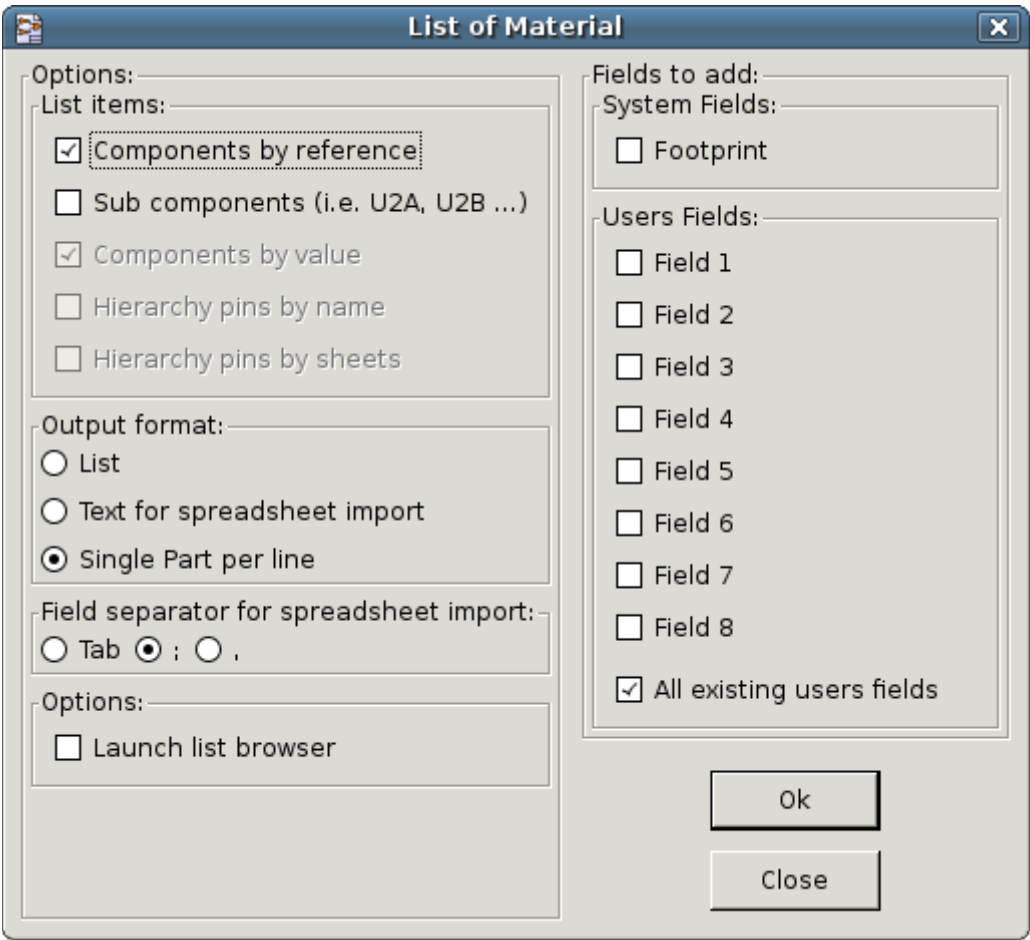
- No error
- Warning
- Error

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

4.7 - BOM (Bill of Material) tool



This menu allows the generation of a file listing of the components and/or hierarchical connections (global labels).



Components can be sorted by :

- Reference.
- Value.

And multi-part components can be detailed.

Global labels can be sorted by :

- Alphabetical classification
- Sub-sheet

Different sortings can be used simultaneously.

Options are:

Components by Reference	Bill of Material sorted by Reference.
Component by Value	Bill of Material sorted by Value.
Sub components	The BOM shows every device of multi-part components (ex U2A, U2B...).
Hierarchy Pins by name	Hierarchical connections sorted alphabetically.
Hierarchy Pins by Sheet	Hierarchical connections sorted by sheet number.
List	Creates a plain text file ready to print
Text for spreadsheet import	Creates an ASCII file which can be easily imported in a spreadsheet
Single Part per line	Creates a csv file combining components with the same Value into a single line, listing reference designators comma separated.
Launch list browser	Run the text editor to load and display the BOM list file after creating.

Eeschema

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:

Component Properties

Options

Unit: 1

Orientation (Degrees):
☒ 0
☐ +90
☐ 180
☐ -90

Mirror:
☒ Normal
☐ Mirror ---
☐ Mirror |

Chip Name: CRYSTAL

☐ Convert

Reset to Library Defaults

Fields

Référence	X1
Valeur	8MHz
Module	HC-18UH
Documentation	
Info	
Fab.	KONY
Ref. Fab	123-456

Add Field
Delete Field
Move Up

Text Justification:

Horiz. Justify: ☐ Left, ☒ Center, ☐ Right

Vert. Justify: ☐ Bottom, ☒ Center, ☐ Top

Visibility

☐ Show
☐ Rotate

Style:
☒ Normal
☐ Italic
☐ Bold
☐ Bold Italic

Field Name: Ref. Fab

Field Value: 123-456

Size ("): 0.060

Pos X ("): 0.000

Pos Y ("): 0.000

OK Cancel

Using the BOM Format **Single Part per line** only requires the Component properties be edited for one Component on the schematic and not all the Components with that same Value.

However, if there exists different parts, both with a Value of 33K, may be one is 1/10 W and another is 1/4 W, or may have a different footprint, specify one as 33K and the other as 33KBig and these will be listed as different parts.

The output is in format than can be imported into a spreadsheet where cost numbers (or optionally even Field4) may be added to derive a board cost and assist with parts procurement.

4.8 - Import tool for back-annotation



This feature allows a schematic to be captured, make footprint assignments using Cvpcb's table and browser tools, then export that assignment back to the schematic.

This function reads a **.stf** file previously created by Cvpcb and initialize the footprint field (F3) of components.

This is not useful for Pcbnew, but useful to add the footprint field when creating the Bill of Material and the netlist.

This feature keeps the component footprint/reference information in a single source file, the schematic, which is the source for the netlist and makes the **.cmp** file redundant.

The footprint assignments will appear in any future netlist export from Eeschema.

This is useful when using some netlist formats.

Note for Pcbnew:

When Pcbnew does not find a **.cmp** file corresponding to the **.net** file, it uses the component footprint/reference found in the **.net** file.

But using the **.cmp** file is better, because if the designer changes a footprint assignment from Pcbnew, the corresponding **.cmp** file is also updated.

Headings:

[1 - Schematic Creation / Editing.](#)

[1.1 - Definitions.](#)

[1.2 - General considerations.](#)

[1.3 - The development chain.](#)

[1.4 - Component placement / editing.](#)

[1.4.1 - Find and place a component.](#)

[1.4.2 - Power ports.](#)

[1.4.3 - Component Editing / Modification \(already placed component\).](#)

[1.4.3.1 - Component modification.](#)

[1.4.3.2 - Text fields modification.](#)

[1.5 - Wires,Buses,Labels,Power ports.](#)

[1.5.1 - Basics.](#)

[1.5.2 - Connections \(Wires and Labels\).](#)

[1.5.3 - Connections \(Buses\).](#)

[1.5.3.1 - Bus members.](#)

[1.5.3.2 - Connections between bus members.](#)

[1.5.3.3 - Global Connections between buses.](#)

[1.5.4 - Power ports connection.](#)

[1.5.5 - "No Connection" symbols.](#)

[1.6 - Complements.](#)

[1.6.1 - Comments.](#)

[1.6.2 - Title block.](#)

5 - Schematic Creation / Editing.

5.1 - Definitions.

A schematic can be represented on a single sheet, but it will mostly require several sheets.

A schematic represented on several sheets is then called **hierarchical**, and all these sheets (each one represented by its own file) constitutes for Eeschema a **project**.

A project consists of a main schematic, called the root schematic, and sub-sheets constituting a hierarchy.

In order to find every file of the project, you will have to follow drawing rules which will be described hereafter.

In the following, when we talk about project, we will be referring to both single sheet and hierarchical multi sheets.

An additional special chapter develops the use of the hierarchy and its characteristics.

5.2 - General considerations.

A schematic designed with Eeschema is more than a simple graphic representation of an electronic device.

It is normally the entry point of a development chain which allows :

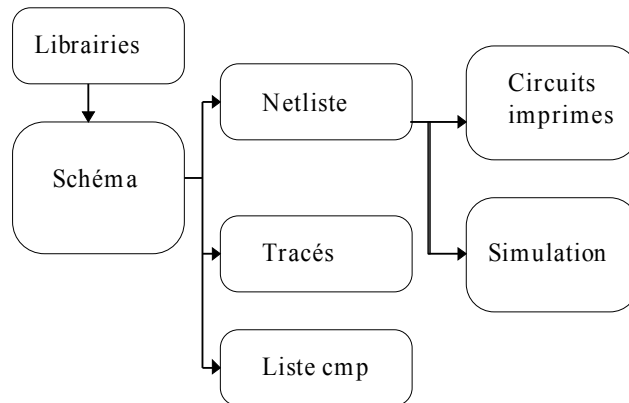
- The control of the electrical rules (control E.RC.) that allows the detection of errors or omissions in the schematic.
- The automatic generation of the bill of material.
- The netlist generation for simulation software such as Pspice.
- The netlist generation for printed circuits board design (PCBNEW). The consistency check between the schematic and the printed circuit board is then automatic and instantaneous.

In order to benefit from these possibilities, you will have to respect certain constraints and conventions, to avoid nasty surprises and errors.

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

For clearness in the schematic, you can place purely graphical elements like bus entries, comments, and dotted lines to draw frames.

5.3 - The development chain.



The schematic software uses component libraries.

In addition to the schematic design file, the netlist file is particularly important because it is used by the other design softwares.

A netlist file gives the list of the components and connections resulting from the schematic.

There is (unfortunately for the user) a great number of netlist formats, some are more known. It is the case of the Spice format for example.

5.4 - Component placement / editing.

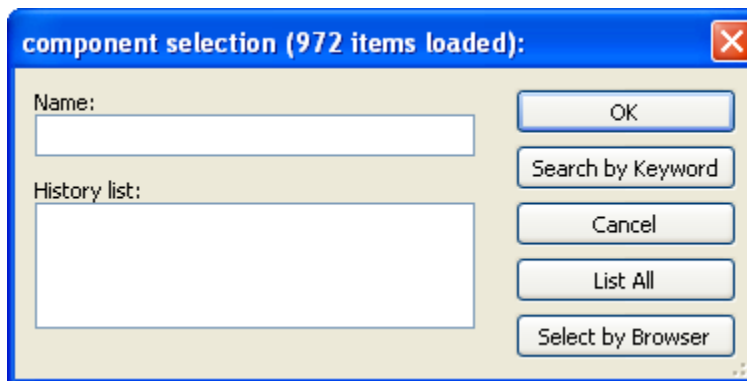
5.4.1 - Find and place a component.



To load a component, use the tool

To place a new component, click at the place you want to draw it.

A dialog box allows you to type the name of the module to load.



The dialog box displays the last two elements loaded.

If you type *, or if you select the button “**list all**”, Eeschema will display the libraries list, and then the available components.

If you type the symbol “=” followed by key words, EESchema will then display a list of components according to **all the** key words.

You can also list a selection : for example if you enter **LM2** *, all the component's names starting with **LM2** will be listed

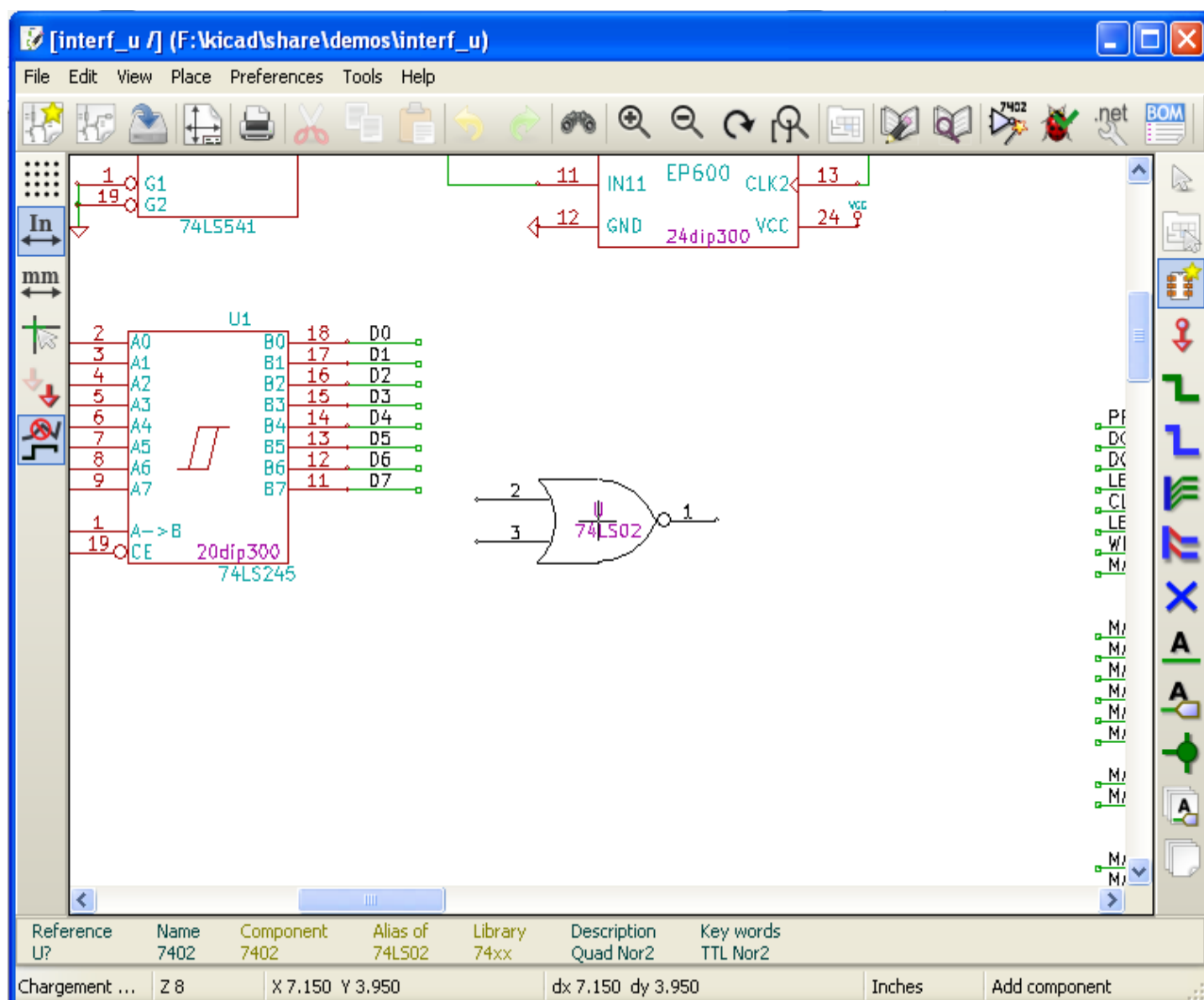
The selected component will appear on the screen, in placement mode.

Before placement in the desired position (with a left click), you can rotate the component 90 degrees by 90 degrees, have a mirror view according to axis X or Y, or select its representation via the fast edit pop-up menu.

This can also easily be done after placement.

If the desired component does not exist, remember that you can often load a similar component and modify it : if a 54LS00 is wanted, you can obviously load a 74LS00 and change the name 74LS00 to 54LS00.

Here is a component during placement:



5.4.2 - Power ports.

A power port symbol is a component (the symbols are grouped in the “**power**” library). So you can use the previous command.

But as these placements are frequent, the tool is available.

This tool is similar to the preceding one, except that the search is done directly in the “**power library**”, saving time.

5.4.3 - Component Editing / Modification (already placed component).

Editions / modifications are of two kinds.

- Modification of the component itself (position, orientation, part selection of a multi-part component).
- Modification of one of the fields (Reference, value, or others) of the component.

When a component has just been placed, you may have to modify its value (particularly for resistors, capacitors...), but it is useless to assign it a reference number immediately, or to select the part of a multi-part component (like a 7400). This can automatically be done by the annotation function.

5.4.3.1 - Component modification.

To do this position the cursor mouse on the component (not to position on a field). One can then:

- Double-click on the component to open the full editing dialog box.
- Right-click to open the Pop Up menu, and use one of the displayed commands (Move, Orientation, Edit, Delete).

5.4.3.2 - Text fields modification.

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

For simple editing:

- Double-click on the text field to modify it.
- Right-click and use one of the displayed commands (Move, Rotate, Edit, Delete) in the Pop Up menu.

For more complete editing, or in order to create fields, double-click on the component; this will open the “**component properties**” dialog box :

Component Properties

Options

Unit: 1

Orientation (Degrees):

☒ 0
☐ +90
☐ 180
☐ -90

Mirror:

☒ Normal
☐ Mirror ---
☐ Mirror |

Chip Name: CRYSTAL

☐ Convert

Fields

Référence	X1	
Valeur	8MHz	
Module	HC-18UH	
Documentation		
Info		
Fab.		
Ref. Fab		

Add Field
Delete Field
Move Up

Text Justification:

Horiz. Justify: ☐ Left ☒ Center ☐ Right

Vert. Justify: ☐ Bottom ☒ Center ☐ Top

Visibility

☒ Show ☐ Rotate

Style:

☒ Normal ☐ Italic ☐ Bold ☐ Bold Italic

Field Name: Référence

Field Value: X1

Size ("): 0.070

Pos X ("): 0.000

Pos Y ("): 0.200

OK Cancel

You can set the orientation and others options of the component, and edit, add or remove fields.

Each field can be visible or not, and displayed horizontally or vertically.

The displayed (and changeable) position is always indicated for a normally displayed component (no rotation or mirror) and relates to the anchoring point of the component.

The option “**Reset to Library Defaults**” set the component to the orientation 0, and the options, size and position of each field.

However, **texts fields are not modified** because this could break the schematic.

5.5 - Wires,Buses,Labels,Power ports.

5.5.1 - Basics.

All these drawing elements can also be placed with the tools on the vertical right toolbar.

These elements are:

- **Wires** for usual connections.
- **Buses** (which use is only to connect bus labels, for esthetic considerations of the drawing)
- **Dotted lines**, for graphic presentation.
- **Junctions**, to force connections between crossing wires or buses.
- **Bus entries** of Wire to Bus or Bus to Bus connections, for aesthetic considerations of the drawing.

Eeschema

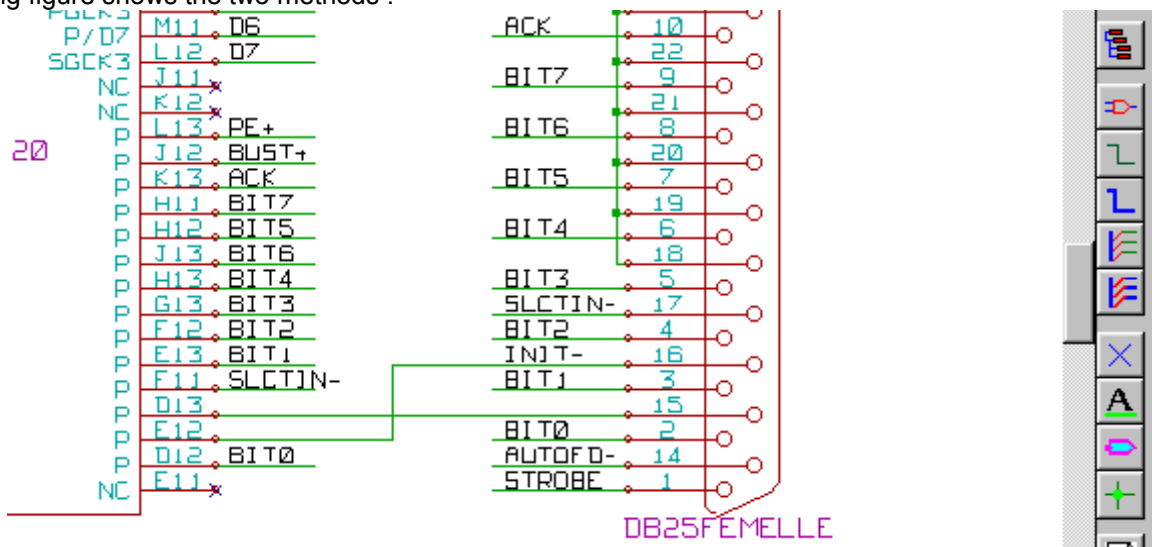
- **Labels** for usual connections.
- **Global labels**, for connections between sheets.
- **Texts** of comment.
- **"No Connection"** symbols.
- **Hierarchy sheets**, and their connection pins.

5.5.2 - Connections (Wires and Labels).

There are two ways to establish connection:

- Pin to pin wires.
- Labels.

The following figure shows the two methods :



Notice 1:

The point of "contact" (or anchoring) of a label is the lower left corner of the first letter of the label. This point must thus be in contact with a wire, or be superimposed at the point of contact of a pin so that this label is taken into account.

Notice 2:

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's end), **there is no connection**.

However, a label will be connected to a wire whatever the position of the anchoring point of the label on this wire.

Notice 3:

If a wire must be connected to another wire, otherwise than by their ends, it will be necessary to place a junction symbol at the crossing point.

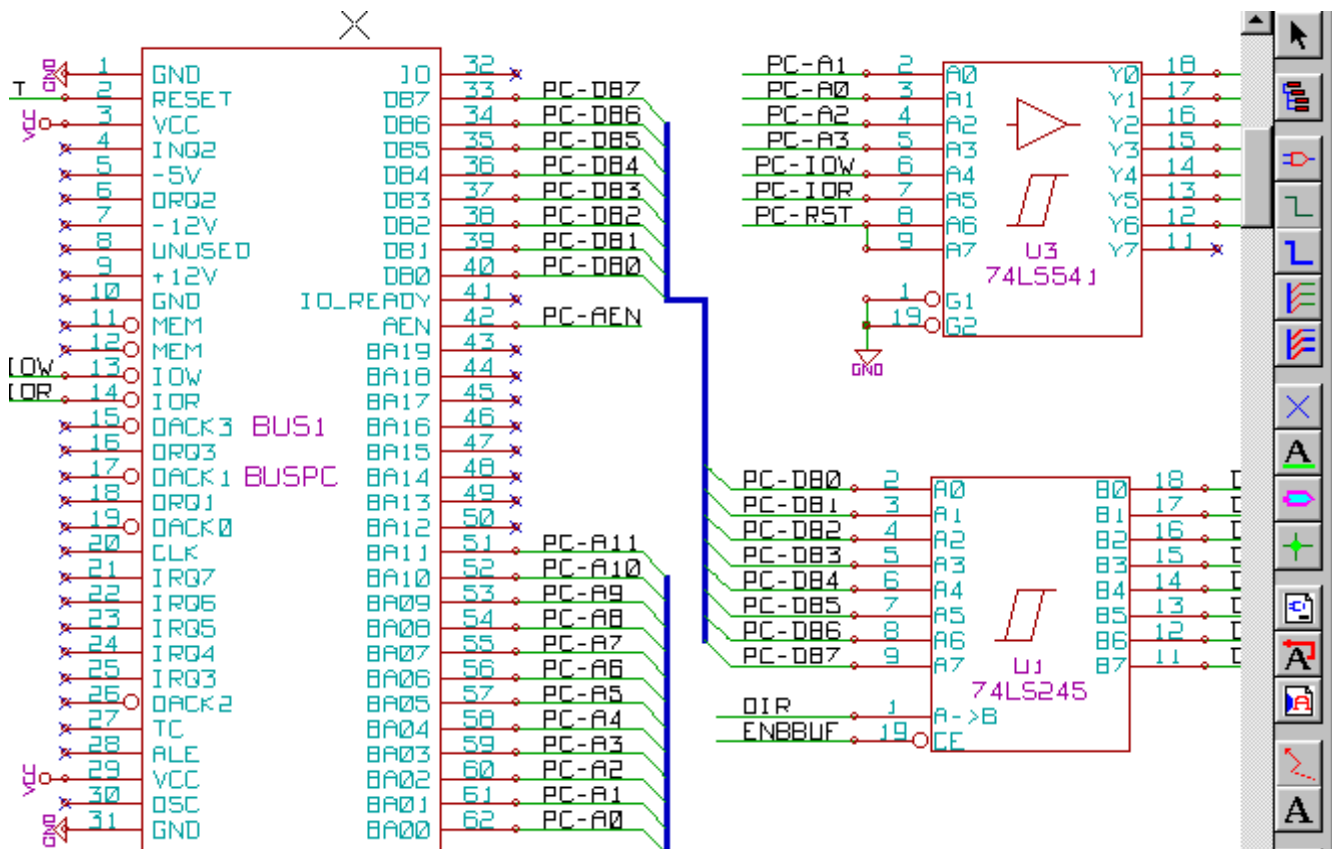
The previous figure (wires connected to DB25FEMALE pins 22, 21, 20, 19) shows such a case of connection using a junction symbol.

Notice 4:

If two different labels are placed on the same wire, they are connected together and become equivalent: all the other elements connected to one or the other label are then connected to all of them.

5.5.3 - Connections (Buses).

Let us consider the following schematic:



Many pins (particularly component U1 and BUS1) are connected to buses.

5.5.3.1 - Bus members.

From the schematic point of view, a bus is a collection of signals, starting with a common prefix, and ending by a number. This concept is not exactly the one which is used for a microprocessor bus. Each signal is a **member** of the bus. PCA0, PCA1, PCA2, are thus members of PCA bus.

The complete bus is named PCA [N..m], where N and m are the first and the last wire number of this bus.

Thus if PCA has 20 members from 0 to 19, the complete bus is noted PCA [0..19].

But a collection of signals like PCA0, PCA1, PCA2, WRITE, READ cannot be contained in a bus.

5.5.3.2 - Connections between bus members.

Pins connected between the same members of a bus must be connected by **labels**.

Indeed, directly connecting a pin to a bus is a non-sense, because a bus is a collection of signals, and this connection will remain ignored by Eeschema.

In the example above, connections are made by the labels placed on wires connected to the pins.

Connections via bus entries (wire segments at 45 degrees) to bus wires have only an esthetic value, and are not necessary on the purely schematic level.

In fact, due to the **repetition command** (**Insert** key), 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. EESchema 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 **Preferences/Options** menu, you can set the parameters of repetition:

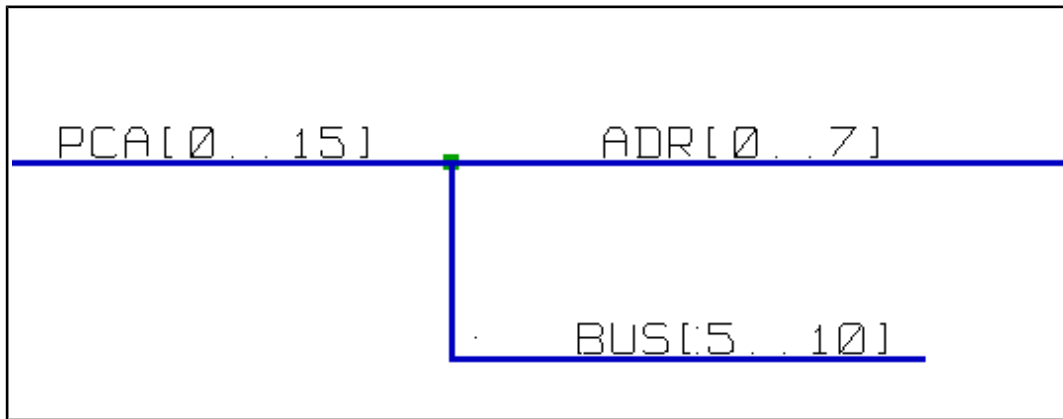
- Vertical step.
- Horizontal step.
- Label increment (which can thus be incremented by 2, 3. or decremented).

