Eeschema

October 31, 2021

Contents

リファレンス・マニュアル

貢献者

Jean-Pierre Charras, Fabrizio Tappero.

翻訳

starfort <starfort AT nifty.com>, 2015. Norio Suzuki <nosuzuki AT postcard.st>, 2015. yoneken <yoneken AT kicad.jp>, 2011-2015. Silvermoon, Zenyouji, Millo, Nenokuni 2011-2012.

フィードバック

バグ報告や提案はこちらへお知らせください:

- KiCad のドキュメントについて: https://github.com/KiCad/kicad-doc/issues
- KiCad ソフトウェアについて: https://bugs.launchpad.net/kicad
- KiCad ソフトウェアの国際化について: https://github.com/KiCad/kicad-i18n/issues

発行日とソフトウエアのバージョン

2015 年 5 月 30 日発行

Chapter 1

Eeschema 入門

1.1 説明

Eeschema は、KiCad の一部として配布されている強力な回路図エディタソフトウェアであり、次のオペレーティングシステムの下で利用可能です。:

- Linux
- Apple OS X
- Windows

OS に関係なく、すべての Eeschema ファイルは一方の OS から他方のものへ 100 %互換性があります。

Eeschema は、図面、コントロール、レイアウト、ライブラリ管理および、PCB 設計ソフトウェアへのアクセスといったすべての機能が Eeschema 内で実行される統合ソフトウェアです。

Eeschema は、プリント基板設計ソフトウェア PcbNew とともに使用されることを想定しています。また、PCB の電気的接続を記述するネットリストファイルをエクスポートすることも可能です。

Eeschema は、コンポーネントの作成／編集やライブラリの管理ができるコンポーネント・ライブラリ・エディタを含んでいます。また、現代の回路図エディタソフトに必要不可欠な機能だけでなく、以下の追加機能も組み込まれています。:

- 誤接続、未接続の自動的な検出を行うエレクトリカル・ルール・チェック (ERC)
- 多くの形式 (Postscript, PDF, HPGL, SVG) をサポートしたプロットファイルのエクスポート。
- (様々なフォーマットを設定できるよう Python スクリプトを使用した）部品表 (BOM) の生成。

1.2 技術的な概要

Eeschema は、コンピュータで利用可能なメモリの大きさにのみ制限を受けます。コンポーネント、ピン、接続、シート、の数に関して明示的な制限はありません。複数シートからなる回路図では、階層的な表示となります。

Eeschema は、次のタイプの複数のシートからなる回路図を扱うことができます:

- 単一の階層 (各図が一度だけ使用される)。
- 複雑な階層 (いくつかの図は、一度以上の複数回使用される)。
- 平面的（フラット）な階層 (マスター図面の中、いくつかの図面は明示的に接続されない)。