5.5.3.3 - Global Connections between buses.

You may need connections between buses, in order to link two buses having different names, or in the case of a

Eeschema

hierarchy, to create connections between different sheets.
You can make these connections in the following way.



Buses PCA [0..15], ADR [0..7] and BUS [5..10] are connected together (note the junction here because the vertical bus wire joins the middle of the horizontal bus segment).

More precisely, the corresponding members are connected together : PCA0, ADR0 are connected, (as same as PCA1 and ADR1... PCA7 and ADR7).

Furthermore, PCA5, BUS5 and ADR5 are connected (just as PCA6, BUS6 and ADR6 like PCA7, BUS7 and ADR7).

PCA8 and BUS8 are also connected (just as PCA9 and BUS9, PCA10 and BUS10)

On the other hand you cannot connect members of different weights in this way.

If you want to connect members of different weights from different buses, you will have to do that member by member like two usual labels, placing them on the same wire.

5.5.4 - Power ports connection.

When the power pins of the components are visible, they must be connected, as for any other signal.

The difficulty comes from components (such as gates and flip-flops) for which the power pins are normally invisible (**invisible power pins**).


The difficulty is double because :

- You cannot connect wires, because of their invisibility.
- You don't know their name.

And moreover, it would be a bad idea to make them visible and to connect them like the other pins, because the schematic would become unreadable and not in accordance with usual conventions.

Note:

If you want to enforce the display of these invisible power pins, you must check the option "**Show invisible power pins**" in

the Preferences/Options dialog box of the main menu, or the icon  of the left toolbar (options toolbar)

Eeschema connects automatically the invisible power pins :

All the invisible power pins of the same name are automatically connected between them without other notice.

However these automatic connections must be supplemented:

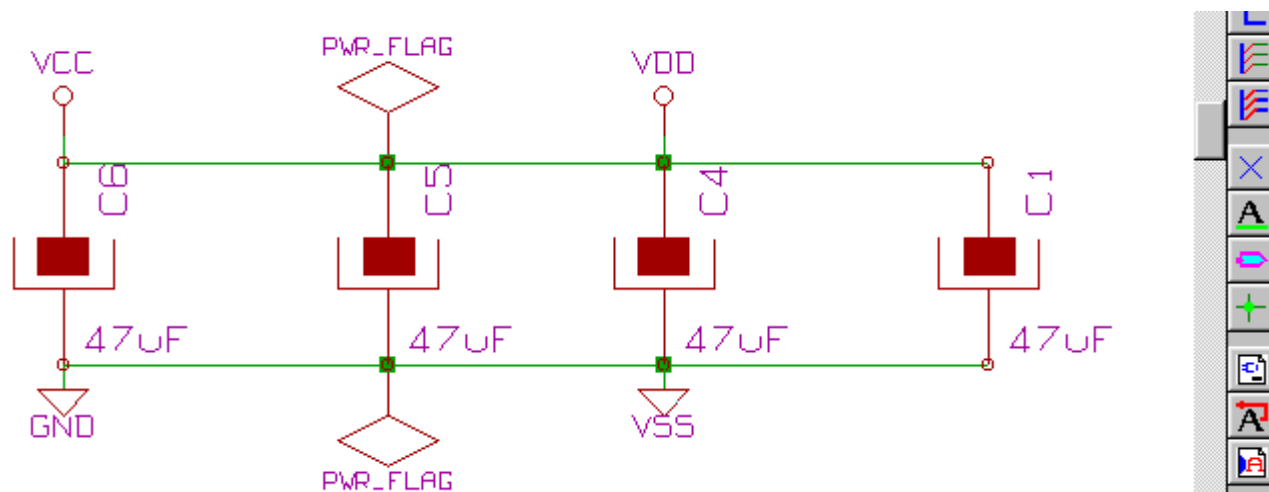
- By connections to the other visible pins, connected to this power port.
- Possibly by connections between groups of invisible pins of different names (for example, the ground pins are usually called "GND" in TTL components and "VSS" in MOS, and they must be connected together).

For these connections, you must use power ports symbols (components especially designed for this use, that you can create and modify with the library editor).

These symbols consist of an invisible power pin associated with the desired drawing.

Don't use labels, which have only a "local" connection ability, and which would not connect the invisible power pins. (See hierarchy concepts for more details).


The figure below shows an example of power ports connections.



In this example, ground (GND) is connected to power port VSS, and power port VCC is connected to VDD. Two PWR_FLAG symbols are visible. They indicate that the two power ports VCC and GND are really connected to a power source. Without these two flags, the ERC tool would diagnose : *Warning: power port not powered.* All these symbols are components of the schematic library "power".

5.5.5 - "No Connection" symbols.



These symbols are very useful for E.R.C. to avoid undesired warnings. (The electric rules check ensures that no connection has been inopportunately left unconnected).

If pins must really remain unconnected, it is necessary to place a NoConnection symbol (tool ) on these pins. These symbols however do not have any influence on the generated netlists.

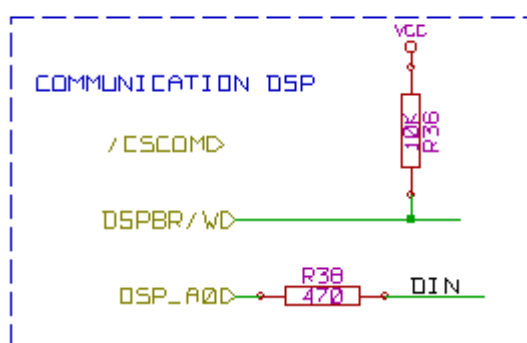
5.6 - Complements.

5.6.1 - Comments.

It can be usefull (for a good comprehension of the schematic) to place indications such as text fields, frames.

Text fields (tool ) and **dotted lines** (tool ) are intended for this use, contrary to labels and wires, which are connection elements.

Example of frame :



5.6.2 - Title block.

The title block is edited with the tool  :

Page Settings [X]

Page Size:

- ☐ Size A4
- ☒ Size A3
- ☐ Size A2
- ☐ Size A1
- ☐ Size A0
- ☐ Size A
- ☐ Size B
- ☐ Size C
- ☐ Size D
- ☐ Size E
- ☐ User size

Number of sheets: 1 Sheet number: 1

Revision: 2B ☐ Export to other sheets

Title: INTERFACE UNIVERSEL ☐ Export to other sheets

Company: KICAD suite GPL ☐ Export to other sheets

Comment1: Comment 1 ☐ Export to other sheets

Comment2: Comment 2 ☐ Export to other sheets

Comment3: Comment 3 ☐ Export to other sheets

Comment4: Comment 4 ☐ Export to other sheets

User Page Size X: 17,000

User Page Size Y: 11,000

OK Cancel

The complete title block is then:

Comment 4			E
Comment 3			
Comment 2			
Comment 1			
KICAD			
File: interf_u.sch			6
Sheet: /			
Title: INTERFACE UNIVERSEL			
Size: A3	Date: 4 jul 2009	Rev: 2B	
KiCad E.D.A. eeschema (2009-11-15-unstable)		Id: 1/1	
			7

The date and the sheet number (Sheet X/Y) are automatically updated:

- Date : when you modify the schematic.
- Sheet number (useful in hierarchy) : by the annotation function.

Headings:

[6 - Hierarchical schematics.](#)

[6.1 - Presentation.](#)

[6.2 - Navigation in the Hierarchy](#)

[6.3 - Local, hierarchical and global labels .](#)

[6.3.1 - properties:](#)

[6.3.2 - Notes:](#)

[6.4 - Hierarchy Creation. Headlines.](#)

[6.5 - Sheet symbol.](#)

[6.6 - Connections: Hierarchy pins.](#)

[6.7 - Connections: Hierarchical labels.](#)

[6.7.1 - Labels, Hierarchical Labels Global labels and Invisible Power Pins.](#)

[6.7.1.1 - Simple labels.](#)

[6.7.1.2 - Hierarchical labels.](#)

[6.7.1.3 - Invisible Power pins.](#)

[6.7.2 - Global Labels:](#)

[6.8 - Complex Hierarchy](#)

[6.9 - Flat Hierarchy](#)

6 - Hierarchical schematics.

6.1 - Presentation.

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 legibility.

From the root sheet, you must be able to find all sub-sheets.

Hierarchical schematics management is very easy with Eeschema, thanks to an integrated "hierarchy navigator" (button



of the upper and right toolbar, further detailed).

In fact, 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 components in the library that appear like traditional components in the schematic, but which actually correspond to a schematic which describes their internal structure.

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

Eeschema 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.**

Eeschema knows all these hierarchies.

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

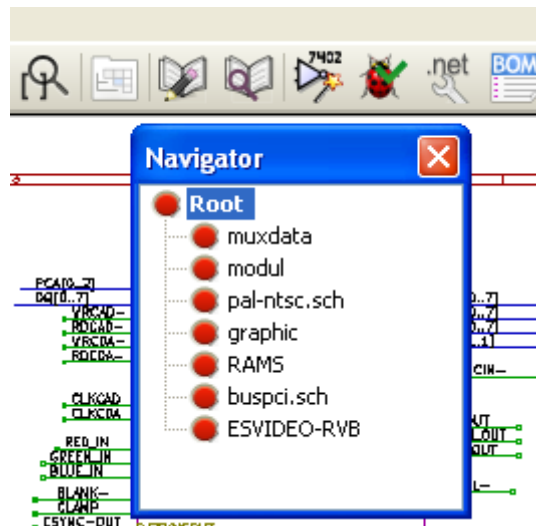
The two points to be known are :

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

6.2 - Navigation in the Hierarchy




It is very easy thanks to the navigator (tool of the **horizontal** toolbar) shown here :



Each sheet is reachable, clicking its name.

Fast navigation:

Right click on a sheet name, and choose **enter sheet**.


You can also quickly reach the root sheet, or a sub-sheet thanks to the tool  of the right **vertical** toolbar. After tool selection :

- Click on a sheet name to selection this sheet.
- Click elsewhere to select the main sheet.

6.3 - Local, hierarchical and global labels .

6.3.1 - properties:

Local labels (tool ) are connecting signals only within a sheet

hierarchical labels (tool ) are connecting signals only within a sheet **and** to a hierarchical pin placed in the parent sheet.

Global labels (tool ) are connecting signals across **all** the hierarchy.

Power pins (type **power in** and **power out**) **invisibles** are like global labels because they are seen as connected between them across all the hierarchy.

6.3.2 - Notes:

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

6.4 - Hierarchy Creation. Headlines.

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 component pins.

6.5 - 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 .

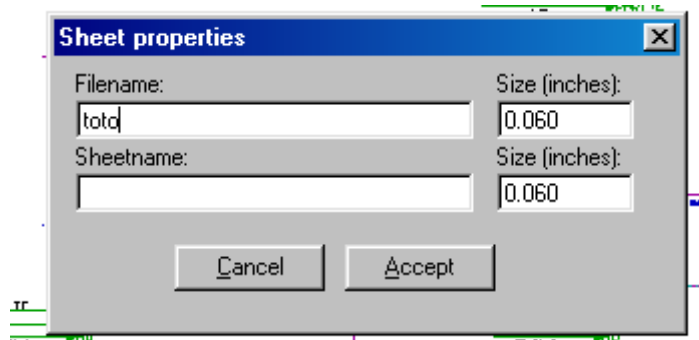
These labels are similar to usual component pins.

Select the tool .

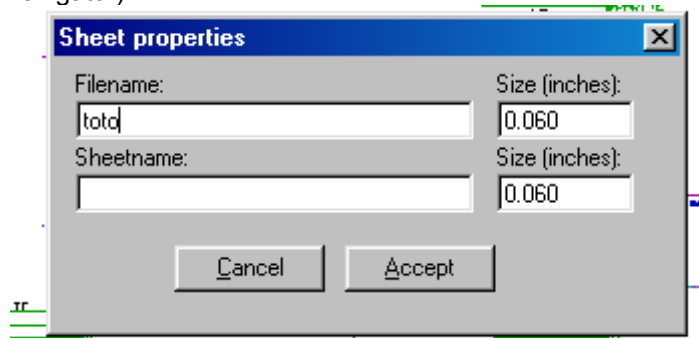
Click to place the upper left corner of the rectangle. Click again to place the lower right corner, having a large enough

Eeschema

rectangle.
Example :



You will then be prompted to type a filename 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 filename will be used as sheet name (usual way to do that).

6.6 - Connections: Hierarchy 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 component pins, with however the possibility to connect a complete bus with only one point of connection.

There are two ways to do this:

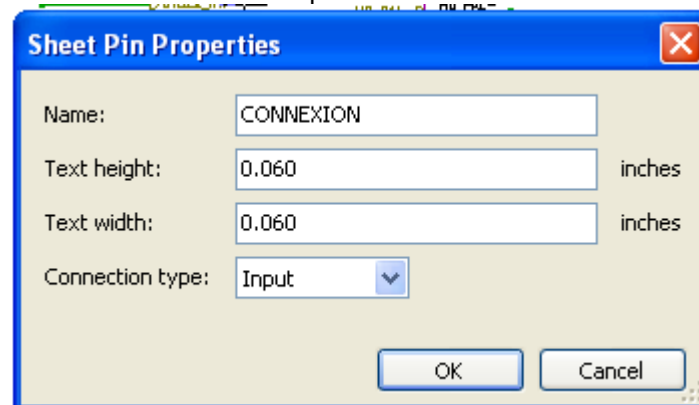
- Place the different pins **before** drawing the sub-sheet (manual placement).
- Place the different pins **after** drawing the sub-sheet, and the global labels (semi-automatic placement).

The second solution is quite preferable, as much as possible.

Manual placement:



- To select the tool
 - Click on the hierarchy symbol where you want to place this pin.
- See below an example of the creation of the hierarchical pin called "CONNEXION".



You can define its graphical attributes, and size or later, by editing this pin sheet (Right click and select Edit in the PopUp menu).

Various pin symbols are available :

- Input

Eeschema

- Output
- BiDir
- Tri State
- Not Specified

These pin symbols are only graphic enhancements, and have no other role.

Automatic placement:



- Select the tool
- Click on the hierarchy symbol from where you want to import the pins corresponding to global labels placed in the corresponding schematic. A hierarchical pin appears, if a new global 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 global labels.

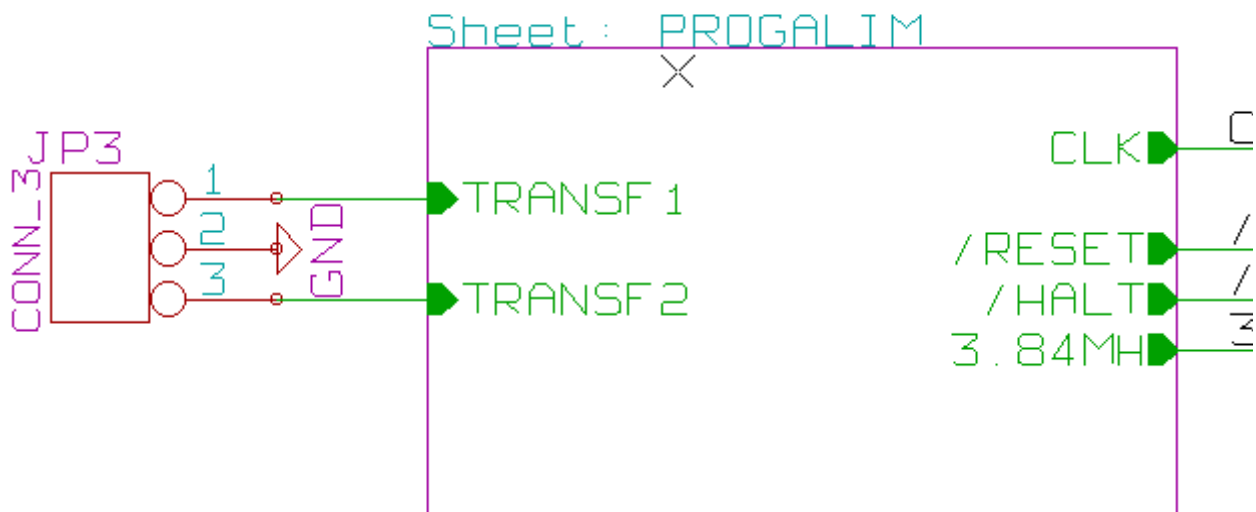
6.7 - Connections: Hierarchical labels.

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 HLabel) is similar.

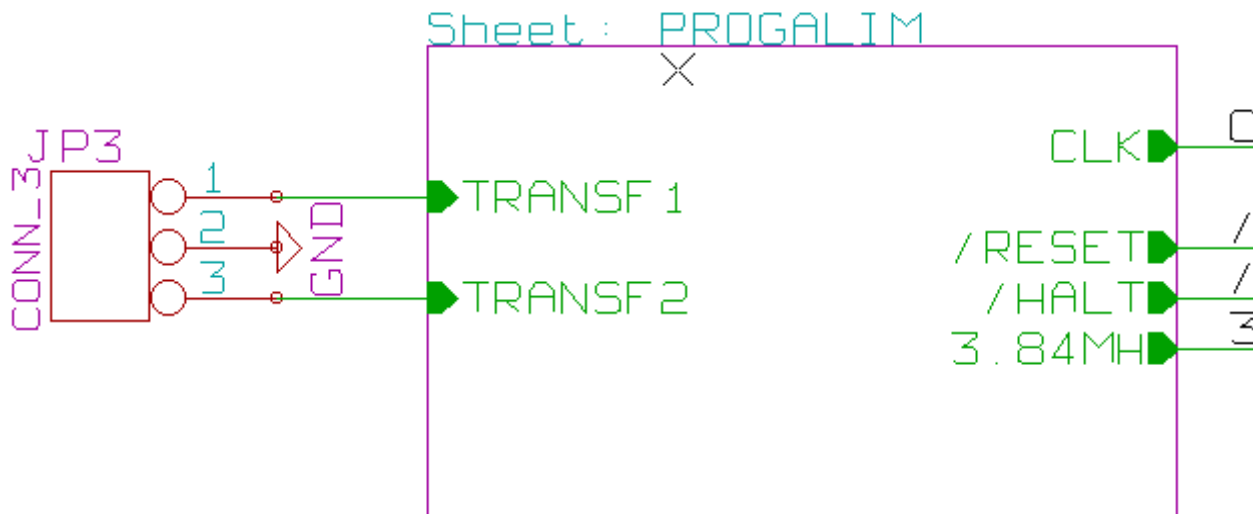


Hierarchical labels creation is made with the tool

See below a root sheet example :



Notice pins TRANSF1 and TRANSF2, connected to connector JP3.
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.

6.7.1 - Labels, Hierarchical Labels Global labels and Invisible Power Pins.

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

6.7.1.1 - Simple labels.

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

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

6.7.1.2 - Hierarchical labels.

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

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

However a HLabel is regarded as to be connected to the corresponding SheetLabel symbol in the hierarchical symbol placed in the root sheet.

6.7.1.3 - Invisible Power pins.

It was seen that they were connected together if they had the same name.

Thus all the power pins declared "Invisible Power Pins" and named VCC are connected and form the equipotential VCC, whatever the sheet they are placed on.

This explains 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 equipotential VCC, it will have to be explicitly connected to an invisible power pin, thanks to a VCC power port.

6.7.2 - Global Labels:

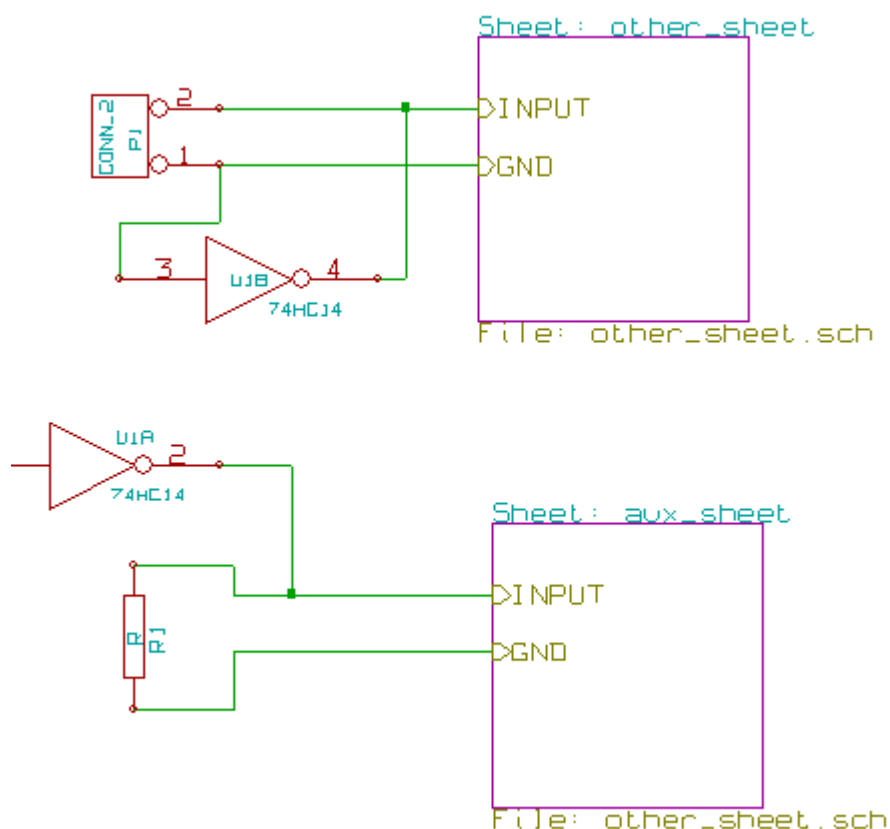
Global labels what have an identical name are connected across the whole hierarchy.
(powers like vcc ... are global labels)

6.8 - Complex Hierarchy

Here is an example : The same schematic is used twice (two instances).

Eeschema

The two sheets share the same schematic because the filename is the same for the two sheets ("other_sheet.sch"). But the sheet names must be different.

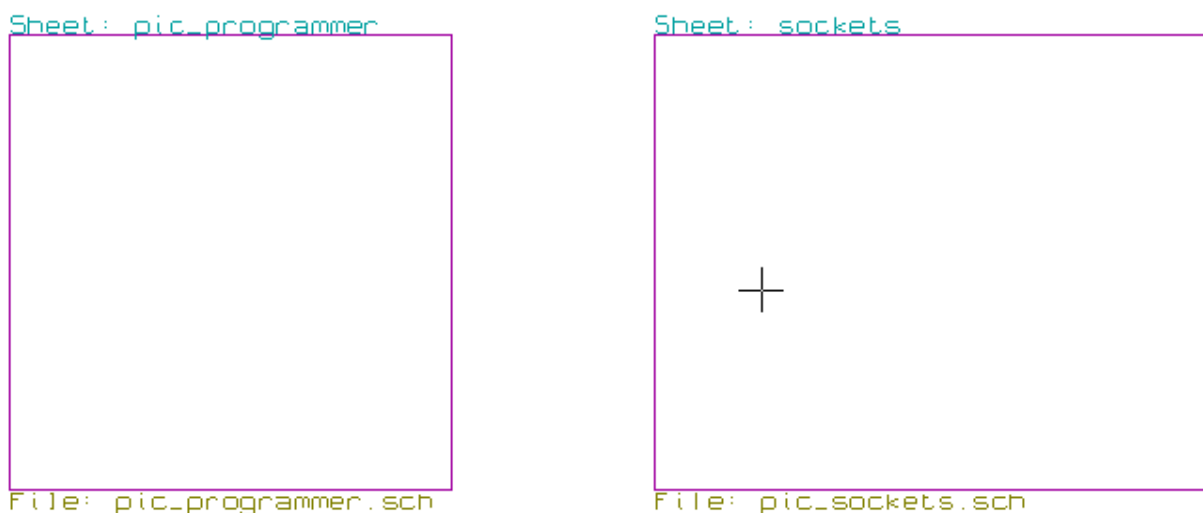


6.9 - Flat Hierarchy

One can create a project using many sheets, without creating connections between these sheets (flat hierarchy) if the next rules are used:

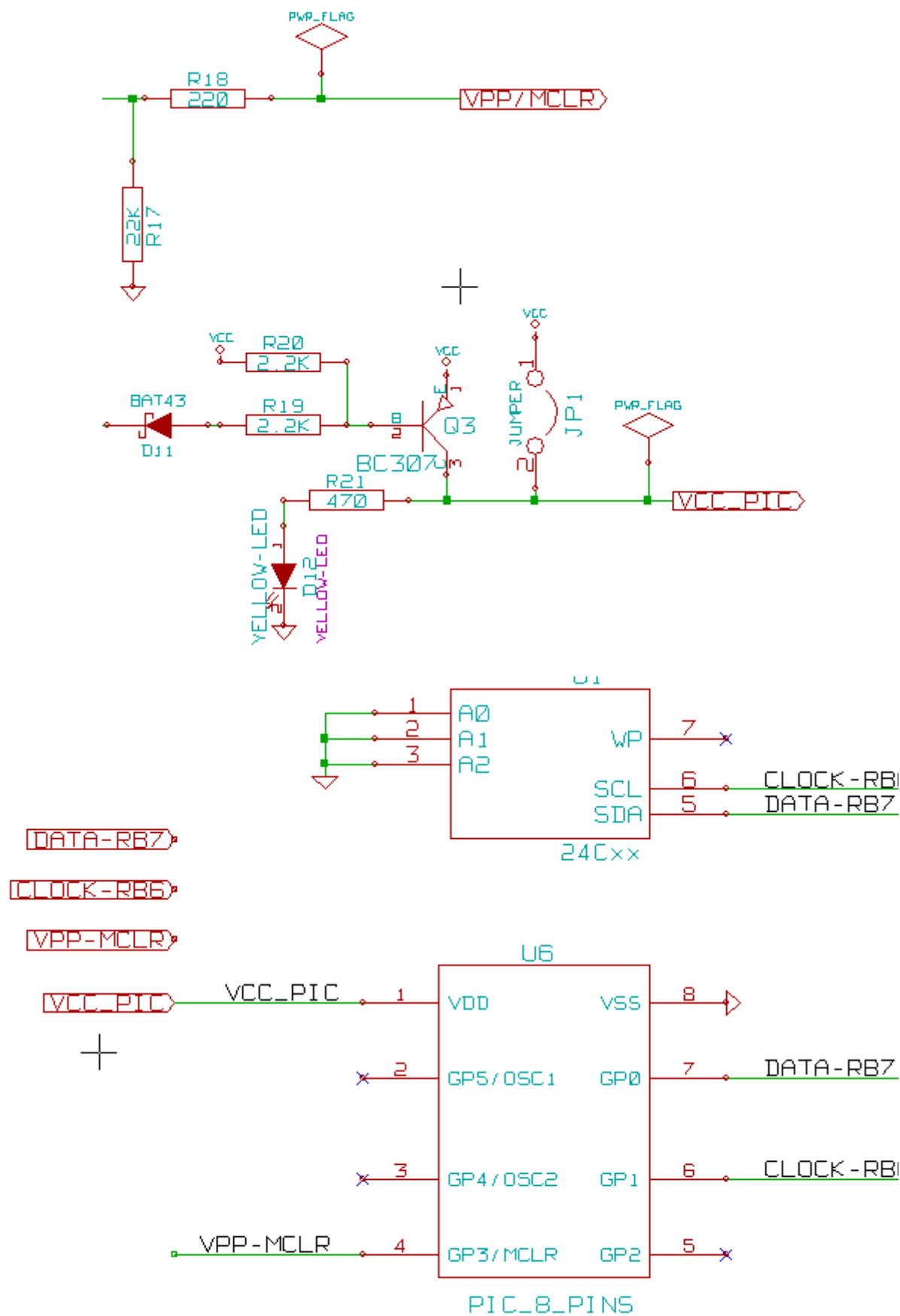
- One must create a root sheet containing the other sheets, which acts as a link between the others sheets.
- No explicit connections are needed.
- All connections between sheets will use Global Labels instead of hierarchical labels.

Here is the root sheet:



Eeschema

Here is the two pages, connected by global labels:



Eeschema

Look at global labels	<div>DATA-RB7</div> <div>CLOCK-RB6</div> <div>VPP-MCLR</div>
-----------------------	--

Headings:

[7 - Automatic classification Annotation.](#)

[7.1 - Purpose:](#)

[7.2 - Example.](#)


[7.2.1 - Annotation Order:](#)

[7.2.2 - Annotation Choice:](#)

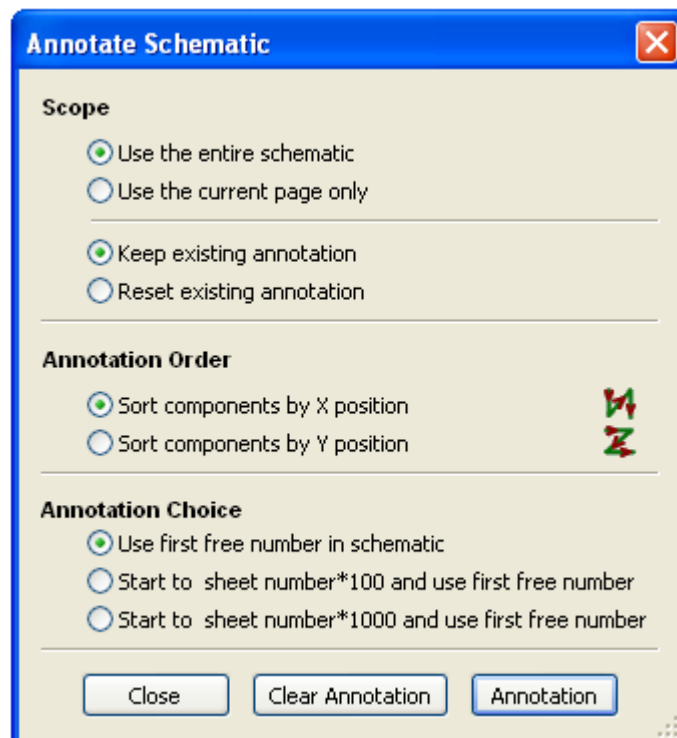
7 - Automatic classification Annotation.

7.1 - Purpose:



This (tool ) allows you to automatically assign a designator to the components, and for multiparts components, assign a multipart suffix to minimize the number of these packages.

The menu is:



Various possibilities are available :

- Annotate all the components (**Reset existing annotation** option)
- Annotate new components only (i.e. those whose reference finishes by? like IC?) (**Keep existing annotation** option).
- Annotate the whole hierarchy (**Use the entire schematic** option).
- Annotate the current sheet only (**Use current page only** option).

The **Annotation Order** choice gives the method used to attribute the reference number inside each sheet of the hierarchy.

Except in 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 Id:

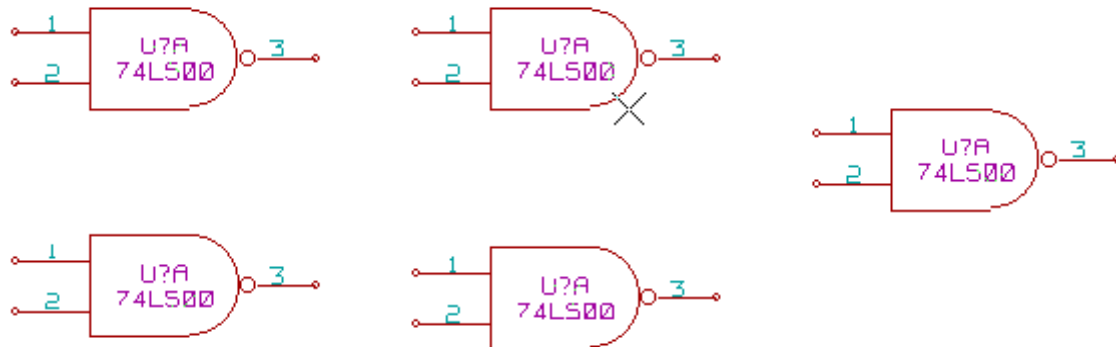
- **Use first free number in schematic:** components are annotated from 1 (for each reference prefix). If a previous annotation exists, not yet in use numbers will be used.
- **Start to sheet number*100 and use first free number:**
Annotation start from 101 for the sheet 1, from 201 for the sheet 2...
If there are more than de 99 items having the same reference prefix (U, R) inside the sheet 1, anntation uses numbers 200 and more, and anntoation for sheet 2 will start from the next free number.

Eeschema

- Start to sheet number*1000 and use first free number.
Annotation start from 1001 for the sheet 1, from 2001 for the sheet 2...

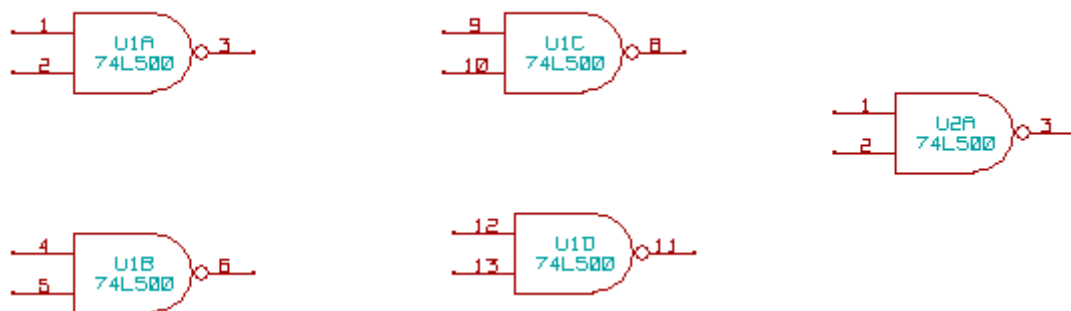
7.2 - Example.

7.2.1 - Annotation Order:

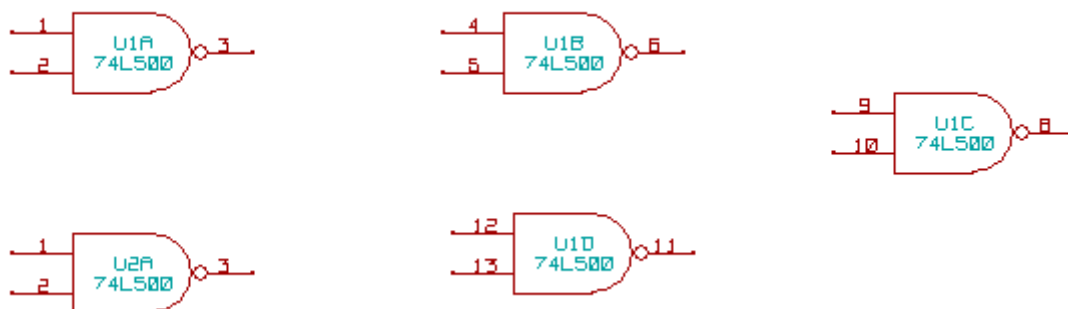


This example shows 5 elements placed, but not annotated.

After annotation :



Sort by X position



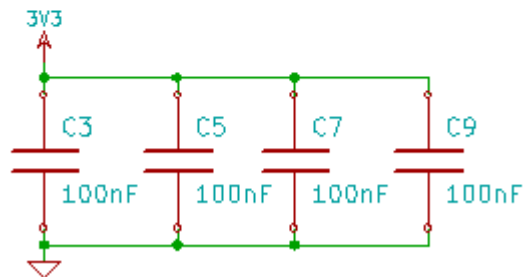
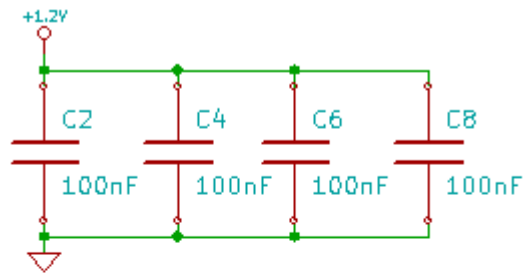
Sort by Y position

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

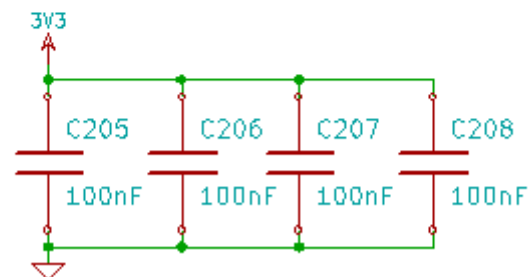
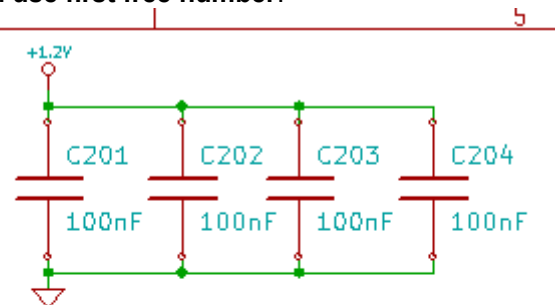
7.2.2 - Annotation Choice:

Here is an annotation in **sheet 2**:

Option **Use first free number in schematic**:

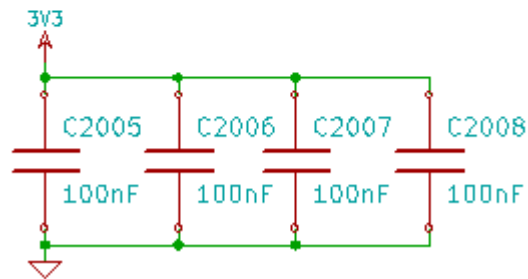
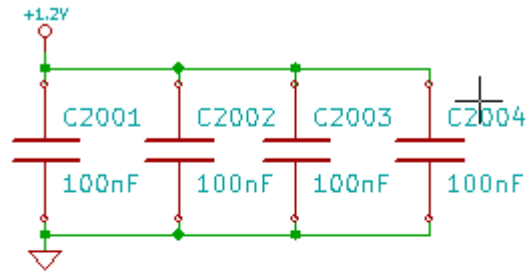


Option **Start to sheet number*100 and use first free number**:



Eeschema

Option **Start to sheet number*1000** and use first free number:



Headings

[8 - Design verification \(E.R.C.\)](#)

[8.1 - Overview.](#)

[8.2 - Use.](#)

[8.3 - Example of ERC :](#)

[8.4 - Displaying diagnostics:](#)

[8.5 - Powers and Power flags:](#)

[8.6 - Configuration](#)

[8.7 - ERC report file.](#)

8 - Design verification (E.R.C.)

8.1 - Overview.

The function “**Electrical Rules Check**” performs an automatic check.

It points out any errors in a sheet, such as unconnected pins, unconnected hierarchical symbols, shorted outputs...

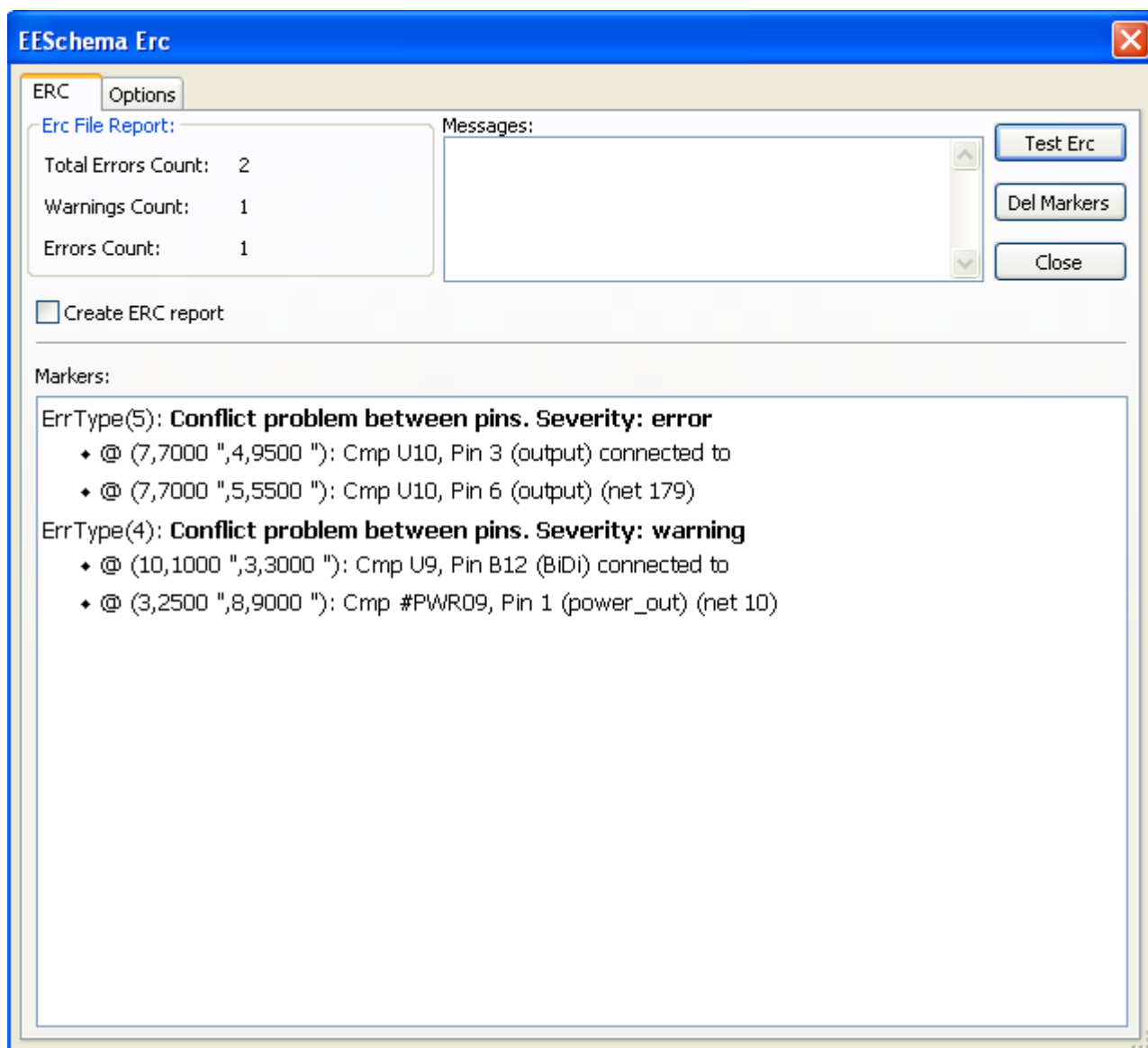
Naturally, an automatic check is not infallible, and the software that make it possible to detect all design errors was not written yet.

But, such a control is very useful, because it allows you to detect many oversights and small errors.

In fact all the detected errors must be checked and then corrected before proceeding as normal.

The quality of check is directly related to the care taken in declaring electrical pin properties during library creation.

ERC errors are reported as “errors” or “warnings”.



8.2 - Use.

The Control of E.R.C. is launched by the icon



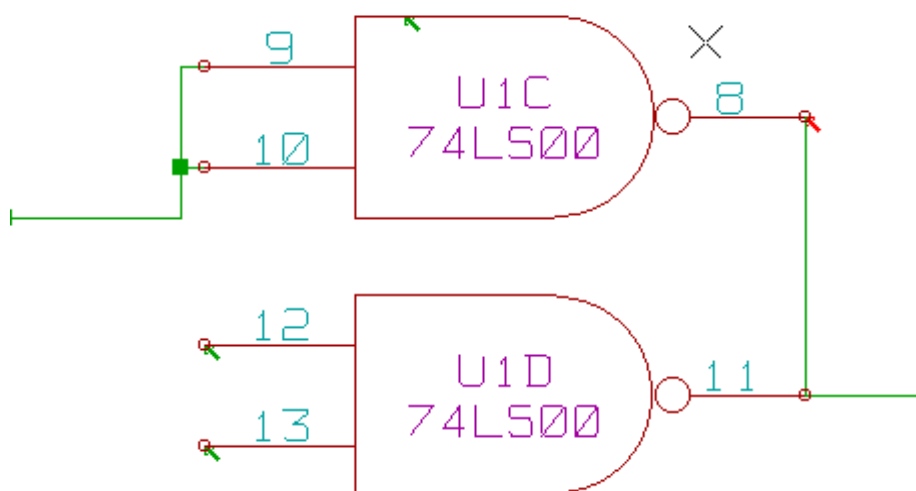
Markers are placed on the elements causing an ERC error (pins, or labels).

Notes:

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

You can also delete error markers from this dialog.

8.3 - Example of ERC :

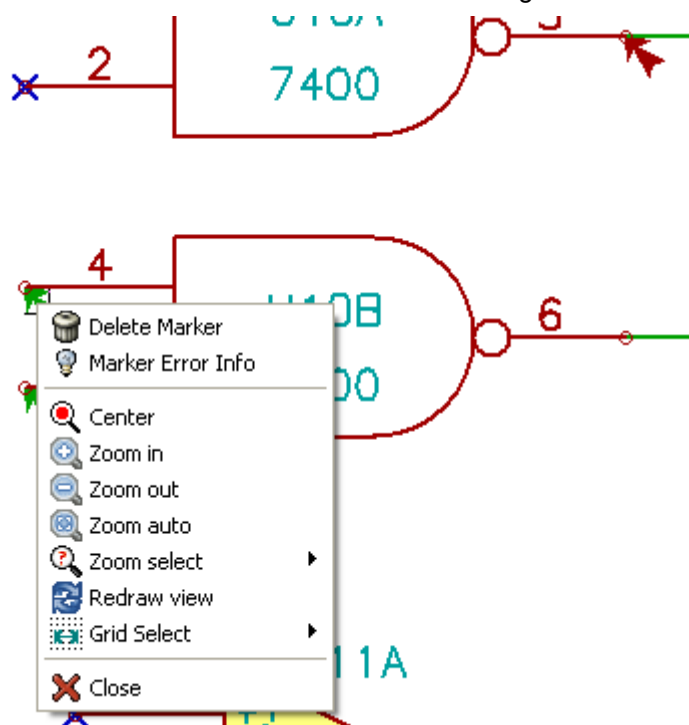


One can see 4 errors :

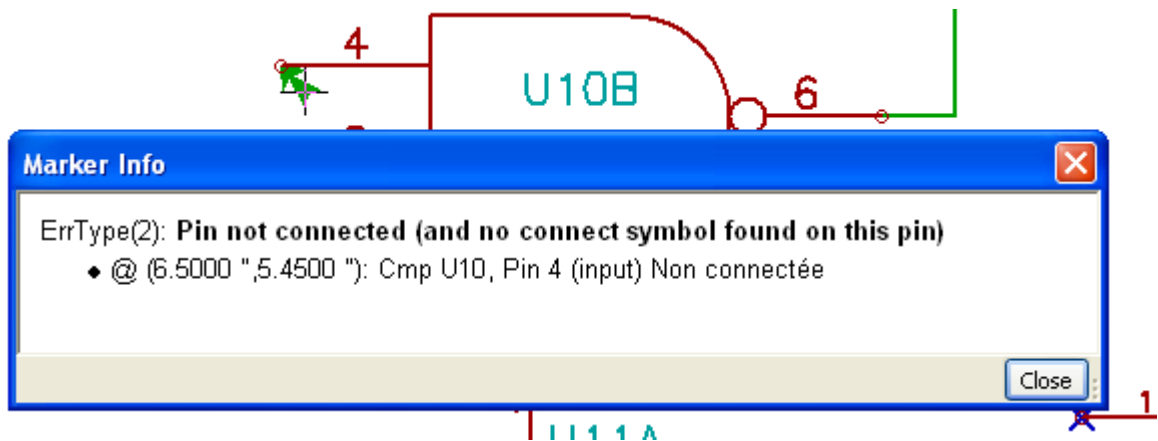
- Two outputs connected together.
- Two inputs left unconnected.
- An error on an **invisible** power port (power flag is missing).

8.4 - Displaying diagnostics:

On a right click on a marker, the pop menu allows to access the ERC marker diagnostic.



and when clicking on **Marker Error Info** :



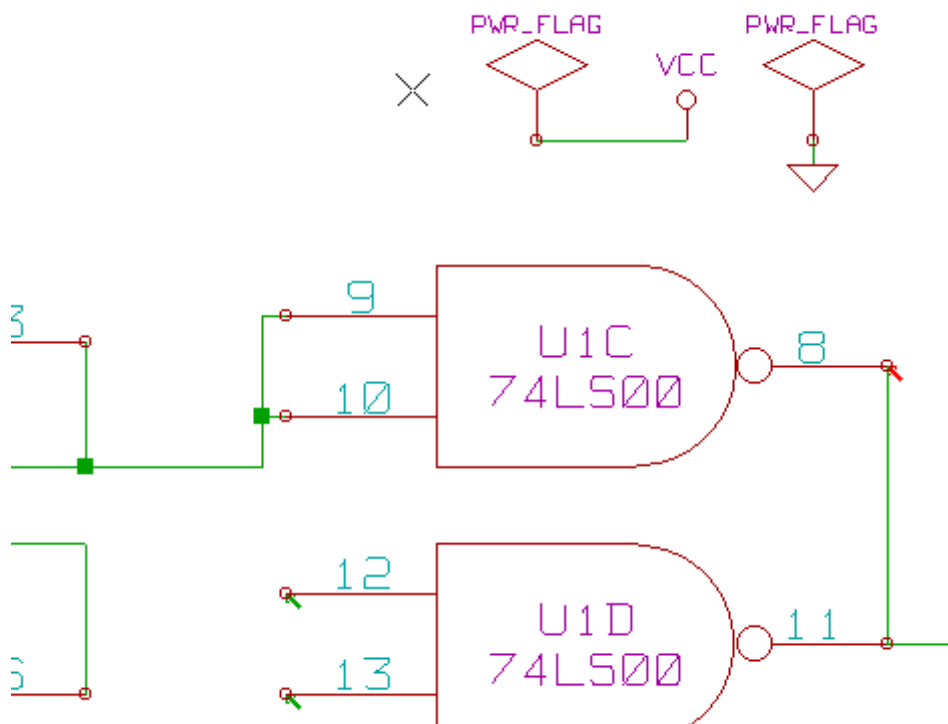
8.5 - Powers and Power flags:

It is common to have an error (warning) on power pins, whereas all seems normal (see example above).

This is because, in most designs, the power is provided by connectors, that aren't **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 (not driven by a power source).

You have to place a "**PWR_FLAG**" on such a power port (which symbolizes in fact power output).

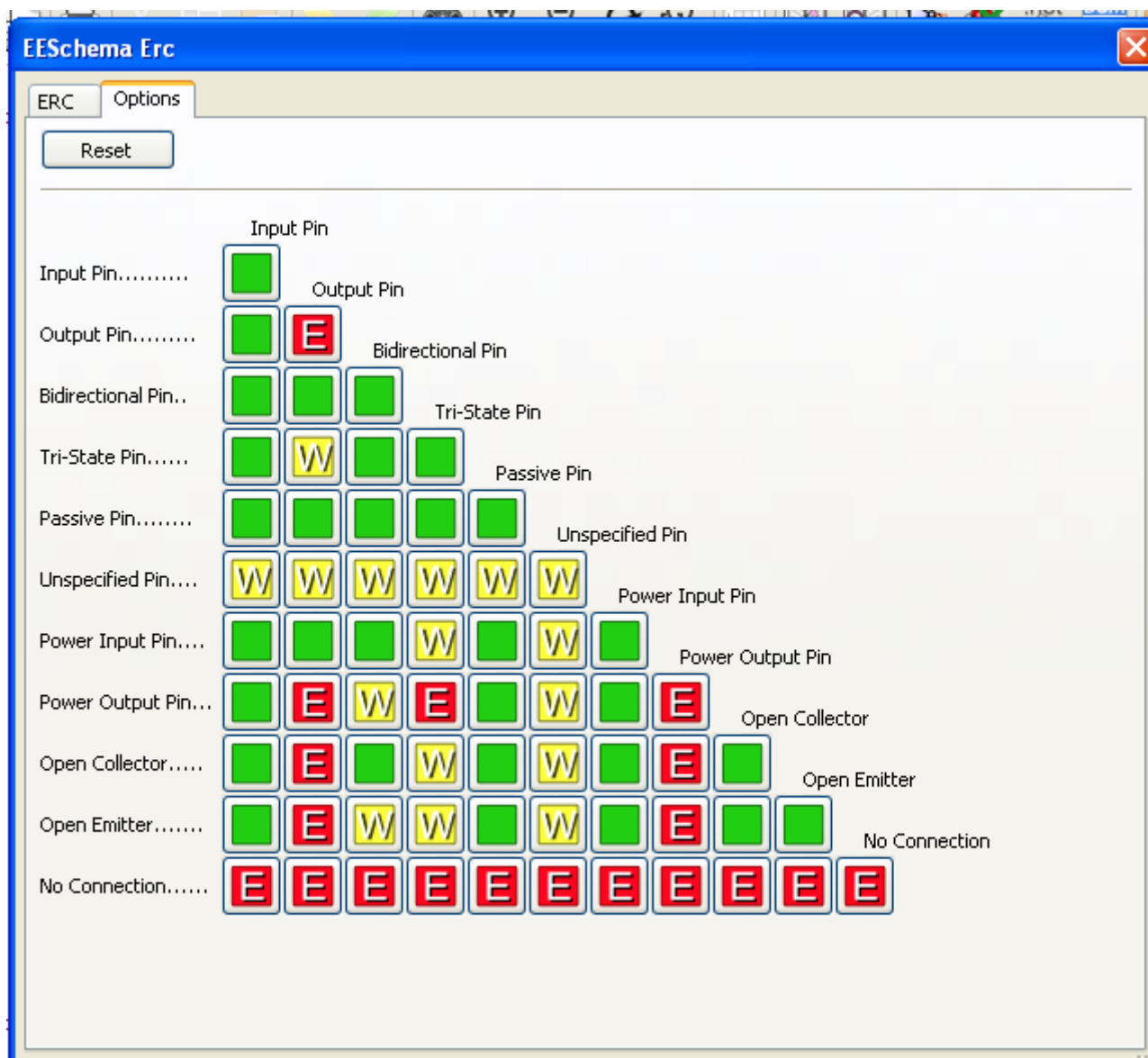


The error marker then disappears.

Most of time, a **PWR_FLAG** must be connected to gnd, because usually 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 pwr_flag.

8.6 - Configuration

The Options panel allows you to configure connectivity rules to define electrical conditions for errors and warnings :



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

8.7 - ERC report file.

An ERC report file can be saved by checking the option **Write ERC report**.

The file extension for ERC report files is **.erc**.

Here is an example:

```
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
```


Headings:

[9 - Netlist Creation.](#)

[9.1 - Overview.](#)

[9.2 - Netlist formats.](#)

[9.3 - Examples.](#)

[9.4 - Note.](#)

[9.4.1 - Precautions.](#)

[9.4.2 - PSPICE netlists.](#)

[9.5 - Other formats, using « plugins »](#)

[9.5.1 - Init the Dialog window:](#)

[9.5.2 - Command line format:](#)

[9.5.3 - The converter or the sheet style \(plug in\)](#)

[9.5.4 - Format of the intermediate netlist text file:](#)

9 - Netlist Creation.

9.1 - Overview.

This command allows you to create the netlist file for your design.

A netlist is a file which describes electrical connections between components.

One thus finds there

- The list of the components
- The list of connections between components, called equipotential nets.

There are different netlist formats. Sometimes the component list and equipotential list 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, like:


- PCB software.
- Simulators.
- CPLD (and other programmable IC's) compilers.

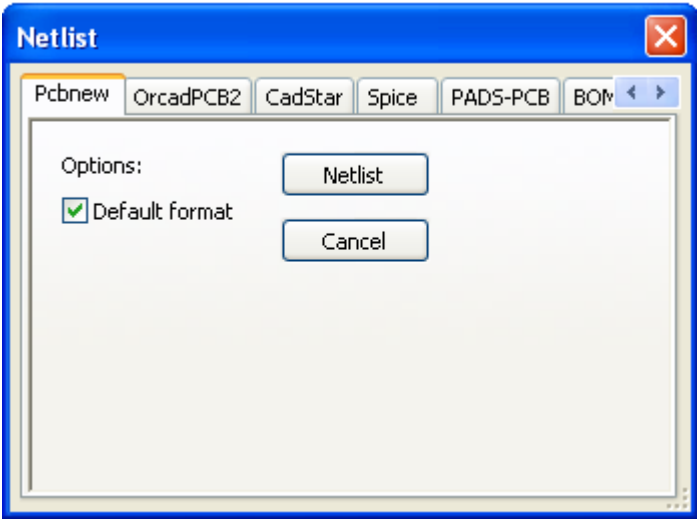
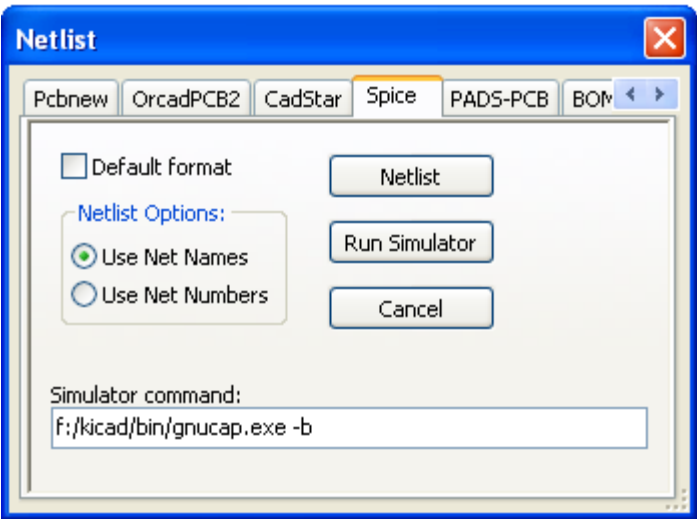
Eeschema supports several netlist formats :

- PCBNEW format (printed circuits).
- ORCAD PCB2 format (printed circuits).
- CADSTAR format (printed circuits).
- Spice format, for the simulators. (format Spice is also used by other simulators).

9.2 - Netlist formats.



Select the tool  to open the netlist creation dialog box :

	<p>Pcbnew selected</p>
	<p>Spice selected</p>

Using the different tabs, you can select the desired format as the default format.
In Spice format, you can generate netlists with either equipotential names (it is more legible) or net numbers (old Spice versions accept numbers only)

By clicking the **Netlist** button, you will be asked for a netlist file name.

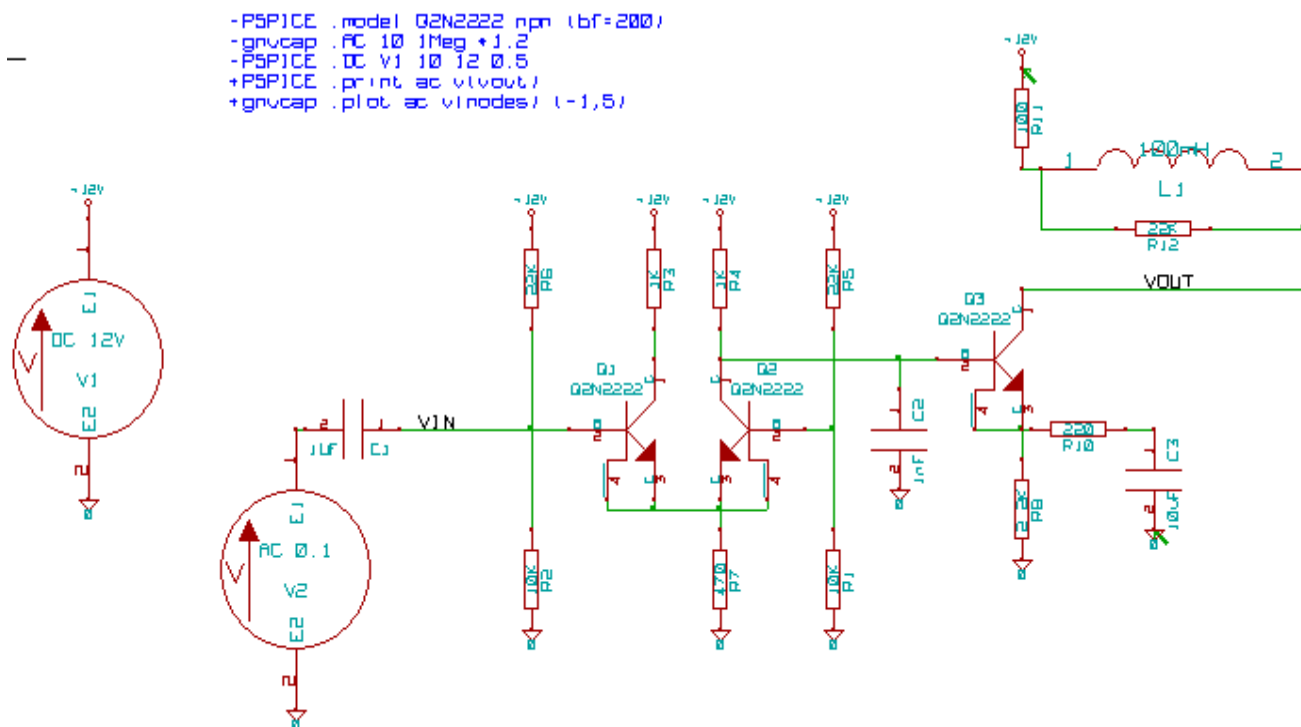
Note :

With big projects, the netlist generation can take a few minutes.

9.3 - Examples.

You can see below a schematic design using the PSPICE library :

Eeschema



Structure of a PCBNEW netlist :

```

# EESchema Netlist Version 1.0 genereé le 21/1/1997-16:51:15
(
(32E35B76 $noname C2 1NF {Lib=C}
(1 0)
(2 VOUT_1)
)
(32CFC454 $noname V2 AC_0.1 {Lib=VSOURCE}
(1 N-000003)
(2 0)
)
(32CFC413 $noname C1 1UF {Lib=C}
(1 INPUT_1)
(2 N-000003)
)
(32CFC337 $noname V1 DC_12V {Lib=VSOURCE}
(1 +12V)
(2 0)
)
(32CFC293 $noname R2 10K {Lib=R}
(1 INPUT_1)
(2 0)
)
(32CFC288 $noname R6 22K {Lib=R}
(1 +12V)
(2 INPUT_1)
)
(32CFC27F $noname R5 22K {Lib=R}
(1 +12V)
(2 N-000008)
)
(32CFC277 $noname R1 10K {Lib=R}
(1 N-000008)
(2 0)
)
(32CFC25A $noname R7 470 {Lib=R}

```

Eeschema

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

In PSPICE format, the netlist is as follows:

* EESchema Netlist Version 1.1 (Spice format) creation date: 18/6/2008-08:38:03

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

R12 /VOUT N-000003 22K
R11 +12V N-000003 100
L1 N-000003 /VOUT 100mH
R10 N-000005 N-000004 220
C3 N-000005 0 10uF
C2 N-000009 0 1nF
R8 N-000004 0 2.2K
Q3 /VOUT N-000009 N-000004 N-000004 Q2N2222
V2 N-000008 0 AC 0.1
C1 /VIN N-000008 1uF
V1 +12V 0 DC 12V
R2 /VIN 0 10K
R6 +12V /VIN 22K
R5 +12V N-000012 22K
R1 N-000012 0 10K
R7 N-000007 0 470
R4 +12V N-000009 1K
R3 +12V N-000010 1K
Q2 N-000009 N-000012 N-000007 N-000007 Q2N2222
Q1 N-000010 /VIN N-000007 N-000007 Q2N2222

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

.end
```

9.4 - Note.

9.4.1 - Precautions.

Many versions of software that exploit netlists do not accept spaces in the component names, pins, equipotentials or others.

Systematically avoid spaces in labels, or names and value fields of components or their pins.

In the same way, certain characters other than letters and numerals can induce problems.

Eeschema

Note that this limitation is not related to Eeschema, but to netlist formats that can then become untranslatable, or to software which uses these netlists.

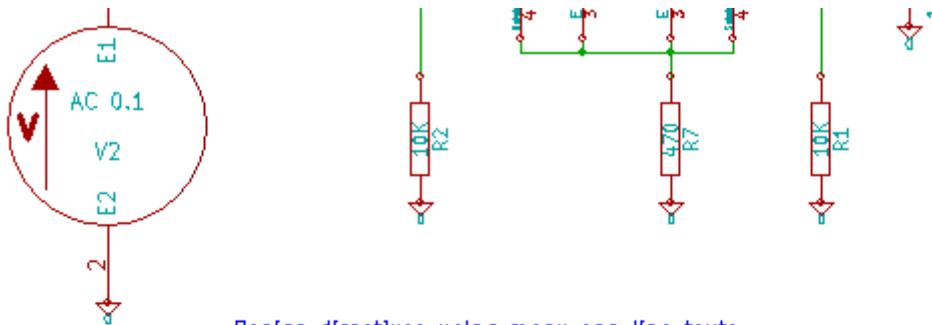
9.4.2 - PSPICE netlists.

For the Pspice simulator, you have to include some command lines in the netlist itself (.PROBE, .AC...).

Any text line included in the schematic diagram starting with the keyword **-pspice** or **-gnuicap** will be inserted (without the keyword) at the top of the netlist.

Any text line included in the schematic diagram starting with the keyword **+pspice** or **+gnuicap** will be inserted (without the keyword) at the end of the netlist.

Here is a sample, using many one line texts and one multiline text:



Pspice directives using many one line texts

```
-PSPICE .model Q2N2222 npn (bf=200)
-gnuicap .AC dec 10 1Meg *1.2
-PSPICE .DC V1 10 12 0.5
+PSPICE .print ac v(vout)
+gnuicap .plot ac v(nodes) (-1.5)
```

Pspice directives using one multiline text:

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

For example: if you type the following text (do not use a label!):

-PSPICE .PROBE

a line .PROBE will be inserted in the netlist.

In the previous example, three lines were inserted at the beginning of the netlist and two at the end with this technique.

If you are using multiline texts, **+pspice** or **+gnuicap** keywords are needed only once:

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

creates the four lines:

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

Besides, also note that the equipotential GND must be named 0 (zero) for Pspice.

9.5 - Other formats, using « plugins »

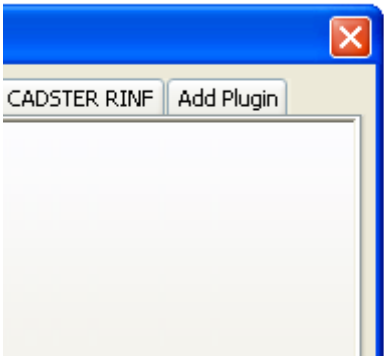
Eeschema

For other netlist formats one can add netlist converters.
These converters are automatically launched by Eeschema
Chapter 14 gives some explanations and examples of converters.

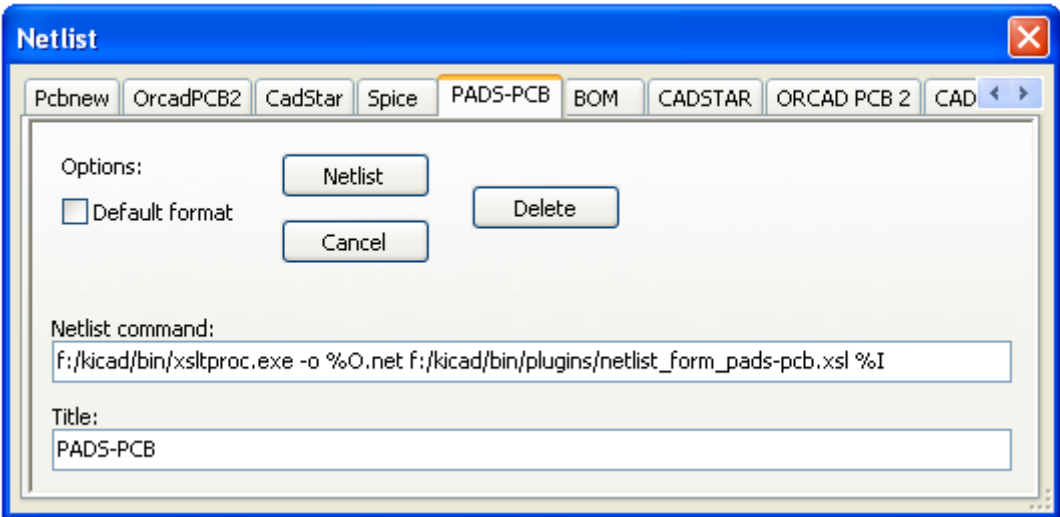
Principe:
A converter is a text file (xsl format) (but one can use other languages like PYTHON).
When using the xsl format, a tool (xsltproc.exe or xsltproc) read the intermediate file created by Eeschema, and the converter file to create the output file.
In this case, the converter file (a sheet style) is very small and very easy to write.

9.5.1 - Init the Dialog window:

One can add a new netlist plug-in by « Add Plugin »



Here is the plug-in « PadsPcb » setup:



- The setup needs::
- A title (for instance: the name of the netlist format)
 - The plug-in to launch.
- On activate the Netlist button: :
1. Eeschema creates an intermediate file *.tmp, for instance test.tmp
 2. Eeschema run the plug-in, what read test.tmp and creates test.net

9.5.2 - Command line format:

Here is an example, using « xsltproc.exe » as tool to convert .xsl files, and a file « /netlist_form_pads-pcb.xsl » as converter sheet style:

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

With:

f:/kicad/bin/xsltproc.exe	A tool to read and convert xsl file
-o %O.net	Output file: %O sera remplacé part le nom de la netliste (nom du schéma « racine »)
f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl	File name converter (a sheet style, xsl format).

%l	Will be replaced by the intermediate file created by Eeschema (*.tmp).
----	--

For a schematic named test.sch, the actual command line is:

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

9.5.3 - The converter or the sheet style (plug in)

This is a very simple software, because his purpose is only to convert an input text file (the intermediate text file) to an other text file

Moreover, from the intermediate text file, one also can create B.OM. lists.

When using xsltproc as converter tool, just a « sheet style » must be written.

9.5.4 - Format of the intermediate netlist text file:

See Chapter 14 for more explanations about xsltproc, the descriptions of intermediate file format, and some examples of sheet style for converters.

Headings:

[10 - Plot and Print](#)

[10.1 - Overview](#)

[10.2 - Common commands:](#)

[10.3 - Plot / Plot HPGL](#)

[10.3.1 - Sheet size selection](#)

[10.3.2 - Offset adjustments](#)

[10.4 - Plot / Plot Postscript](#)

[10.5 - Plot / Plot SVG](#)

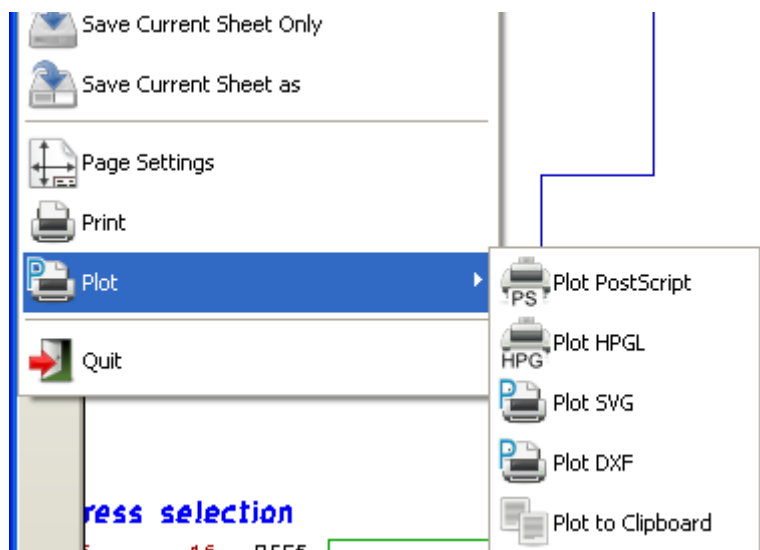
[10.6 - Plot / Plot DXF](#)

[10.7 - Print:](#)

10 - Plot and Print

10.1 - Overview

You can access both commands via the file menu:



The output formats are POSTSCRIPT, HPGL, SVG or DXF.
You can also print directly on your printer.

10.2 - Common commands:

Plot All allows you to plot the whole hierarchy (a file is generated for each sheet).

Plot CURRENT generates a file for the current sheet only.

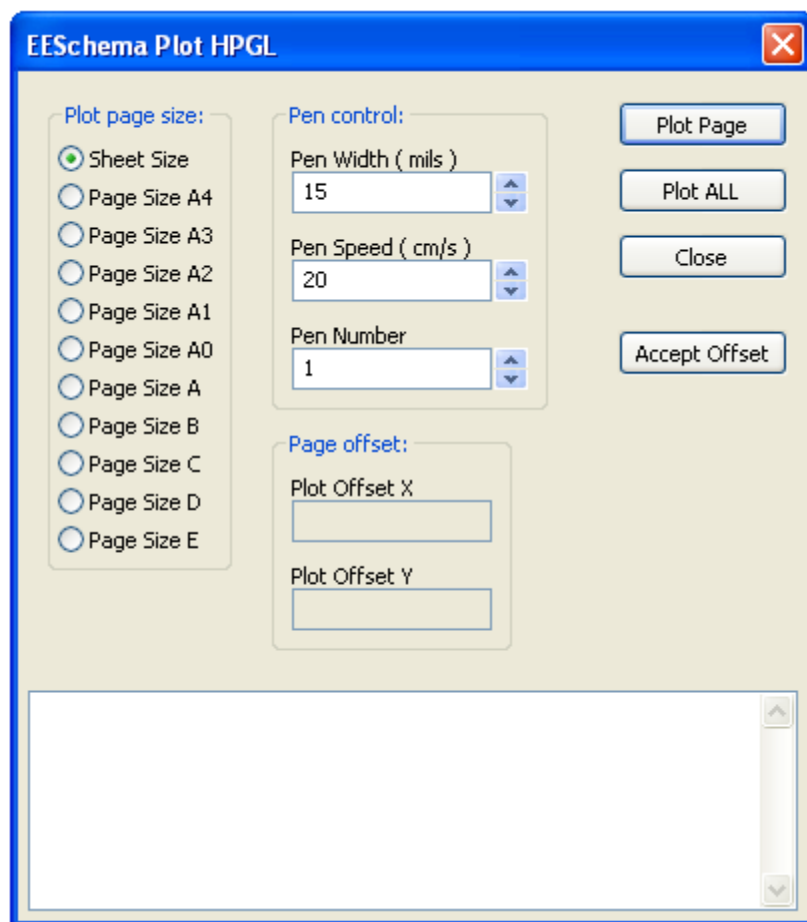
10.3 - Plot / Plot HPGL

This command allows you to create an **HPGL** file.

In this format you can define:

- Pen number
- Pen thickness (in 0,001 inch).
- Drawing speed (in cm/S).
- Sheet size.
- Print offsets.

Plotter setup dialog box :



The file name is the sheet name with the extension .plt.

10.3.1 - 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 scale is 1. If another sheet size is selected (A4 with A0, or A with E), the scale is automatically adjusted to fill the page.

10.3.2 - Offset adjustments

For all standard dimensions, one can adjust “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:

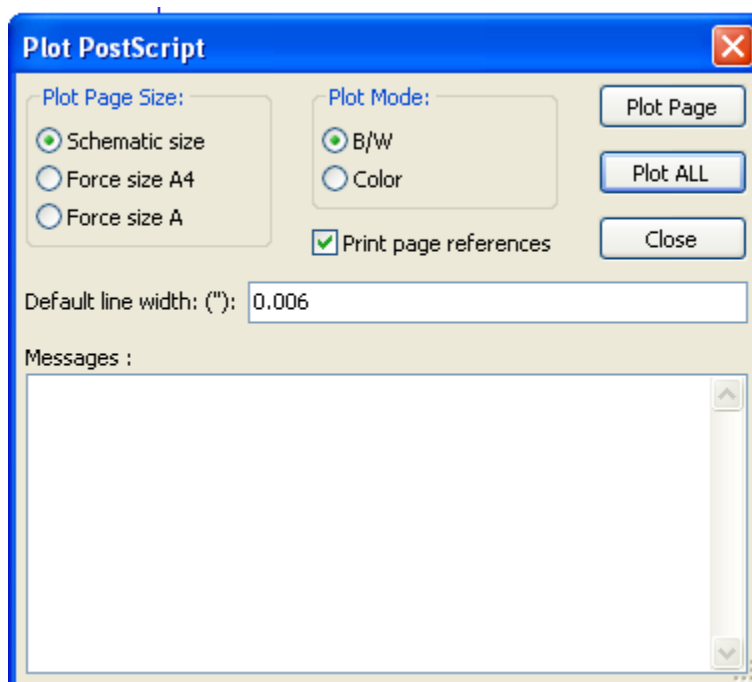
- For plotters having their origin point at the center of the sheet, the offsets must be negative and set at half of the sheet dimension.
- For plotters having their origin point at the lower left corner of the sheet, the offset must be set close to 0.

Set an offset:

- Select sheet size.
- Set Offset X and Offset Y.
- Click on **Accept Offset**.

10.4 - Plot / Plot Postscript

This command allows you to create **PostScript** files.

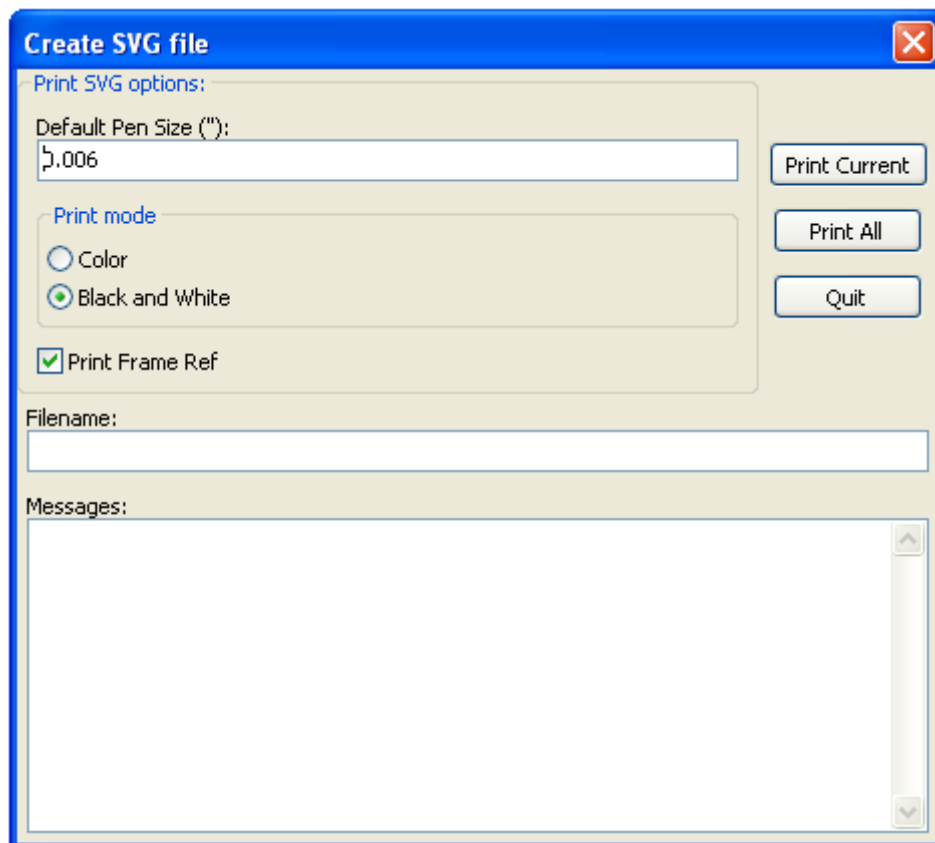


The file name is the sheet name with an extension .ps.

You can uncheck the option : Print title block. This is useful if you want to create a postscript file for encapsulation (format .eps) to insert a diagram in a word processing software.

The message window displays the file names created.

10.5 - Plot / Plot SVG

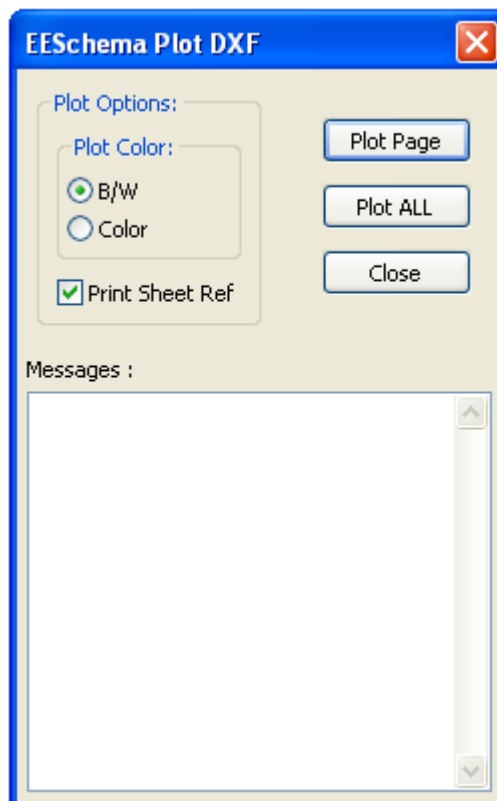


Allows you to create plot files using the vectored format SVG.

Eeschema

The file name is the sheet name with an extension .svg.


10.6 - Plot / Plot DXF

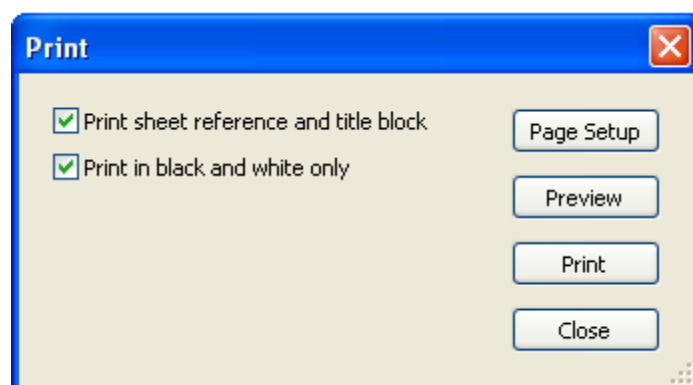


Allows you to create plot files using the format DXF.
The file name is the sheet name with an extension .dxf.

10.7 - Print:



This command, similar to the  toolbar icon command, allows you to visualize and generate design files on the standard printer.



Print sheet reference and title block option enable or disable sheet references and title block.

Print in black and white force printing in monochrome. This option is generally necessary if you uses a black and white laser printer, because colors are printed into not very readable half-tones.

Headings

[11 - LibEdit : Components Management](#)

[11.1 - General information about libraries](#)

[11.1.1 - Libraries :](#)

[11.1.2 - Management Menus](#)

[11.2 - Components overview](#)

[11.3 - Load a component for editing](#)

[11.3.1 - Main Toolbar](#)

[11.3.2 - Library selection and maintenance](#)

[11.3.3 - Component selection and saving](#)

[11.3.3.1 - Selection](#)

[11.3.3.2 - Save](#)

[11.3.3.3 - Transfer into another library](#)

[11.3.3.4 - Cancellation of component editing](#)

[11.4 - Component creation](#)

[11.4.1 - Create of a new component](#)

[11.4.2 - Creation based on another component](#)

[11.4.3 - Editing of the main characteristics](#)

[11.4.4 - Multi-part components](#)

[11.5 - Component design](#)

[11.5.1 - Graphic elements membership options](#)

[11.5.2 - Geometrical graphic elements](#)

[11.5.3 - Graphic elements of text type](#)

[11.6 - Pin creation and editing](#)

[11.6.1 - Pins overview](#)

[11.6.2 - Multi-part components, double representation.](#)

[11.6.3 - Pins: basic option](#)

[11.6.4 - Pins: Defining characteristics](#)

[11.6.5 - Pins shapes](#)

[11.6.6 - Pins : Electric types](#)

[11.6.7 - Pins : global modifications](#)

[11.6.8 - Pins : Multi-part components and double representations](#)

[11.7 - Field Editing](#)

[11.8 - Power port symbols : Creation](#)

11 - LibEdit : Components Management

11.1 - General information about libraries

11.1.1 - Libraries :

All the components used in a schematic are described in the component libraries.

In order to be able to have a reasonably simple management of these components, several libraries are used, grouping components by topic (by functions, or manufacturers.).

The Library management menu allows you to create libraries, add, delete or transfer components.

It also naturally allows you to quickly visualize the components of a library.

11.1.2 - Management Menus

There are two library management menus :

- **ViewLib** which allows you only to visualize the components, but gives quick access to the components.



Click on

- **LibEdit** which really allows you to manage the components and libraries.



Click on

11.2 - Components overview

A component in a library is composed of :

- Its graphic design (lines, circles, text fields).
- Pins which, (which must respect the usual graphic standards (regular pin, or clock pin, or inverted, or low level active...)) describe electrical properties, used by E.R.C. function.
- Fields (text) such as reference, value, corresponding module name for PCB design...

It can also have an alias, i.e. several names (thus one 7400 can also have several alias like 74LS00, 74HC00, 7437, because all these components are identical to the schematic diagram design).

The use of aliases is a very interesting method of creating complete but compact and (relatively) quickly built libraries.

Designing a component is:

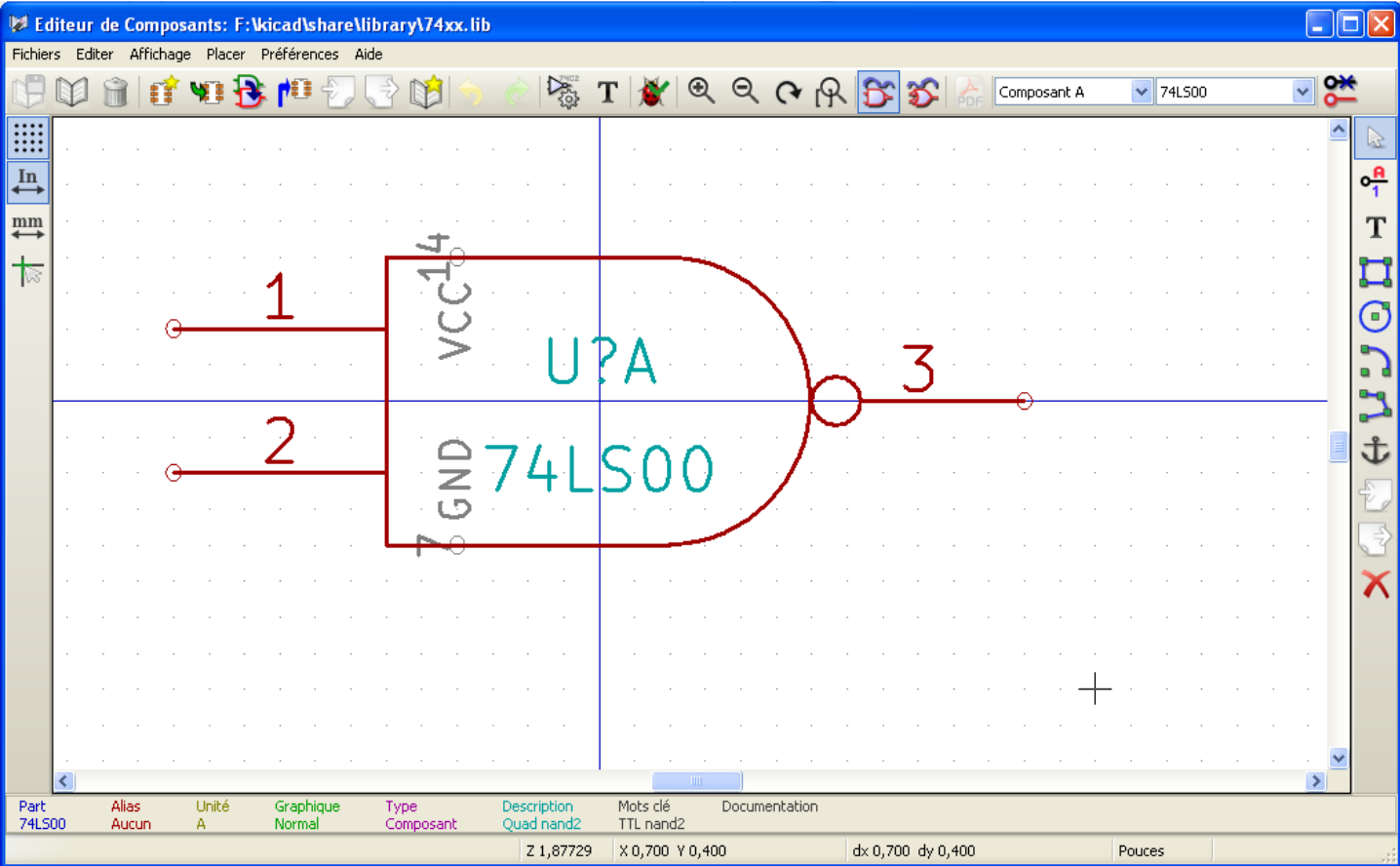
- Defining general properties: does it have multi-parts, and how many, does it have a double representation (known as Morgan, and in Eeschema normal and converted representation).
- Designing it (pins excepted) using lines, rectangles, circles, polygons and texts.
- Adding pins, carefully defining its graphic design, name and the number of pins, and their electrical properties (input, output, 3 states, power port...).
- Adding an alias if other components have the same design and pinout (or removing one if the component has been created by copying of another component).
- Adding fields if desired (it is optional, the name of the module is used by the PCB design software) and/or defining their visibility.
- Documenting the component.
- Saving it in the desired library.

11.3 - Load a component for editing











Click on the tool to open **Libedit**, the component editing and library management window .

Libedit looks as shown below :









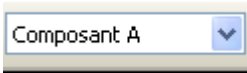
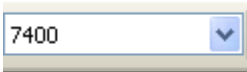



11.3.1 - Main Toolbar




	Save current library on hard disc
	Current library selection
	Delete a component in current library
	Create a new component
	Load component from current library for editing
	Create a new component from the current loaded component.
	Save the current component in current library only in RAM. The library file on disk is not changed.
	Import one component.

Eeschema

	Export the current component.
	Create a new library file with the current component.
	Undo / Redo command
	Edit the component properties.
	Edit fields of component: reference, value/name in lib and/or others fields
	Show the representation: Normal or converted (De Morgan)
	Show the associated documentation (if exists)
	Part selection (for multi part components)
	Alias selection (if the current components has aliases)
	Pin editing: independant editing for pin shape and position (for multi parts and De Morgan representation)
	

11.3.2 - Library selection and maintenance



The selection of the current library is possible by the icon , which displays for selection the list of the available libraries.

When a component is loaded or saved, it will remain it in this library.
The library name of a component is also its field « Value ».


Note:

You have to load a library in Eeschema, in order to use it.



The current library can be saved after modification, by clicking .



A component can be removed from the library by clicking .

11.3.3 - Component selection and saving

When a component is edited, you don't really work on the component in library, but on its copy in RAM.

Thus you can easily cancel any editing.

A component may come from a library, or from an old component.

Once loaded, it will be displayed.

11.3.3.1 - Selection



The icon displays the list of the available components, to select and load the desired component.

Notice 1:

If a component **alias** is selected, the **main component** will be loaded (Eeschema always displays the name of the component really loaded).

- The list of component aliases is always loaded with each component, and can thus be edited.
- When you want to edit one **alias**, this alias must be selected in the toolbar window :



The first item of the list is the root component.

Notice 2:



Alternatively, the **Import** command () allows you to load a component which has been



previously saved by the **Export** command ().

11.3.3.2 - Save

After modification, a component can be saved in the current library, or in a new library, or exported in a backup file.



To save in current library, use the **Update** command ().

However the update is done only in RAM (By this way, you can make up your mind in the schematic diagram.



If one wants to completely save the component, you have to use the save tool which will modify the library file on hard disc.



If you want to create a new library for this component, use the **NewLib** command (). You will be asked for a new library name.

Note:

If you want to be able to find it, don't forget to add it to the list of libraries to be searched by Eeschema (see Eeschema configuration).






Finally, one can use the **Export** command () to create a file containing only this component (this file is a standard library file, which contains only one component).

In fact the **NewLib** and **Export** commands are identical, the first proposes by default to create a library in the default library directory, and the second in the user directory.

11.3.3.3 - Transfer into another library

One can very easily copy a component from a source library into a destination library using the following commands:

- Select the source library as current library 
- Load the component to be transferred 
- Select the destination library as the current library 

- Save the component in RAM .


- Save the modified library .

11.3.3.4 - Cancellation of component editing

The edited component is only a working copy of the component really in the library. As long as it hasn't been saved to RAM, you just have to reload it (or to reload another library) to cancel the changes made to this component. If you have already saved it in RAM, and you haven't saved the library file to hard disc, one can quit and start Eeschema again, and then read from the library again.

11.4 - Component creation

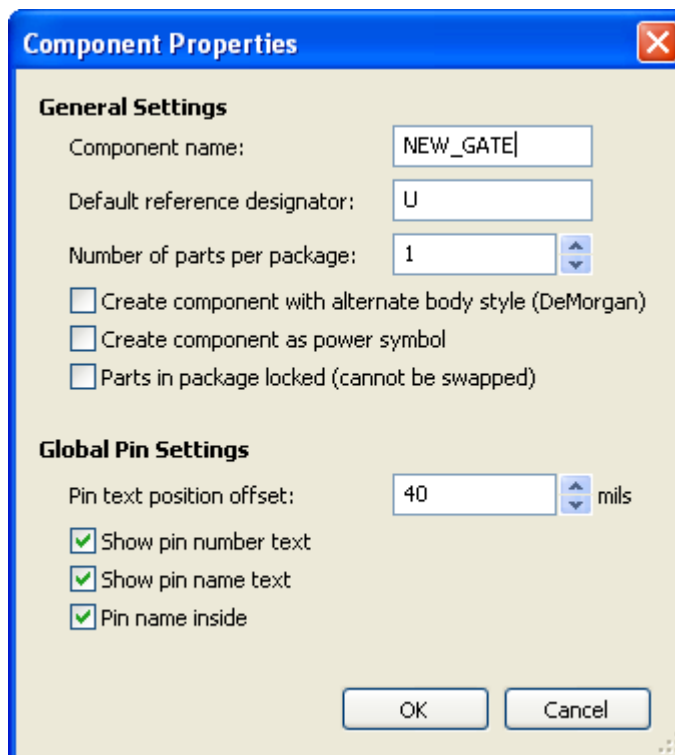
11.4.1 - Create of a new component

A new component can be created with the **NewPart** command (.

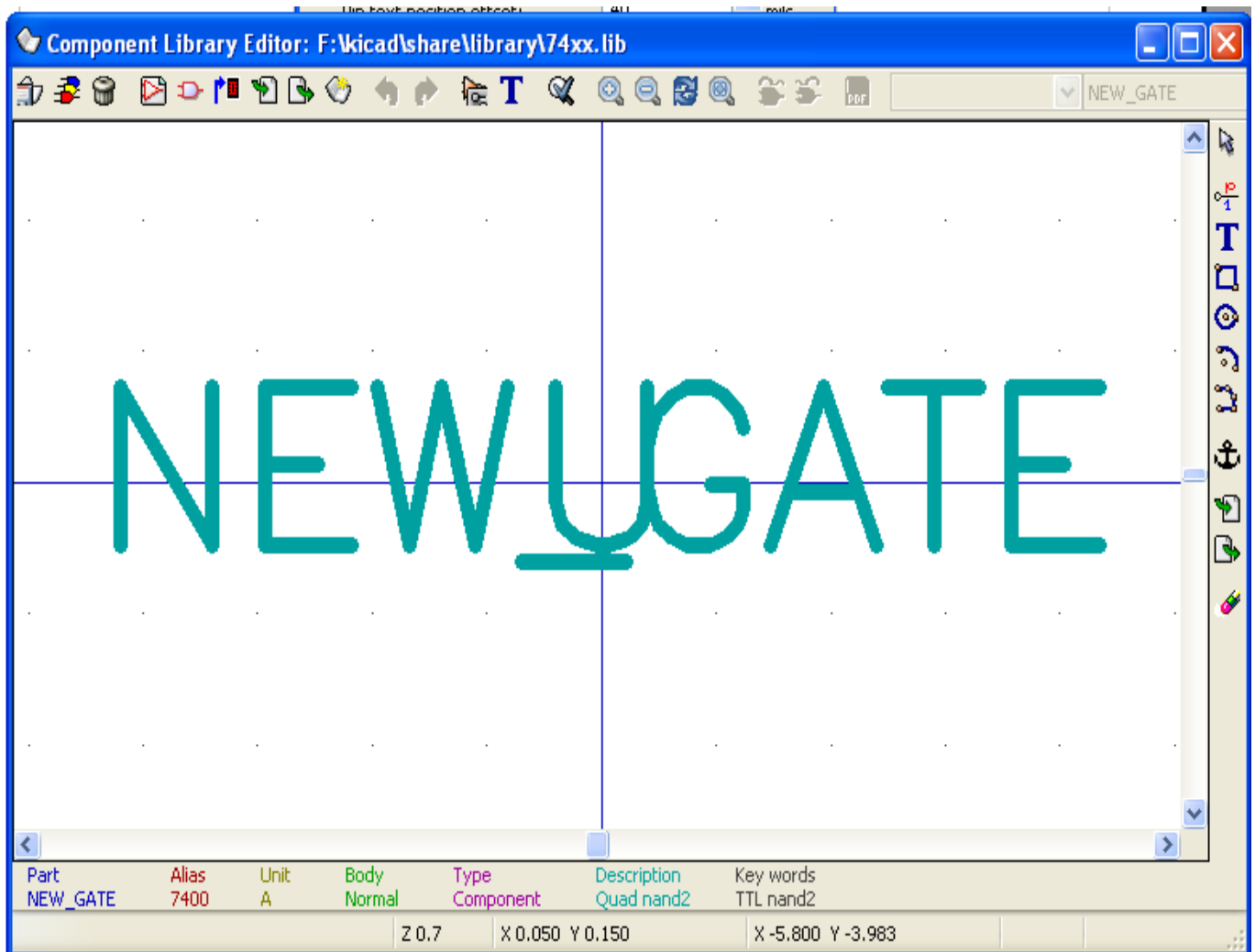
You will be asked for a component name to load it in the library (this name is also the field value for Libedit and used as default value for the field « Value » in schematic editor), the reference (U, IC, R...), the number of parts per package (for example a standard component 7400 A 4 parts per package) and if a converted representation exists (de Morgan as standard).

If the field reference is left empty, the reference will default to “U”.

All this data can be set later, but it is preferable to set it at the beginning of component design.







The beginning of a component look like this :



11.4.2 - Creation based on another component

When a component strongly looks like another, it is often advantageous to load this other component, and to modify it. Then you have to :

- Load the component which will be used as a model.
- Click on .
Or modify its name (right click on the name and edit the text « Value »).
User will be prompted for the new component name.
- If the model component has aliases, user will be prompted to remove aliases from new component which conflict with model. If the answer is "no" the new component creation will be aborted.
- Modify the component name
- Edit the new component.
- Save the new part into the loaded library with .
or save to a new library with 
or (if you want to save this new component in an other existing library) select the other library and save the new part.
- Save the library to disk with File Update. 

11.4.3 - Editing of the main characteristics

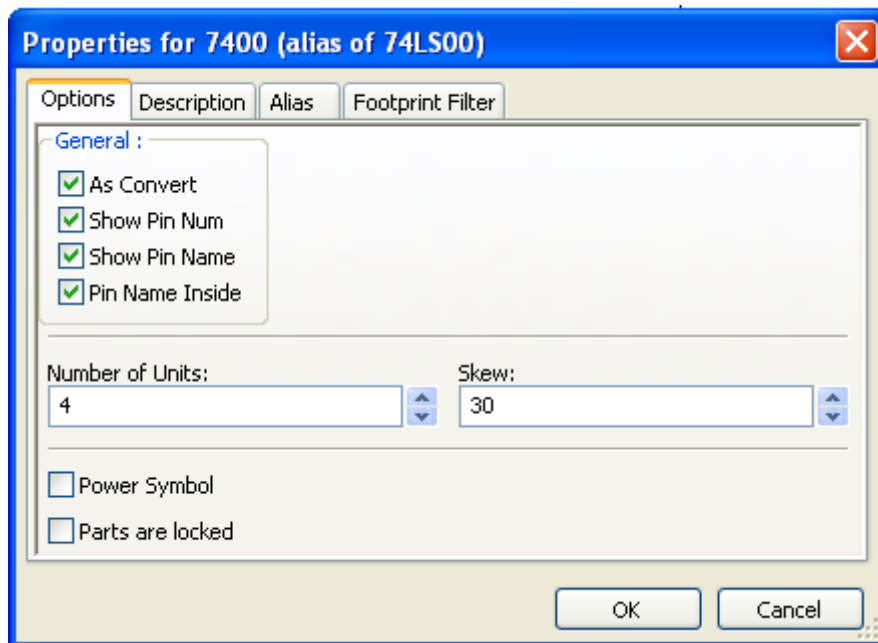
The main characteristics are:

- The number of parts per package.
- The presence of a converted representation.
- Associated documentation.
- The update of various fields.

These characteristics should be correct, because they are requested during the component creation, or they come from the model component.

So, however they are modified, it is necessary to call the edit command .

The editing window then appears as follows:



The **important options** that define the general properties are:

Number of Units to define the number of parts per package.

As Convert : check if the component has a double representation.

It is important that these two parameters are correctly set, because when pins are edited or created, the corresponding pins of all the parts will be published or created together.

If you increase the number of parts after pin creation/editing, there will be additional work introduced by this increase. Nevertheless, it is possible to modify these parameters at any moment.

Graphic options:

- **Show Pin Num** and
- **Show Pin Name**

defines the visibility of the pin number and pin name text (these texts will be visible if the corresponding options are active).

The option :

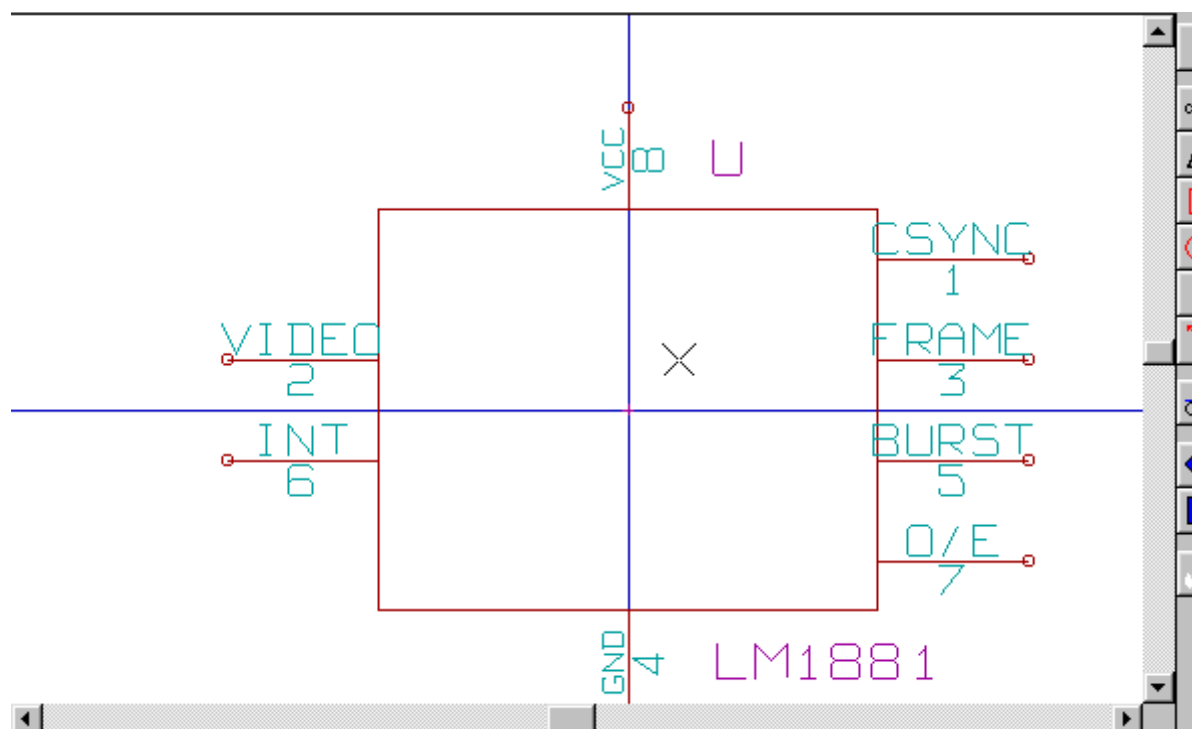
- **Pin Inside**

defines the pin name position : this text will be displayed inside the component outline if the option is active).

In this case the Pin Name Skew parameter defines the shift of the text towards the interior.



A value from 30 to 40 (in 1/1000 inch) is reasonable.

The example below shows the same component, with the Pin Inside option unchecked (notice the position of the names and pin numbers):

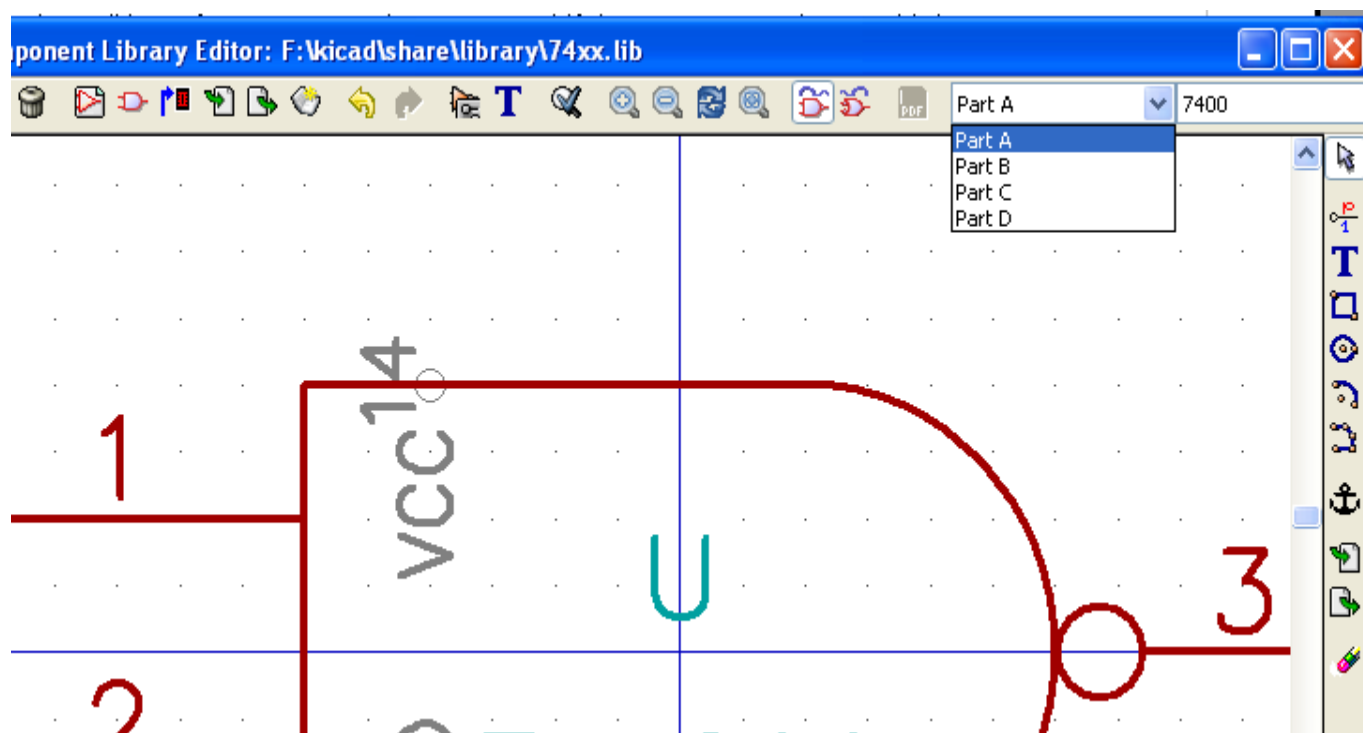


11.4.4 - Multi-part components

During the editing of component elements, and if the component has multiple parts or representations, you will have to select the different parts or representations of this component.

For the representation selection click on  or .

For the part selection



11.5 - Component design

The vertical toolbar allows you to place all the elements of a component :



To draw a component, you can use the following graphic elements:

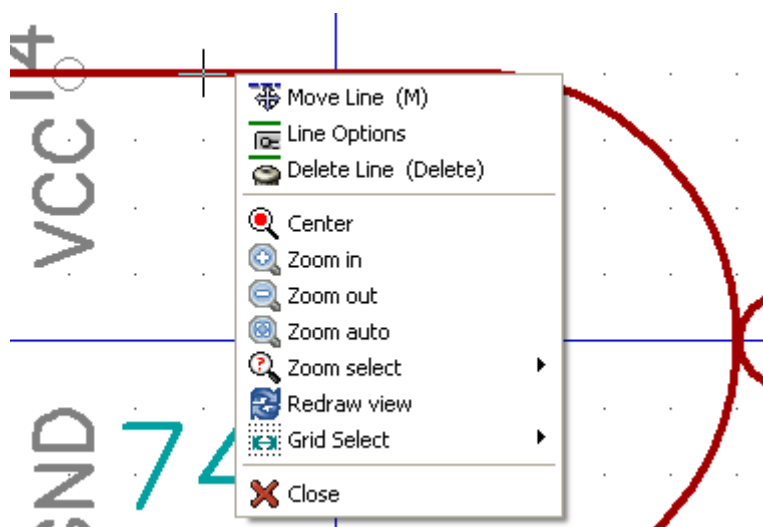
- Lines (and polygons, simple or filled).
- Rectangles
- Circles
- Arcs of circle.
- Texts (others that fields and texts of pins).

The pins and the fields (value, reference) are treated differently, because they are not pure graphic elements.

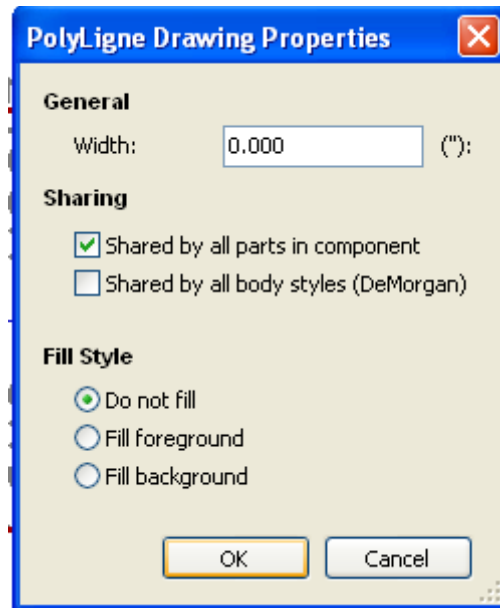
11.5.1 - Graphic elements membership options

Each graphic element can be defined as ordinary or specific, either to a type of representation (Normal or converted), or to the different parts of a component.

The options menu is accessible by a right click on the concerned element (here a line) :



or by a double-click on this element:



The normal options of a graphic element are:

- **Shared by all parts in component** *checked*, because generally the different parts of a component have the same graphic representation, and it is thus enough to draw one part only.
- **Shared by all body styles (DeMorgan)** *unchecked*, because a double representation is introduced, to have a different graphic representation with each kind of representation.

It will then be necessary to draw each graphic representation.

For elements of the type "polygon" (lines successively traced) the **Fill background** or **Fill foreground** option allow you to generate a filled polygon.

However, you can thus treat the case (fortunately rare) multi-parts components, designed with different graphic types, by uncheck **Shared by all parts in component**.

Each part will then have to be drawn, and if the option "Specific to the representations" is checked, for each part it will be necessary to draw the two representations.

Finally it can be interesting to check the option **Shared by all body styles (DeMorgan)** for components drawn with the modern IEEE standard, since the graphics essentials are identical in the normal and converted representations.

11.5.2 - Geometrical graphic elements

Their design is possible using tools :

- Lines and polygons, simple or Filled if option is checked.
- Rectangles defined by a diagonal.
- Circles defined by the center and a point of the circumference.
- Arcs defined by the starting and ending point of the arc and its center. An arc goes from 0 to 180 degrees.

11.5.3 - Graphic elements of text type



Allows the creation of graphic text (free text).

The text is always readable, even if the component is mirrored.

11.6 - Pin creation and editing



Click on  to create a pin.

The editing is done whilst double-clicking on the pin. You can right-click to open the fast editing menu .

Pins must be created carefully, because any error will have consequences for the PCB design, or will make E.R.C function inefficient.

Any pin already placed can be re-edited, erased or moved.

11.6.1 - Pins overview

A pin is defined by its form (length, graphic aspect), its name and its “number” which is not always a number (PGA socket pins are defined by a letter and a number, like A12 or AB45)

In EEschema, the “ pin number” is defined by a set of 4 letters or numbers.

For E.R.C. controls, the “electric” type (input, output, 3states...) must also be defined.

If this type isn't well defined, E.R.C control will be inefficient.

Note:

- Avoid spaces in the pin names and numbers.
- An pin name with an inverted signal begins with the symbol “~”.
- If the name is reduced to this single symbol, the pin is regarded as unnamed.
- Pin names starting with “#”, are reserved for power port symbols.
- A pin number consists of 1 to 4 letters or numbers. 1,2,..9999 are valid numbers, but also A1, B3... (standard PGA notation) or Anod, Gnd, Wine...

11.6.2 - Multi-part components, double representation.

Let us recall that, particularly for logic gates, a symbol can have two representations (representation known as “De Morgan”, and an IC can include several parts (e.g. several NOR gates)

For certain IC's, you may desire several different elements of graphics and pins.

For example a relay can be represented with different elements:

- Coil
- switch contact 1
- switch contact 2

The management of the multi-part IC's, and the components with double representation is flexible.

Indeed, a pin can be:

- Common or specific to different parts.
- Common run to both representation or specific to each representation.

By default, pins are specific to each representation of each part, because their number differs for each part, and their design is different for each representation.

When a pin is common you just have to draw it once (e.g. in the case of power pins)

It is also the case of the design which is almost always identical for every part (but differs between the normal and the converted representation).

11.6.3 - Pins: basic option

The components with multiple parts and/or representations pose a particular problem for pin creation and editing.

Insofar as the majority of the pins are specific to each part (because their pin number is specific to each part) and to each representation (because their form is specific to each representation), the creation and the edition of the pins would thus be likely to be long and tiresome.

In fact, Eeschema allows the simultaneous handling of the pins :

By default, for the multi-part components and/or double representations, these modifications are made for all the pins corresponding to the parts and the representations when you create, edit (except form, and number) delete or move a pin, (i.e. **for all the pins placed at the same co-ordinate**).


- For the design, the modifications made for the current representation, are for all parts.
- The pin numbers are modified for the current part, for the 2 representations.
- The names are modified independently.

This dependence was established to allow fast modifications in most of the cases.

This dependence in the modifications can be disabled in the Options Menu, allowing you to create components with parts and representations of completely independent characteristics.

This dependence option is managed by the tool



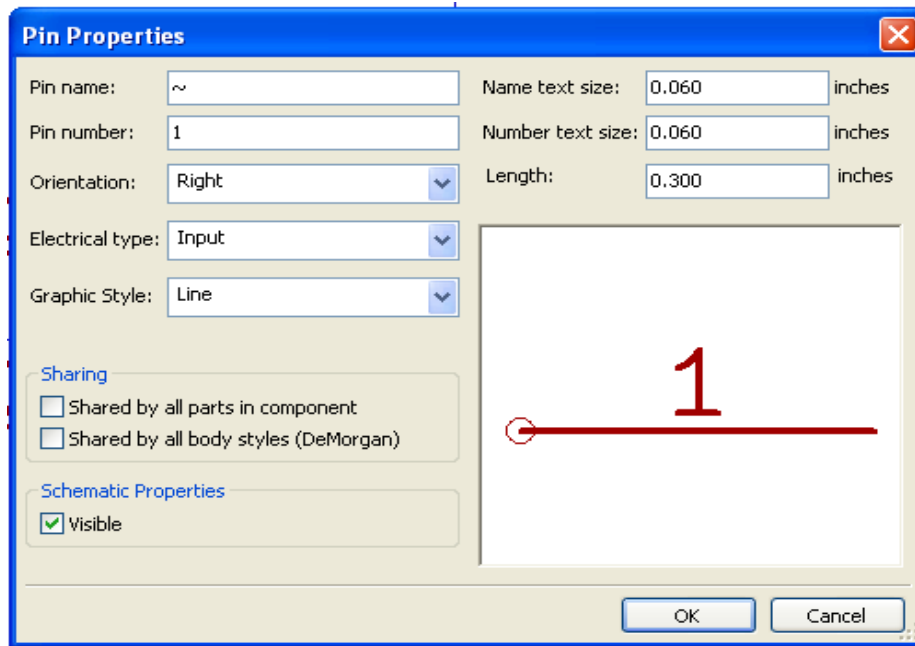
- If  is not activated (not highlighted): the edits will apply to all parts and to all representations. ***This is the normal option.***



- If  is activated (highlighted) : the edits will apply only to the current part and in the current representation (i.e. on what you see on the screen). ***This option is rarely used.***

11.6.4 - Pins: Defining characteristics

The pin properties window allows you to edit a pin's characteristics.



This menu pops up automatically as you create a pin, or while you doubleclick on an existing pin. It allows you to define or modify:

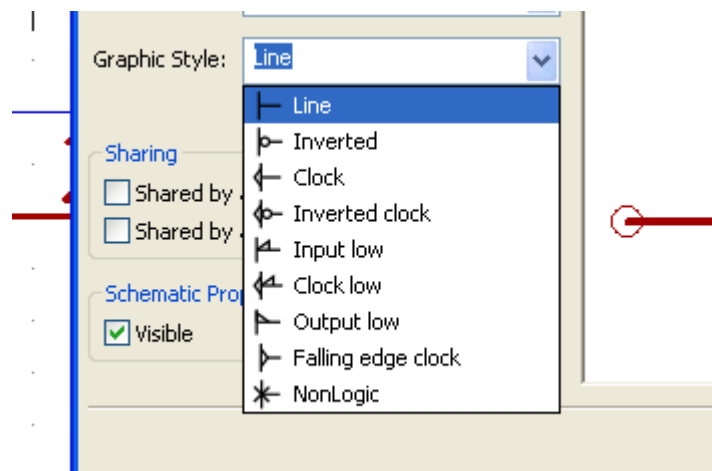
- The name and name's size of a pin.
- The number and size of a pin number.
- Pin length.
- Electrical type and design.
- Its membership (common to normal and "Morgan" representation or not)
- Invisible pin (used for power pins).

Recall:

- The pin name begins with a "~", for **inverted** signals.
- If the name is reduced to this character only, the pin is regarded as without name.
- The pin number consists of 1 to 4 characters (letters or digits).
 - 1,2..9999 are thus valid numbers, but also A1, B3... (standard notation PGA) or Anod, Gnd, V.in ...

11.6.5 - Pins shapes

You can see on the figure below different pin shapes :



The choice of the form has a purely graphic influence, and does not have any influence on the E.R.C or netlist functions.

11.6.6 - Pins : Electric types

The choice of the type is important, for E.R.C. function.

The choice is commonplace for **input** and **output** pins of IC's.

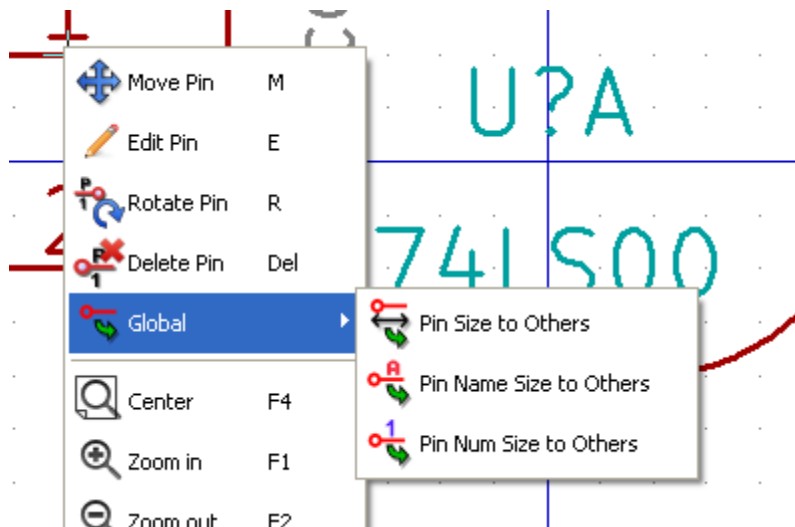
- **The BiDi** type indicates bidirectional pins commutable between input and output (microprocessors data bus for

example).

- The type **3 States** is the usual 3 states output.
- **The Passive** type is used for passive component pins, resistors, connectors....
- **The Unspec** type (unspecified) can be used when E.R.C. check doesn't matter.
- **The Power In** type is to be used for the components power pins.
In particular **if the pin is a power in port, and is declared as "Invisible", it is not displayed in schematic diagram, and it is automatically connected to the other pins of the same type and same name (Invisible Power In Pin)**.
- **Power Out** is for regulator outputs.
- You can use **Open Emitter** and **Open Collector** types too.

11.6.7 - Pins : global modifications


One can modify the length of all the pins, or the text sizes (name, part number), using the **Global** command of the Pop Up menu to set one of these three parameters.



To click on the parameter you want to modify and type the new value, which will then apply to all component pins, for the current representation.

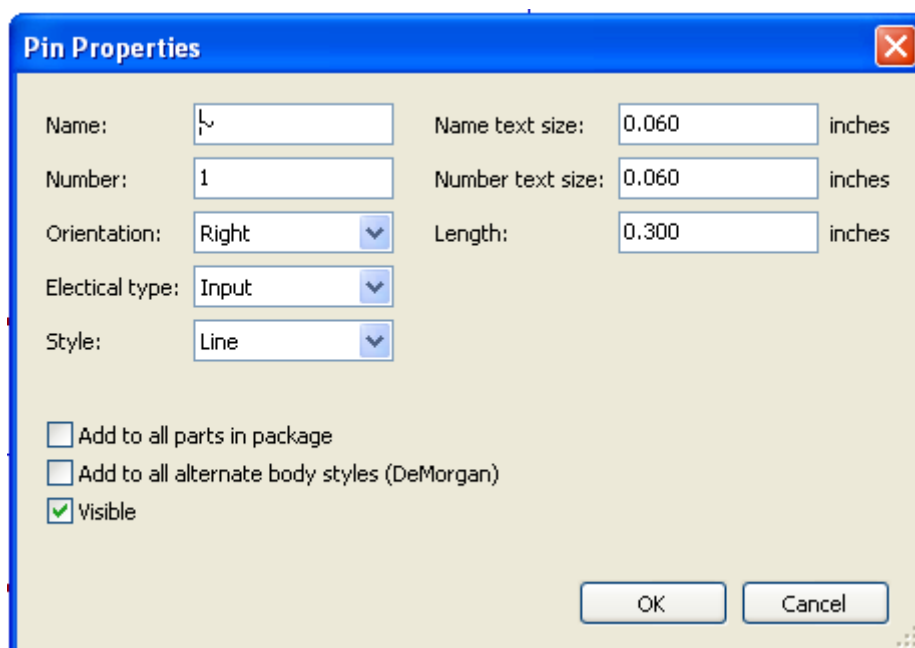
11.6.8 - Pins : Multi-part components and double representations

Various parts or representation (such as met in a 7400, 7402...) can need a complementary edition. This complementary work will be limited if the following precautions are taken:

- General option **Edit pin part per part**  must remain unchecked.
- The power pins will be created with the attribute Common Unit and Common Convert active (They can be also invisible (No Draw)).

The correct setup is like this:

When the other pins have been created, they have been created for each part and each representation.



Pin Properties

Name: Name text size: 0.060 inches

Number: 1 Number text size: 0.060 inches

Orientation: Right Length: 0.300 inches

Electrical type: Input

Style: Line

☐ Add to all parts in package

☐ Add to all alternate body styles (DeMorgan)

☒ Visible

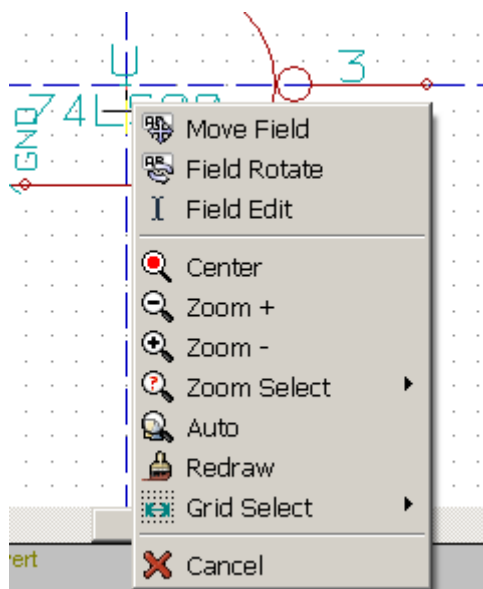
OK Cancel


For example the output pin of part A of a 7400 will have been created by Eeschema in 8 specimens: 2 per part (there are 4 parts A, B, C, D and for each part, the normal representation, and the converted representation known as of Morgan.) However will have, at the beginning, probably correctly created the part A in its normal representation. It will thus be necessary for each part :

- To select the converted representation, and to edit the form and the length of each pin.
- For the other parts, to edit the pin numbers.

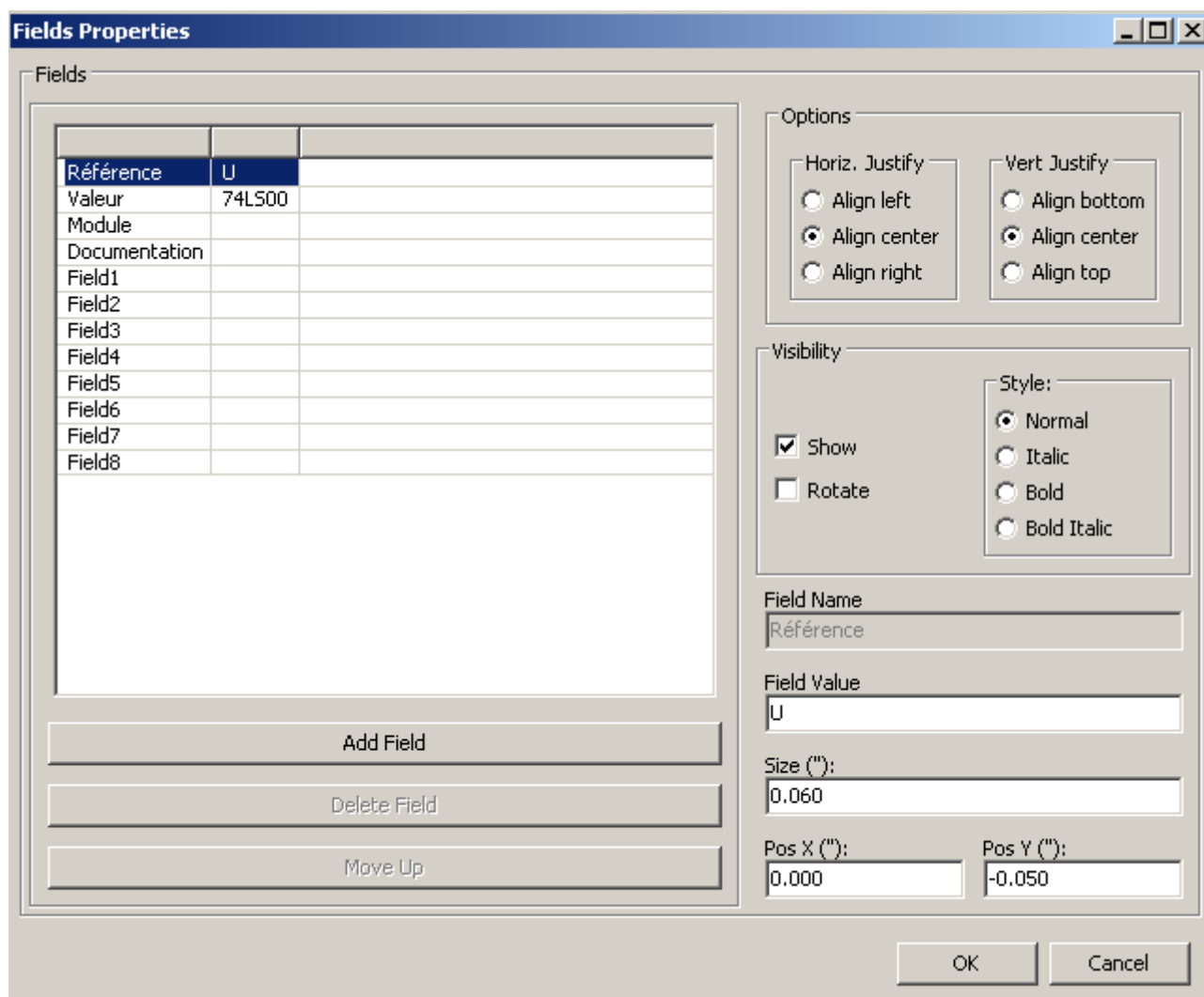
11.7 - Field Editing

For the already existing fields, you can use the fast editing commands with a right-click :



For more complete editing or empty fields, it is necessary to call the editing fields window : 

The dialog window is:



The folder *Reference* is selected here.

The fields are texts associated to the component, not to be confused with the texts belonging to the graphic representation of this component.

These fields are always available :

- Value
- Reference
- Name of the associated module (footprint for the PCB)
- link to a documentation file (mainly intended to be used in schematic).
- Template fields defined in schematic editor (for comments)

The value and reference fields are defined during the component creation, and can be modified here.

It can possibly be useful to edit the Name field of the associated module to directly generate netlists (for the PCB software) including the module (footprint) name.

The Name field of the associated diagram is of particular use for some other electronic CAD software.

For the library, the edition of the Value and Reference fields allows the definition of their size and position.

Important remarks :

- **Modifying the text** of the **value** field, equals to **create a new component**, starting from an old one used as a model, because this new component has the name contained in the **value field** when you save it in the library.
- To edit an invisible field (i.e. empty, because even if the field has the Invisible attribute, it is displayed in LibEdit) you will have to use the general edition window above.

11.8 - Power port symbols : Creation

Eeschema

The power port symbols are created just like usual components.

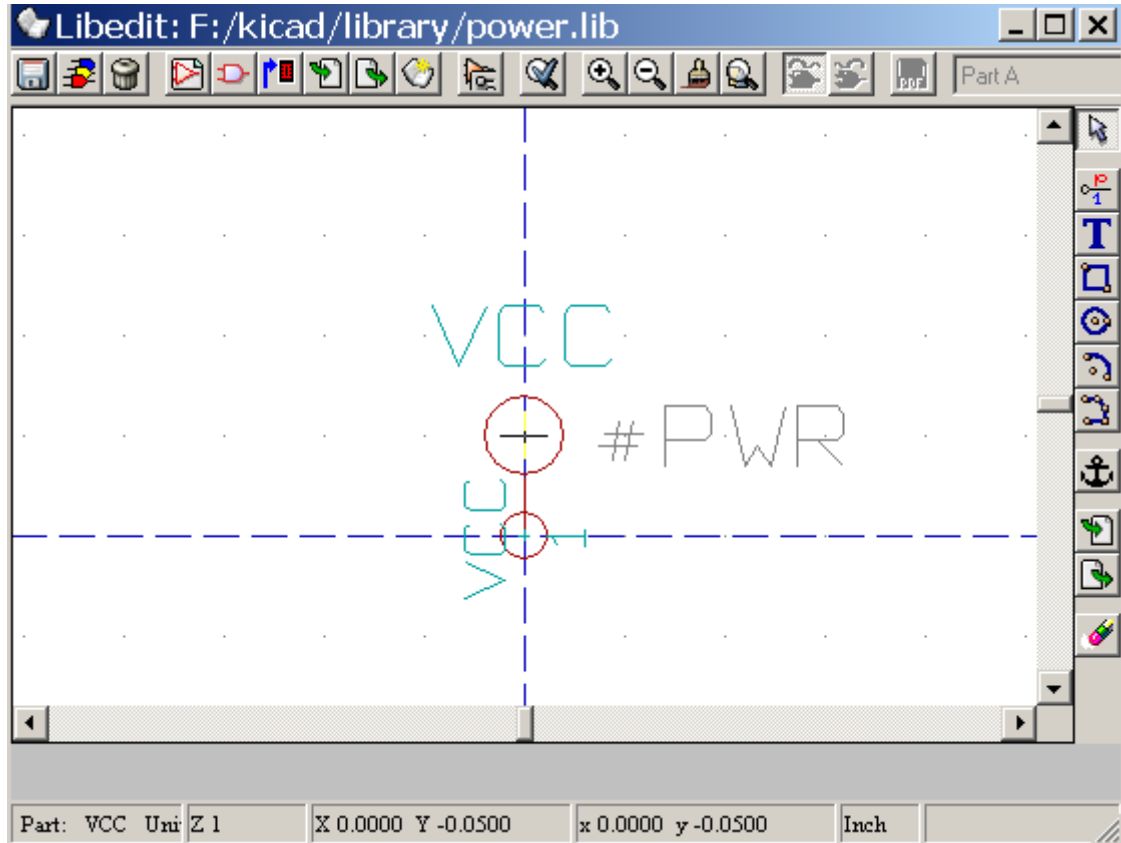
It may be useful to gather them in a dedicated library such as **Power.lib**.

They consist in a graphic symbol (the desired form) and in a **pin** of the **type "Power Invisible"**.

They will thus be handled like any other component in the schematic capture software.

However, some precautions are essential.

Here a symbol (power + 5V) :



The symbol is carried out in the following way :

- A pin "**Invisible Power**" named **+ 5V** (important because this name will establish connection to the net + 5V), of pin number 1 (number of no importance) and null length.
- The shape is of the type "Line", and obviously the type is "Power" and the attribute is "Invisible".
- A graphics : here a small circle and a segment from the pin to the circle.
- The anchor of the symbol is on the pin.
- The value is **+ 5V** like the pin name, to display the value of this symbol (the pin being invisible by default, its name does not appear).
- The reference is **# + 5V** (thus displayed **# + 5V**) like the pin name. The reference text has no importance except the first character which must be "#". By convention, every component whose reference starts with this symbol will appear neither in the components list, nor in the netlist. Moreover, in Option of symbol, the reference is declared invisible.

The creation of a new power port symbol is easy and fast if you use another symbol as model.

You just have to :

- Load the model.
- Edit the **pin name** (which then takes the name of new power port).
- Edit the field **Value** (same name as the pin, if you want to display the power port value...).
- Save the new component.

Headers

[12 - LibEdit : Complements](#)

[12.1 - Overview](#)

[12.2 - Positioning the anchor](#)

[12.3 - Alias](#)

[12.4 - Fields:](#)

[12.5 - Component documentation](#)

[12.5.1 - Keywords](#)

[12.5.2 - Component documentation \(Doc\)](#)

[12.5.3 - Associated documentation file \(DocFileName\)](#)

[12.5.4 - Footprint filtering for CVPCB](#)

[12.6 - Symbol library](#)

[12.6.1 - Export/Create a symbol](#)

[12.6.2 - Import a symbol](#)

12 - LibEdit : Complements

12.1 - Overview

A component consist of several elements :

- Its graphic representation (geometrical shapes, texts).
- Pins.

- Fields, or associated texts, used by the post processors: netlist, components list...

Two fields are to be initialized : reference and value.

The name of the design associated with the component, and the name of the associated footprint, the other fields are the free fields, they can generally remain empty, and could be filled during schematic capture.

However, managing the documentation associated with any component facilitates the research, use and maintenance of the libraries.

The associated documentation consists of :

- A line of comment.
- A line of key words such as TTL CMOS NAND2..., separated by spaces
- An attached file name (for example an application note, a pdf file...). The default directory for attached files is **kicad/share/library/doc** and if not found, **kicad/library/doc**.
and under linux also in
/usr/local/kicad/share/library/doc
/usr/share/kicad/library/doc
/usr/local/share/kicad/library/doc

The key words allow you to selectively search for a component according to varying selection criteria.

Comment and key words are displayed in various menus, and particularly when you select a component in a library.

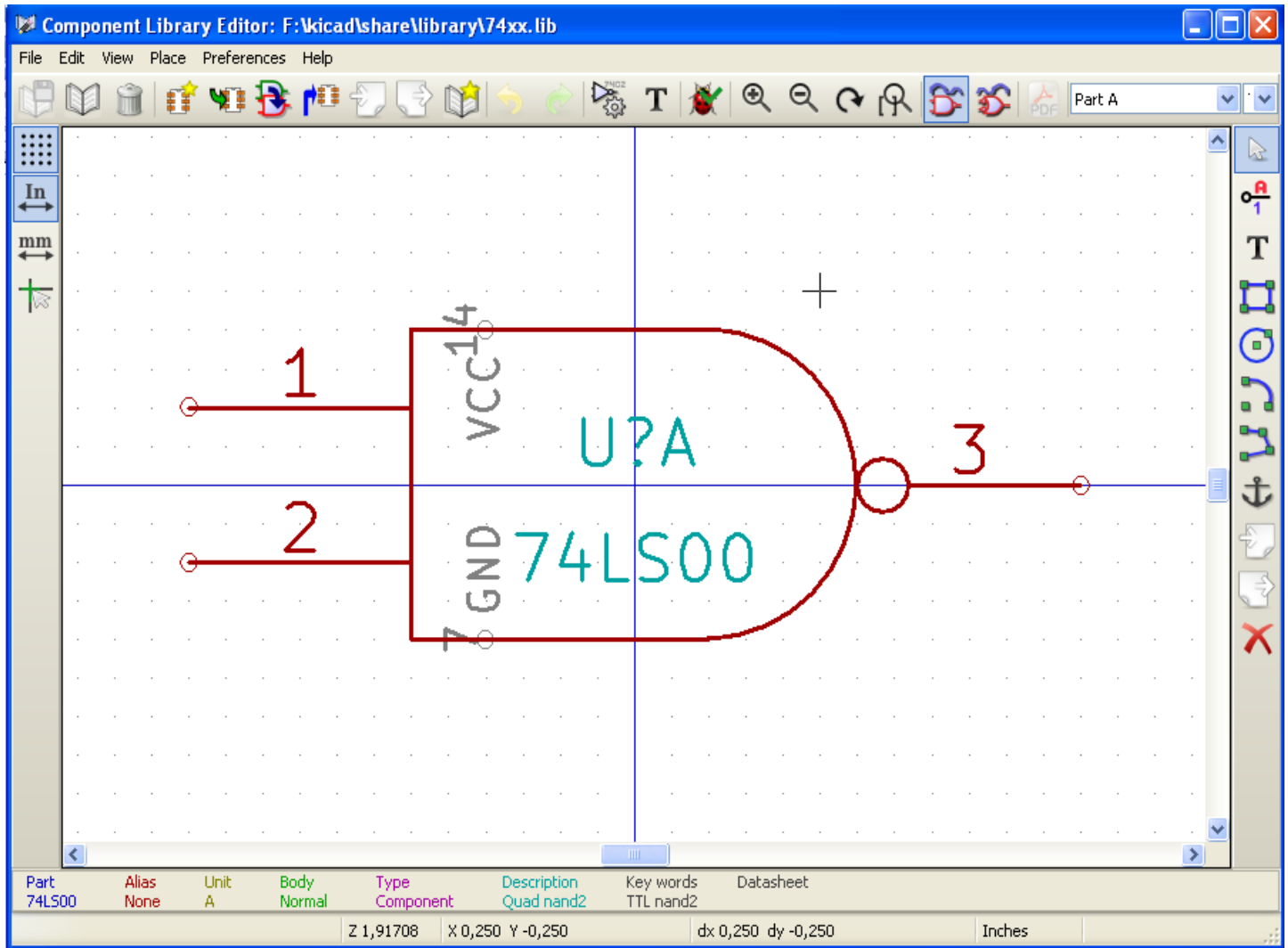
The component also has an anchoring point. A rotation or a mirror is made relatively to this anchor point, and during a placement, this point is used as a reference position. It is thus useful to position this anchor accurately.

A component can also have aliases, i.e. equivalent names. This allows you to considerably reduce the number of components that need to be created (for example, a 74LS00 can have aliases such as 74000, 74HC00, 74HCT00...).

Finally, the components are distributed in libraries (classified by topics, or manufacturer...), in order to facilitate their management.

12.2 - Positioning the anchor

The anchor is at the co-ordinates 0,0 shown by the blue axes displayed on the screen :



The anchor can be repositioned as follows :



Select the tool and then click on the new desired anchor position. The drawing will be automatically re-centered on this new anchor point.

12.3 - Alias

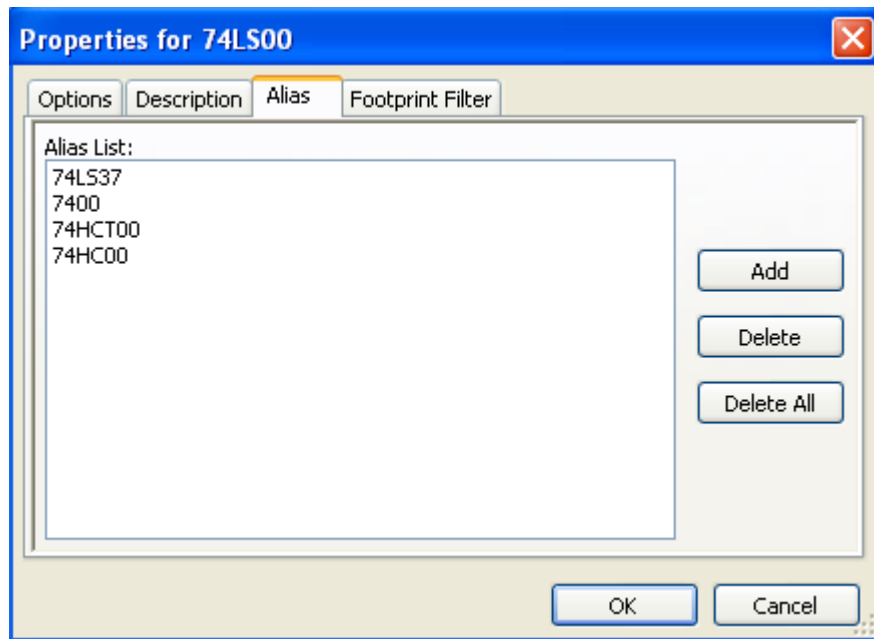
An alias is another name corresponding to the same component in the library.

Components with similar pin-out and representation can then be represented by only one component, having several aliases (ex: 7400 with alias 74LS00, 74HC00, 74LS37.).

The use of aliases allows you to build complete libraries much more quickly. In addition these libraries, being much more compact, are loaded more quickly.



To modify the list of aliases, you have to select the main editing window (tool) and select the **Alias** folder :



You can thus add or remove the desired alias.

The current alias cannot obviously be removed since it is edited.

To remove all aliases, **you have firstly to select the root component** (first component in the alias list in the window of selection of the main toolbar).

12.4 - Fields:



Field editor is called by

There are 4 special fields (texts attached to the component), and configurable user fields

Fields Properties

Fields

Name	Value
Reference	U
Value	74LS00
Footprint	
Datasheet	

Options

Horiz. Justify

☐ Align left
☒ Align center
☐ Align right

Vert. Justify

☐ Align bottom
☒ Align center
☐ Align top

Visibility

☒ Show
☐ Rotate

Style:

☒ Normal
☐ Italic
☐ Bold
☐ Bold Italic

Field Name

Reference

Field Value

U

Size ("):

0,060

Pos X ("):

0,000


Pos Y ("):

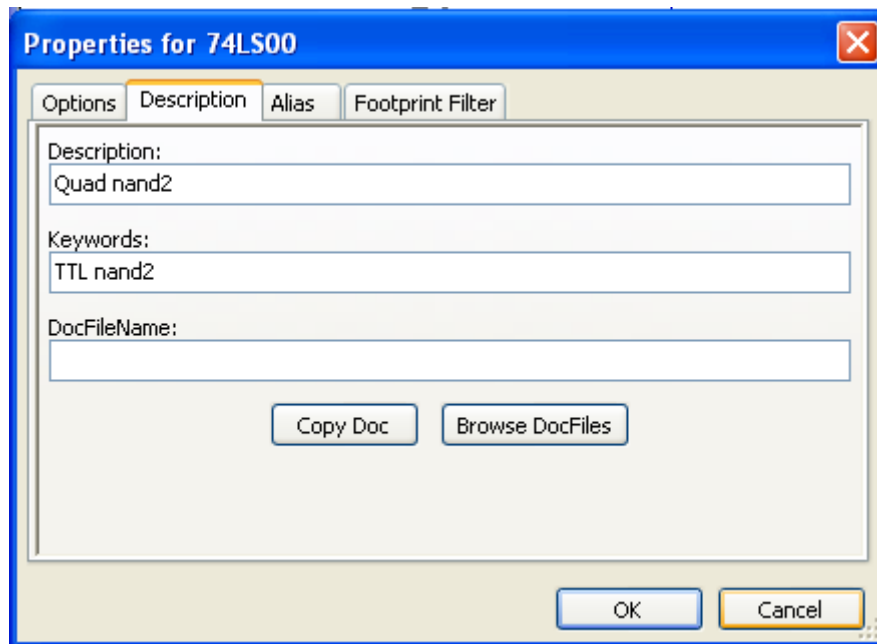
-0,050

Special fields:

- Reference
- Value: It is the component name in library, and the default value field in schematic.
- Footprint : footprint name used for the board.
Not very useful when using CvPcb to setup the footprint list, but mandatory if CvPcb is not used.
- Sheet: reserved (not used at this time).

12.5 - Component documentation

To edit documentation information, it is necessary to call the main editing window of the component (tool ) and to select the **Doc.** Folder :



Caution:

Be sure to select the right alias, or the root component, because this documentation is the only characteristic which differs between aliases.

The "Copy Doc" button allows you to copy the documentation information from the root component towards the currently edited alias.

12.5.1 - Keywords

The key words allow you to search in a selective way for a component according to specific selection criteria (function, technological family.)

The EESchema research tool isn't case sensitive.

The most current key words used in the libraries are :

- CMOS TTL for the logic families
- AND2 NOR3 XOR2 INV... for the gates (AND2 = 2 inputs AND gate, NOR3 = 3 inputs NOR gate).
- JKFF DFF... for JK or D flip-flop.
- ADC, DAC, MUX...
- OpenCol for the gates with open collector output.

Thus if in the schematic capture software, you search the component: by keys words **NAND2 OpenCol** EESchema will display the list of components having these 2 key words.

12.5.2 - Component documentation (Doc)

The line of comment (and key words) is displayed in various menus, particularly when you select a component in the displayed components list of a library and in the **ViewLib** menu.

If this Doc. file exists, it is also accessible in the schematic capture software, in the popup menu displayed by a right-click on a component.

12.5.3 - Associated documentation file (DocFileName)

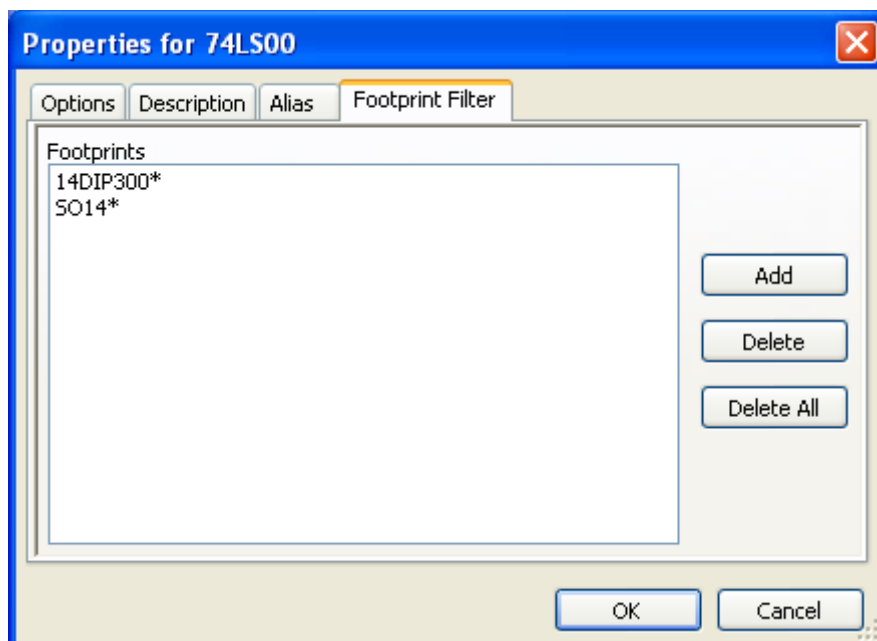
Indicates an attached file (documentation, application schematic...) available (pdf file,schematic diagram...).

12.5.4 - Footprint filtering for CvPcb

One can enter a list of allowed footprints for the component.

This list acts as a filter used by CvPcb to display the allowed footprints only.

A void list does not filter anything.

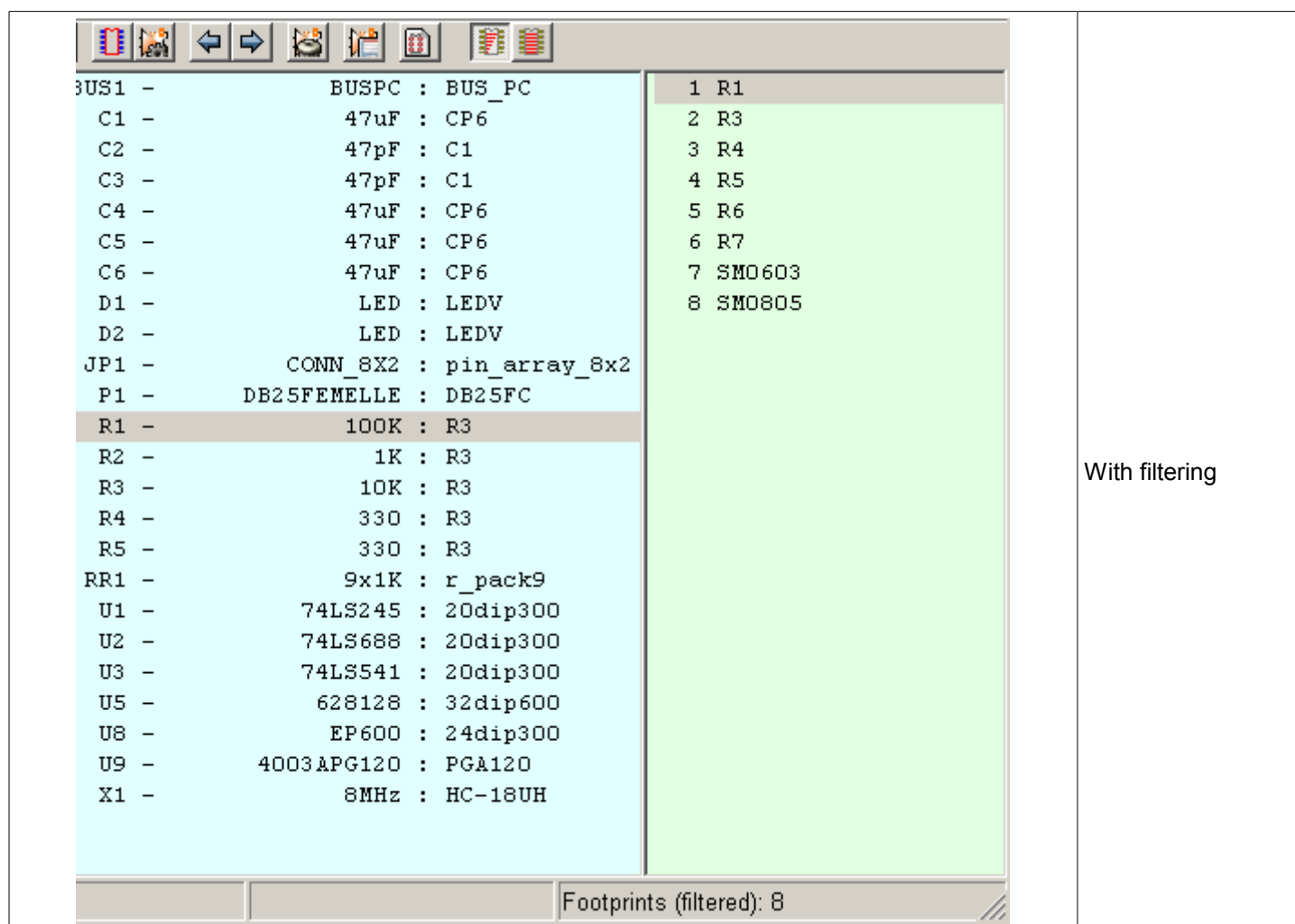


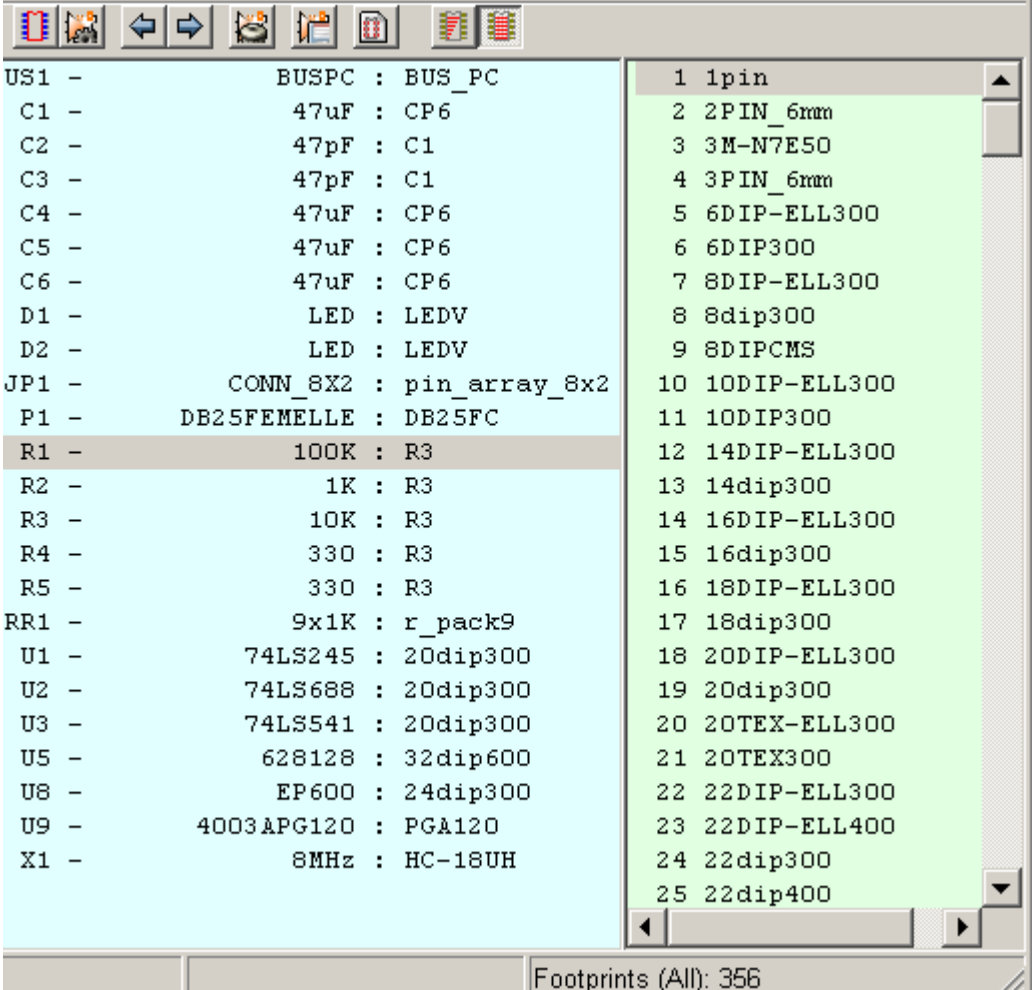
Wild cards are allowed.

S014* allows CvPcb to show all the footprints with a name starting by **S014**

For a resistor, **R?** shows all the footprints with a 2 letters name starting by **R**.

Here are samples: with and without filtering





Without filtering

12.6 - Symbol library

You can easily compile a graphic symbols library file containing frequently used symbols. This can be used for the creation of components (triangles, the shape of AND, OR, Exclusive OR gates...), for saving and subsequent re-use.

These files are stored by default in the library's directory and have a **.sym** extension.

The symbols are not gathered in libraries like the components because they are generally not very numerous.

12.6.1 - Export/Create a symbol



A component can be exported as a symbol with the tool

You generally create only one graphic, also it will be a good idea to delete all pins, if they exist.

12.6.2 - Import a symbol

That allows you to add graphics to a component you are editing.



A symbol is imported with the tool

Imported graphics are added as such as it was created in existing graphics.

Headings:

13 - Viewlib

13.1 - Role


13.2 - Main screen

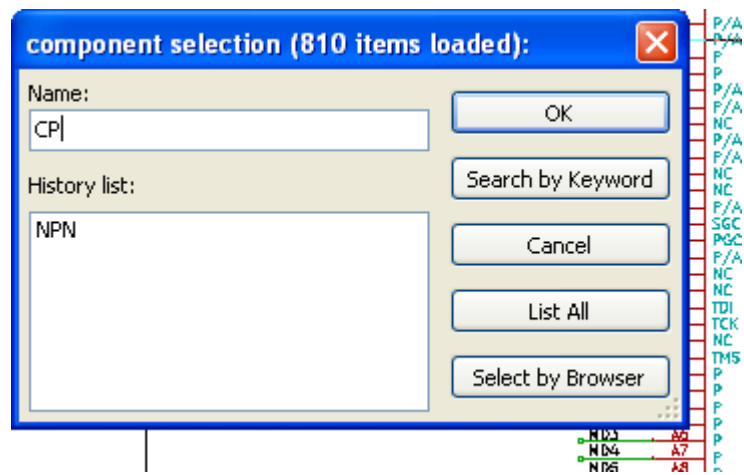
13.3 - Viewlib Toolbar

13 - Viewlib

13.1 - Role

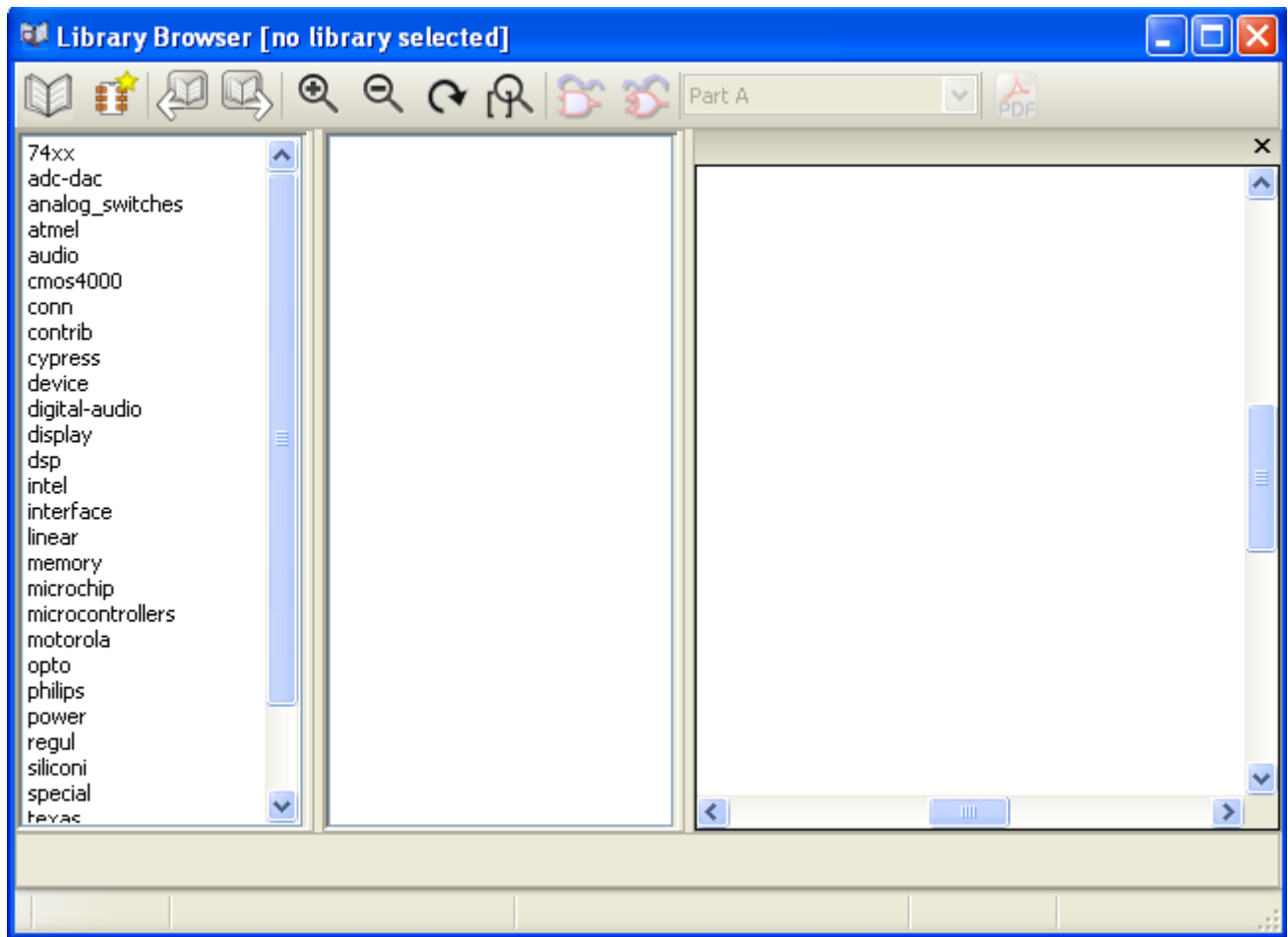
Viewlib allow you to examine the content of the libraries quickly.

Viewlib is called by the tool  ,

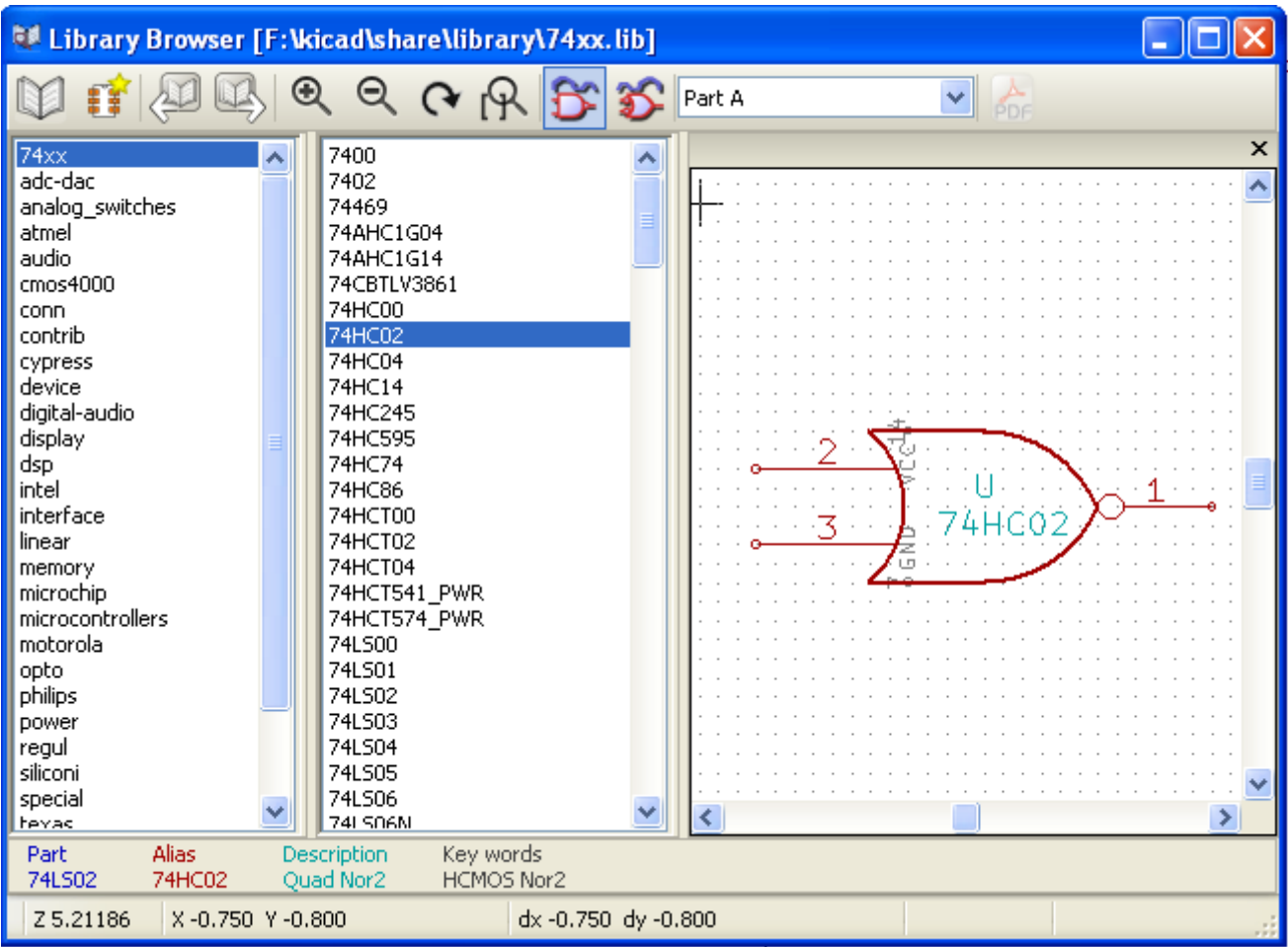


or by the place component tool:

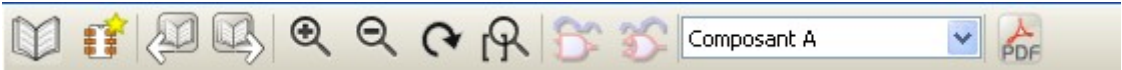
13.2 - Main screen



To examine a library, you have to select it in the list displayed to the left.
Its content then appears in the second list which allow you to select a component.









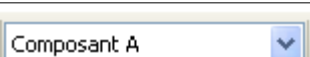
13.3 - Viewlib Toolbar






or (when called by the place component dialog frame from Eeschema)



The commands are :

	Selection of the desired library (which can be also selected in the displayed list).
	Selection of the component (which can be also selected in the displayed list).
	Display the previous component.
	Display the next component.
	Zoom management.
	Selection of the representation (normal or converted) if exist.
	Selection of the part (if multi-part component).

Eeschema

	Selection of the desired library (which can be also selected in the displayed list).
	Exists only when called by the place component dialog frame from Eeschema Display the associated documents (if they exist).
	Close Viewlib et and place the selected component in Eeschema.

14 - Customize Netlists and BOM

Headers:

[14 - Customize Netlists and BOM](#)

[14.1 - The Intermediate Netlist:](#)

[14.1.1 - Schematic sample:](#)

[14.1.2 - The Intermediate Netlist file sample:](#)

[14.2 - Conversion to a new format:](#)

[14.3 - XSLT approach:](#)

[14.3.1 - Sample to create a Pads-Pcb netlist file](#)

[14.3.2 - Sample to create a Cadstar netlist file](#)

[14.3.2.1 - The Style-Sheet file:](#)

[14.3.2.2 - The output file:](#)

[14.3.3 - Sample to create a OrcadPCB2 netlist file](#)

[14.3.3.1 - The Style-Sheet file:](#)

[14.3.3.2 - The output file:](#)

[14.3.4 - Using Eeschema plugins interface:](#)

[14.3.4.1 - Init the Dialog window:](#)

[14.3.4.2 - Parameters:](#)

[14.3.4.3 - The command line:](#)

[14.3.4.4 - Command line format:](#)

[14.4 - The Intermediate Netlist structure:](#)

[14.4.1 - General structure:](#)

[14.4.2 - The header section:](#)

[14.4.3 - The components section:](#)

[14.4.3.1 - Note about time stamps for components:](#)

[14.4.4 - The libparts section:](#)

[14.4.5 - The libraries section:](#)

[14.4.6 - The nets section:](#)

[14.5 - More info about xsltproc:](#)

[14.5.1 - Introduction](#)

[14.5.2 - Synopsis](#)

[14.5.3 - Command Line Options](#)

[14.5.4 - Return values](#)

[14.5.5 - More Information](#)

14.1 - The Intermediate Netlist:

BOM and netlists can be converted from an Intermediate Netlist file created by Eeschema.

This file uses an XML syntax.

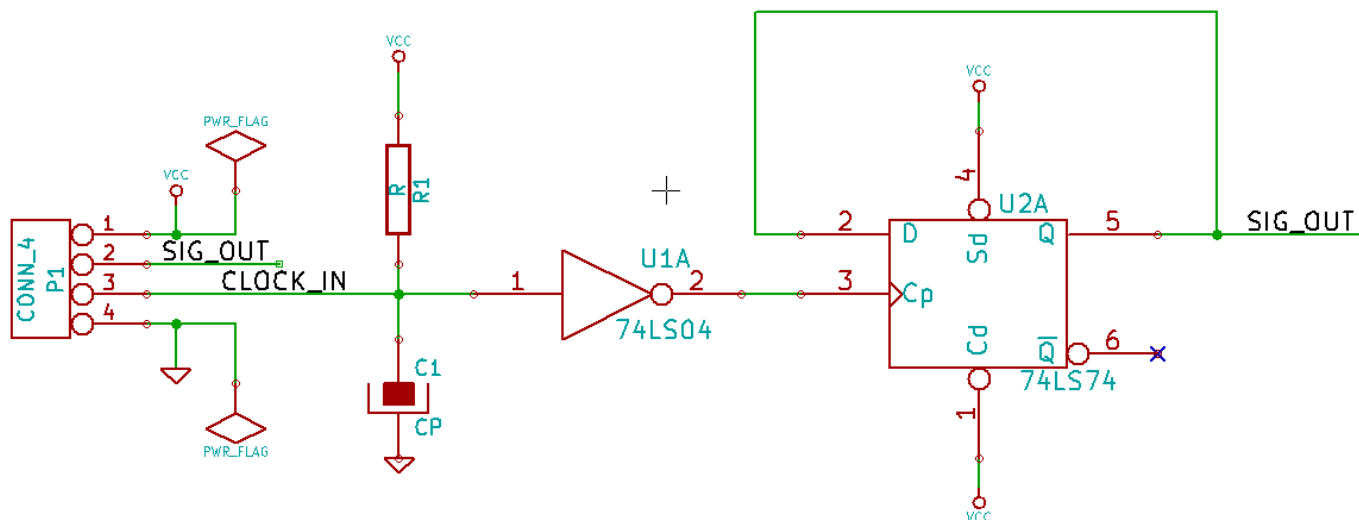
This file is called Intermediate Netlist.

But because it includes a lot of data about components, it can also be used to create BOM or others reports, not just netlists.

Depending on the output (BOM, new netlist), only some of sections in this Intermediate Netlist will be used.

Here is a sample:

14.1.1 - Schematic sample:



14.1.2 - The Intermediate Netlist file sample:

The corresponding intermediate netlist (using XML syntax) is:

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

```

    <sheetpath names="/" tstamps="/" />
    <tstamp>4C6E208A</tstamp>
  </comp>
</components>
<libparts>
  <libpart lib="device" part="C">
    <description>Condensateur non polarise</description>
    <footprints>
      <fp>SM*</fp>
      <fp>C?</fp>
      <fp>C1-1</fp>
    </footprints>
    <fields>
      <field name="Reference">C</field>
      <field name="Value">C</field>
    </fields>
    <pins>
      <pin num="1" name="~" type="passive"/>
      <pin num="2" name="~" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="device" part="R">
    <description>Resistance</description>
    <footprints>
      <fp>R?</fp>
      <fp>SM0603</fp>
      <fp>SM0805</fp>
      <fp>R?-*</fp>
      <fp>SM1206</fp>
    </footprints>
    <fields>
      <field name="Reference">R</field>
      <field name="Value">R</field>
    </fields>
    <pins>
      <pin num="1" name="~" type="passive"/>
      <pin num="2" name="~" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="conn" part="CONN_4">
    <description>Symbole general de connecteur</description>
    <fields>
      <field name="Reference">P</field>
      <field name="Value">CONN_4</field>
    </fields>
    <pins>
      <pin num="1" name="P1" type="passive"/>
      <pin num="2" name="P2" type="passive"/>
      <pin num="3" name="P3" type="passive"/>
      <pin num="4" name="P4" type="passive"/>
    </pins>
  </libpart>
  <libpart lib="74xx" part="74LS04">
    <description>Hex Inverseur</description>
    <fields>
      <field name="Reference">U</field>
      <field name="Value">74LS04</field>
    </fields>
    <pins>
      <pin num="1" name="~" type="input"/>
      <pin num="2" name="~" type="output"/>
      <pin num="3" name="~" type="input"/>
      <pin num="4" name="~" type="output"/>
      <pin num="5" name="~" type="input"/>
      <pin num="6" name="~" type="output"/>
      <pin num="7" name="GND" type="power_in"/>
      <pin num="8" name="~" type="output"/>
      <pin num="9" name="~" type="input"/>
      <pin num="10" name="~" type="output"/>
      <pin num="11" name="~" type="input"/>
      <pin num="12" name="~" type="output"/>
      <pin num="13" name="~" type="input"/>
    </pins>
  </libpart>

```

```

    <pin num="14" name="VCC" type="power_in"/>
  </pins>
</libpart>
<libpart lib="74xx" part="74LS74">
  <description>Dual D FlipFlop, Set & Reset</description>
  <docs>74xx/74hc_hct74.pdf</docs>
  <fields>
    <field name="Reference">U</field>
    <field name="Value">74LS74</field>
  </fields>
  <pins>
    <pin num="1" name="Cd" type="input"/>
    <pin num="2" name="D" type="input"/>
    <pin num="3" name="Cp" type="input"/>
    <pin num="4" name="Sd" type="input"/>
    <pin num="5" name="Q" type="output"/>
    <pin num="6" name="~Q" type="output"/>
    <pin num="7" name="GND" type="power_in"/>
    <pin num="8" name="~Q" type="output"/>
    <pin num="9" name="Q" type="output"/>
    <pin num="10" name="Sd" type="input"/>
    <pin num="11" name="Cp" type="input"/>
    <pin num="12" name="D" type="input"/>
    <pin num="13" name="Cd" type="input"/>
    <pin num="14" name="VCC" type="power_in"/>
  </pins>
</libpart>
</libparts>
<libraries>
  <library logical="device">
    <uri>F:\kicad\share\library\device.lib</uri>
  </library>
  <library logical="conn">
    <uri>F:\kicad\share\library\conn.lib</uri>
  </library>
  <library logical="74xx">
    <uri>F:\kicad\share\library\74xx.lib</uri>
  </library>
</libraries>
<nets>
  <net code="1" name="GND">
    <node ref="U1" pin="7"/>
    <node ref="C1" pin="2"/>
    <node ref="U2" pin="7"/>
    <node ref="P1" pin="4"/>
  </net>
  <net code="2" name="VCC">
    <node ref="R1" pin="1"/>
    <node ref="U1" pin="14"/>
    <node ref="U2" pin="4"/>
    <node ref="U2" pin="1"/>
    <node ref="U2" pin="14"/>
    <node ref="P1" pin="1"/>
  </net>
  <net code="3" name="">
    <node ref="U2" pin="6"/>
  </net>
  <net code="4" name="">
    <node ref="U1" pin="2"/>
    <node ref="U2" pin="3"/>
  </net>
  <net code="5" name="/SIG_OUT">
    <node ref="P1" pin="2"/>
    <node ref="U2" pin="5"/>
    <node ref="U2" pin="2"/>
  </net>
  <net code="6" name="/CLOCK_IN">
    <node ref="R1" pin="2"/>
    <node ref="C1" pin="1"/>
    <node ref="U1" pin="1"/>
    <node ref="P1" pin="3"/>
  </net>

```

```
</nets>
</export>
```

14.2 - Conversion to a new format:

By applying a filter to this Intermediate Netlist file, one can generate netlists in a lot of others format, and B.O.M. Because this is a text to text transform, this filter can be easily written using PYTHON or XSLT.

XSLT itself is a an XML language to define transformation for XML.

When using XSLT, *Xsltproc* program can be used to read an XML file, applies a style-sheet (the "filter" and save the resulting output.

So the used has to create a style-sheet file using XSLT conventions.

The full conversion process is handle by Eeschema, and is transparent.

14.3 - XSLT approach:

The XSL Transformations (XSLT) documentation is here:

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

14.3.1 - Sample to create a Pads-Pcb netlist file

This format has two sections:

- A footprint list
- A Nets list (grouping pads references by nets)

Here is a style-sheet sample to convert 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">
```

Eeschema

```
<xsl:text>*SIGNAL* </xsl:text>
<xsl:choose>
  <xsl:when test = "@name != '' ">
    <xsl:value-of select="@name"/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:text>N-</xsl:text>
    <xsl:value-of select="@code"/>
  </xsl:otherwise>
</xsl:choose>
<xsl:text>&nl;</xsl:text>
<xsl:apply-templates select="node"/>
</xsl:if>
</xsl:template>

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

</xsl:stylesheet>

Here is the final output, after applying *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*
```

And the command line to convert the netlist files is:

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

14.3.2 - Sample to create a Cadstar netlist file

This format has two sections:

- A footprint list
- A Nets list (grouping pads references by nets)

14.3.2.1 - The Style-Sheet file:

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

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

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

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

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

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

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



```

        <xsl:otherwise>
            <xsl:text>"</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:text>&nl;</xsl:text>
</xsl:template>

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

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

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

</xsl:stylesheet>

```

14.3.2.2 - The 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
```

14.3.3 - Sample to create a OrcadPCB2 netlist file

This format has only one section: the footprint list:

Each footprint includes its list of pads with reference to a net.

14.3.3.1 - The Style-Sheet file:

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

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

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

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

<!--
Netlist header
Creates the entire netlist
(can be seen as equivalent to main function in C
-->
<xsl:template match="/export">
```

```

<xsl:text>( { EESchema Netlist Version 1.1 </xsl:text>
<!-- Generate line .TIM <time> -->
<xsl:apply-templates select="design/date"/>
<!-- Generate line eeschema version ... -->
<xsl:apply-templates select="design/tool"/>
<xsl:text>}&nl;</xsl:text>

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

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

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

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

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

```

```

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

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

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

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

  <!-- search net name in nets section and write it: -->
  <xsl:variable name="pinNum" select="@num" />
  <xsl:for-each select="/export/nets/net">

```

```

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

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

<!--
This template writes the pin netname of a given pin of a given component
from the nets list
The nets list description is something like
<nets>
  <net code="1" name="GND">
    <node ref="J1" pin="20"/>
    <node ref="C2" pin="2"/>
  </net>
  <net code="2" name="">
    <node ref="U2" pin="11"/>
  </net>
</nets>
This template has 2 parameters:
"cmp_ref" (schematic reference of the given component)
"pin_num" (pin number)
-->
<xsl:template name="Search_pin_netname" match="node">
  <xsl:param name="cmp_ref" select="0" />
  <xsl:param name="pin_num" select="0" />
  <xsl:param name="pin_cnt_in_net" select="0" />

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

```

```
</xsl:if>

</xsl:template>

</xsl:stylesheet>
```

14.3.3.2 - The output file:

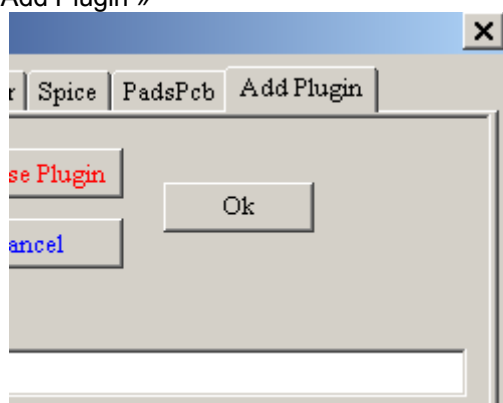
```
( { EESchema Netlist Version 1.1 29/08/2010 21:07:51
eeschema (2010-08-28 BZR 2458)-unstable}
( 4C6E2141 $noname P1 CONN_4
( 1 VCC )
( 2 /SIG_OUT )
( 3 /CLOCK_IN )
( 4 GND )
)
( 4C6E20BA $noname U2 74LS74
( 1 VCC )
( 2 /SIG_OUT )
( 3 N-04 )
( 4 VCC )
( 5 /SIG_OUT )
( 6 ? )
( 7 GND )
( 14 VCC )
)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

14.3.4 - Using Eeschema plugins interface:

Intermediate Netlist converters can be automatically launched by Eeschema

14.3.4.1 - Init the Dialog window:

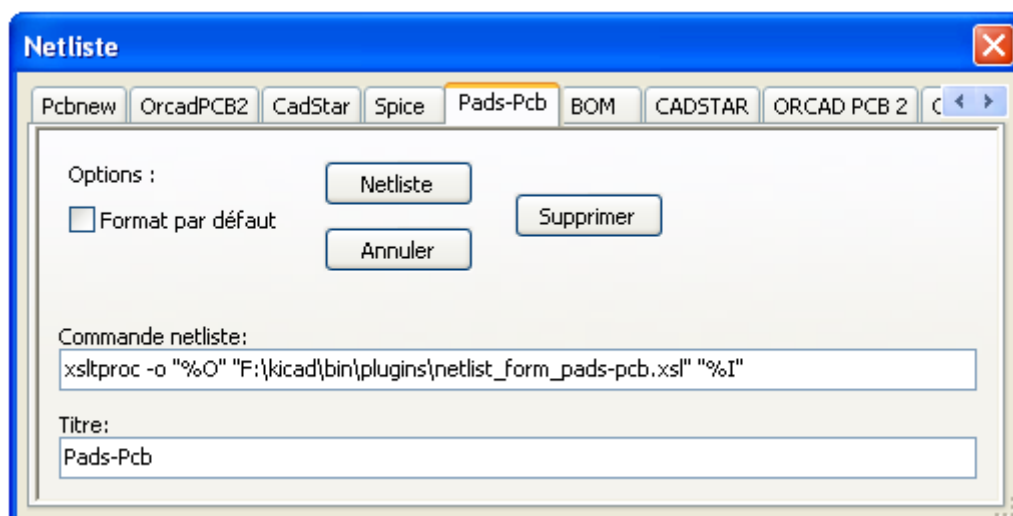
One can add a new netlist plug-in by « Add Plugin »



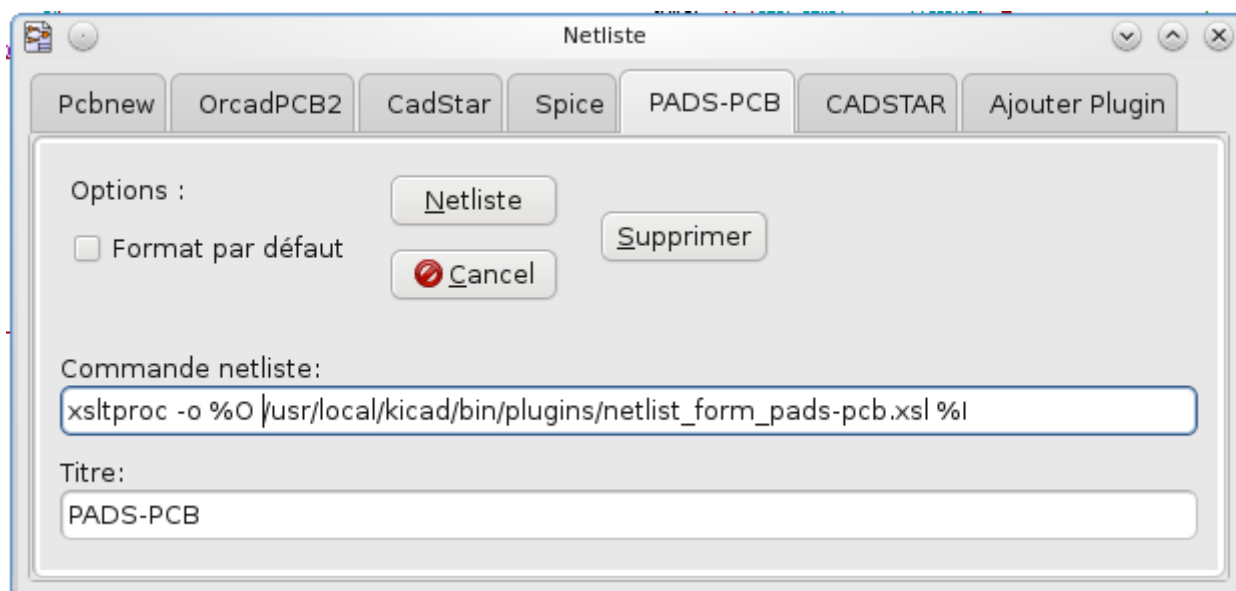
Eeschema

Here is the plug-in « PadsPcb » setup:

Windows:

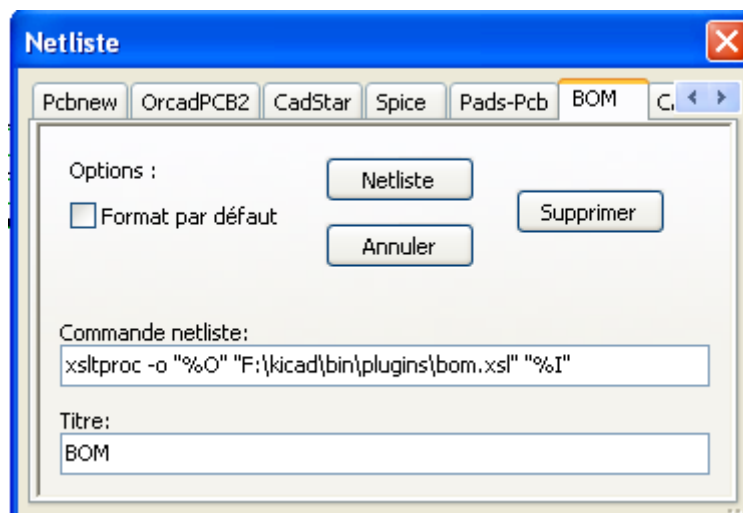


Linux:



Because this intermediate file contains most of info about components, a BOM can be extracted from it this file. Here is the plug-in setup to create a customized Bill Of Material (BOM):

Windows:



14.3.4.2 - Parameters:

The setup needs:

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

On activate the Netlist button:

1. Eeschema creates an intermediate netlist file *.xml, for instance *test.xml*
2. Eeschema runs the plug-in, what read *test.xml* and creates *test.net*

14.3.4.3 - The command line:

Assuming we are using *xsltproc.exe* to apply the sheet style to the intermediate file:

xsltproc.exe needs a command like:

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

So, in Kicad the command line is (using Windows):

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

or (using Linux)

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

netlist_form_pads-pcb.xml is the style-sheet.

Do not forget to quote file names if they have spaces in name.

14.3.4.4 - Command line format:

<path of *xsltproc* >*xsltproc* < *xsltproc* parameters>

For parameters, supported formatting sequences are:

- %B => base filename and path of selected output file, minus path and extension.
- %I => complete filename and path of the temporary input 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 (the final netlist file)

And a command line could be

- under Windows:
f:/kicad/bin/xsltproc.exe -o %O f:/kicad/bin/plugins/netlist_form_pads-pcb.xml %I
- under Linux)
xsltproc -o %O /usr/local/kicad/bin/plugins/netlist_form_pads-pcb.xml %I

Assuming ***xsltproc*** is installed on your PC b(under Windows, all files in *kicad/bin*).

14.4 - The Intermediate Netlist structure:

This sample gives an idea of the file.

```
<?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>
  </components>
</export>
```



```

</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/>
<libraries/>
<nets>
  <net code="1" name="GND">
    <node ref="U1" pin="7"/>
    <node ref="C1" pin="2"/>
    <node ref="U2" pin="7"/>
    <node ref="P1" pin="4"/>
  </net>
  <net code="2" name="VCC">
    <node ref="R1" pin="1"/>
    <node ref="U1" pin="14"/>
    <node ref="U2" pin="4"/>
    <node ref="U2" pin="1"/>
    <node ref="U2" pin="14"/>
    <node ref="P1" pin="1"/>
  </net>
  <net code="3" name="">
    <node ref="U2" pin="6"/>
  </net>
  <net code="4" name="">
    <node ref="U1" pin="2"/>
    <node ref="U2" pin="3"/>
  </net>
  <net code="5" name="/SIG_OUT">
    <node ref="P1" pin="2"/>
    <node ref="U2" pin="5"/>
    <node ref="U2" pin="2"/>
  </net>
  <net code="6" name="/CLOCK_IN">
    <node ref="R1" pin="2"/>
    <node ref="C1" pin="1"/>
    <node ref="U1" pin="1"/>
    <node ref="P1" pin="3"/>
  </net>
</nets>
</export>

```

14.4.1 - General structure:

The intermediate Netlist has 5 sections:

- the header section.
- The component section.
- The lib parts section.
- The libraries section.
- The nets section.

The whole file has the delimiter <export>

```

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

```

14.4.2 - 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 seen as comments.

14.4.3 - 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 is the list of components in 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.

14.4.3.1 - Note about time stamps for components:

To identify a component in a netlist (and therefore on a board), the reference is used, and is unique for each component. However Kicad provides an auxiliary way to identify a component, and its corresponding footprint on the board. This allows the re-annotation of components in a schematic project and do 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

14.4.4 - The libparts section:

The libparts section has the delimiter <libparts>, and the data in this section is defined in the schematic libraries. Its contains for each component:

- The allowed footprints names (names use jokers) 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.
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 collector sometimes found in logic.
Not connected	Must be left open in schematic

14.4.5 - The libraries section:

The libraries section has the delimiter <libraries>
Gives the list of schematic libraries used in the schematic 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>
```

14.4.6 - The nets section:

The nets section has the delimiter <nets>.
This is 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"/>
  </net>
```

```

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

```

This sections lists all nets in the schematic.

A given net is :

```

<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

14.5 - More info about xsltproc:

See <http://xmlsoft.org/XSLT/xsltproc.html>

14.5.1 - Introduction

xsltproc is a command line tool for applying XSLT stylesheets to XML documents.

It is part of libxslt, the XSLT C library for GNOME.

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 stylesheet to be used followed by the name of the file or files to which the stylesheet is to be applied. It will use the standard input if a filename provided is - .

If a stylesheet is included in an XML document with a Stylesheet Processing Instruction, no stylesheet need be named at the command line. xsltproc will automatically detect the included stylesheet and use it.

By default, output is to stdout. You can specify a file for output using the -o option.

14.5.2 - 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] [...]

```

14.5.3 - 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

Eeschema

- 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 parameter 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 or --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 on 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.
- noddattr
Do not apply default attributes from the document's DTD.

14.5.4 - Return values

xsltproc's return codes provide information that can be used when calling it from scripts.

- 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

14.5.5 - More Information

libxml web page: <http://www.xmlsoft.org/>
W3C XSLT page: <http://www.w3.org/TR/xslt>