Chapter 2

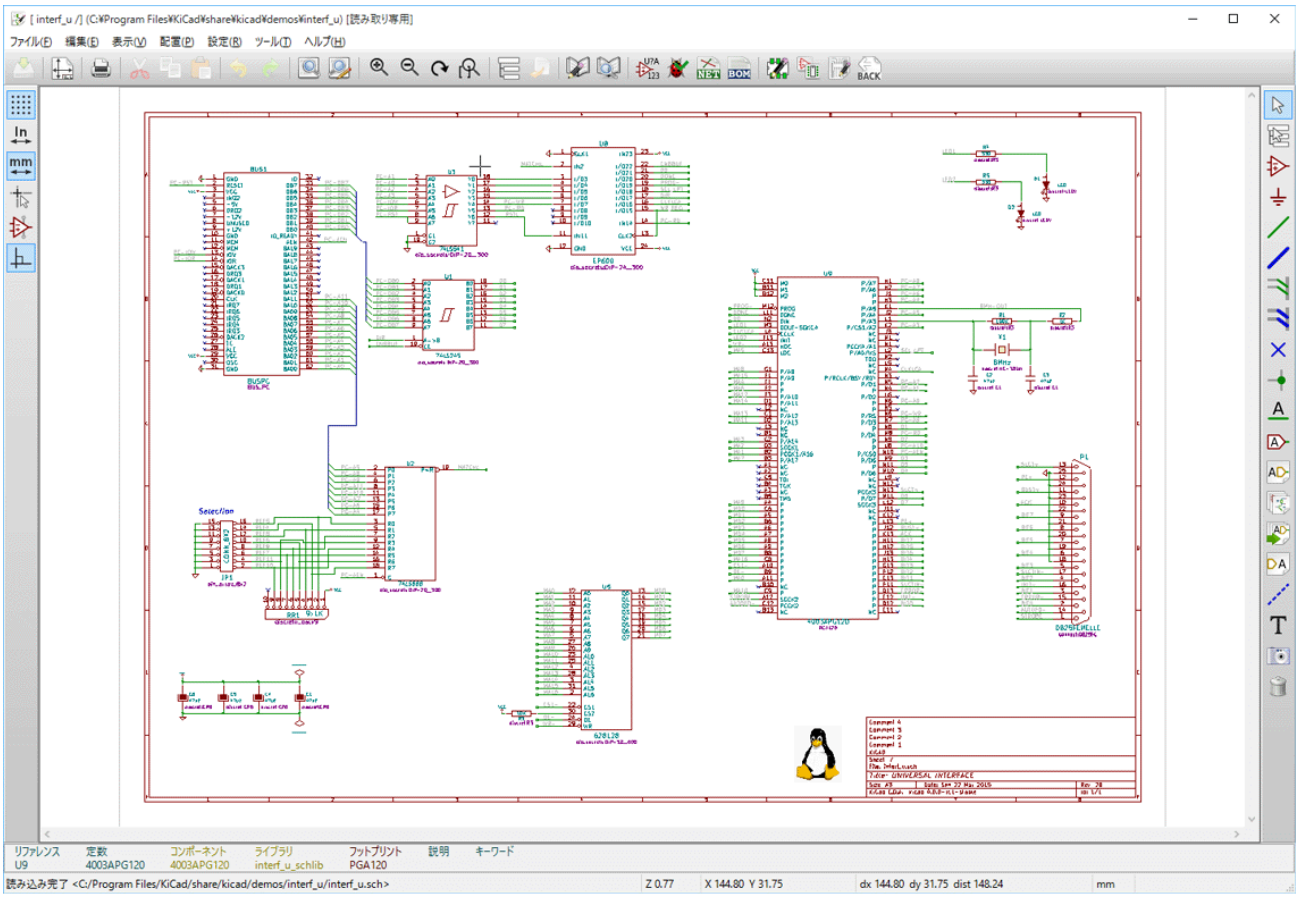
Eeschema コマンド全般

2.1 Eeschema コマンドへのアクセス

以下に示す様々な方法でコマンドを起動できます：

- 画面上部のメニューバーをクリックする。
- 画面上部のアイコンをクリックする（一般コマンド)。
- 画面右側のアイコンをクリックする（特殊コマンド、または“ツール”)。
- 画面左側のアイコンをクリックする（表示オプション)。
- マウスボタンをクリックする（重要な補助コマンド)。特に右クリックでは、カーソル下の要素に対応したコンテキストメニューを開きます（ズーム、グリッドと要素の編集)。
- キーボードのファンクションキー（F1，F2，F3，F4，インサートとスペースキー)。特に“ESC”キーは，多くの進行中のコマンドを中断できます。“Insert”キーは、最後に作成された要素を複写します。

下図に様々なコマンドの配置を示します。:



2.2 マウスコマンド

2.2.1 基本コマンド

左ボタン

- シングルクリック: カーソルに下にあるコンポーネントあるいはテキストの属性をステータスバーへ表示する。
- ダブルクリック: コンポーネントあるいはテキストを編集する。(要素が編集可能な場合)

右ボタン

- コンテキストメニューを開く。

2.2.2 ブロックの操作

Eeschema では、選択範囲を移動、ドラッグ、コピー、削除することができます。

マウスの左ボタンを押しながらドラッグして範囲を選択し、左ボタンを離すと範囲の選択ができます。

“Shift” と “Ctrl” キーのうちどちらか 1 つ、もしくは “Shift” と “Ctrl” キーの両方を押しながら選択することで、選択した範囲をコピーするか、ドラッグするか、削除するか、が変わります。

左マウスボタン	選択範囲を移動。
Shift + 左マウスボタン	選択範囲をコピー。
Ctrl + 左マウスボタン	選択範囲をドラッグ。
Ctrl + Shift + 左マウスボタン	選択範囲を削除。

ドラッグまたはコピー中、次のことができます:

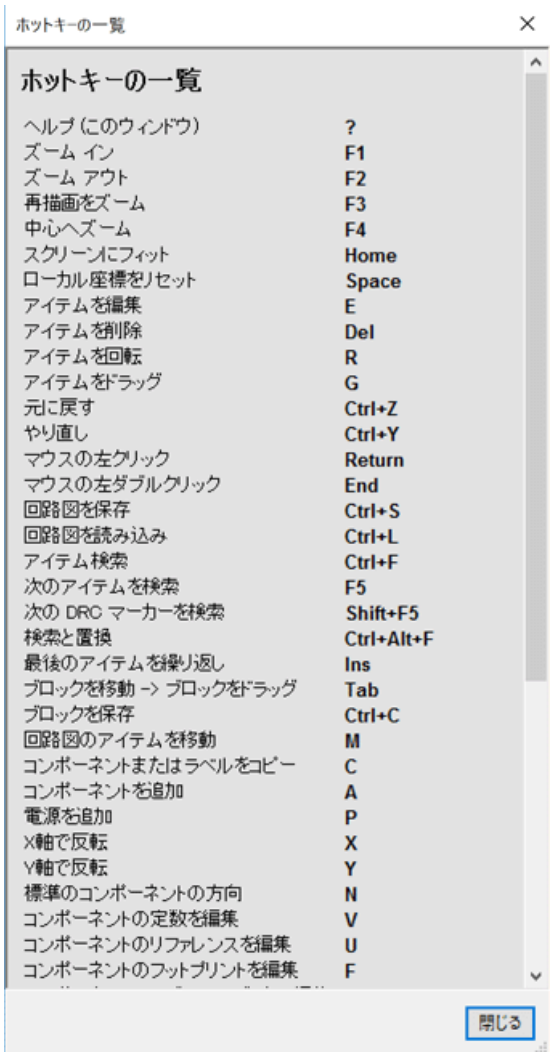
- もう一度クリックして要素を置き直す。
- 右ボタンをクリックして取消す。

ブロック移動コマンドが実行されているとき、マウスの右ボタンからコンテキストメニューを開くと、他のブロックコマンドも選択できます:

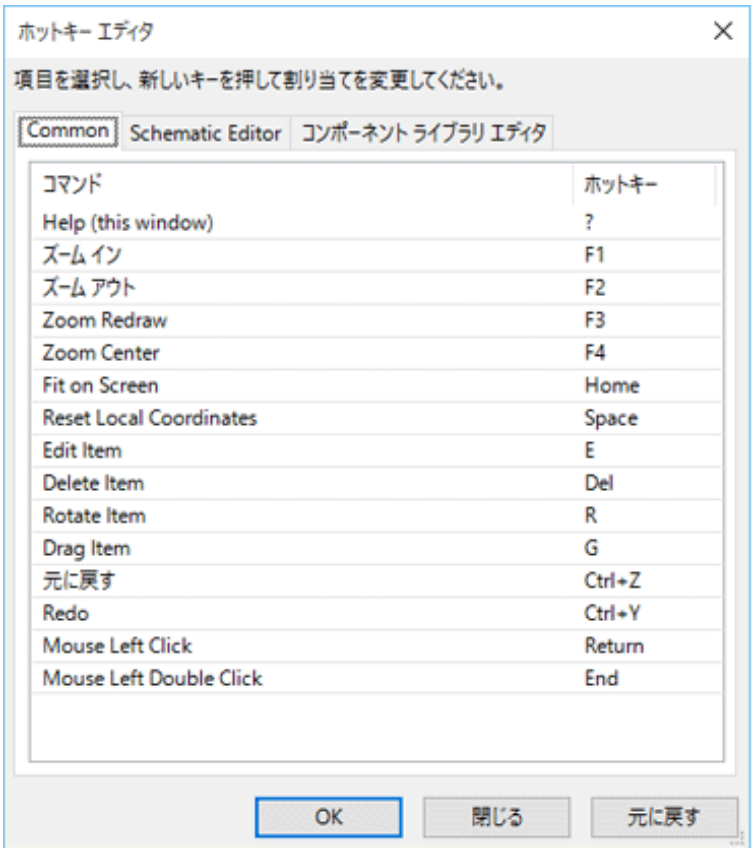
2.3 ホットキー

- “?” で現在のホットキーのリストを表示します。
- ホットキーは、上部メニューバーの” 設定” から” ホットキーの編集” を選択すると変更できます。

以下はデフォルトのホットキーのリストです:



全てのホットキーは、ホットキーエディタでユーザが再定義できます。



2.4 グリッドサイズの選択

Eeschema では、表示／非表示が可能なグリッド上でカーソルが動きます。コンポーネント・ライブラリ・エディタでもグリッドは表示されます。

コンテキストメニューあるいは「設定／オプション」メニューから、グリッドサイズを変えることができます。

デフォルトのグリッドサイズは、50 mil (0.050″) あるいは 1,27 mm です。

これは回路図上で配線や部品を配置したり、コンポーネント・ライブラリ・エディタ上で回路記号のピンを配置するのに適したサイズです。

25 mil から 10 mil のより細かいグリッドでも作業できます。これは、ピンの配置や配線ではなく、コンポーネント本体のデザインあるいはテキストやコメントを配置する時に使うことを意図しています。

2.5 ズームの選択

ズームレベルを変えるには:

- 右クリックしてコンテキストメニューを開き、希望のズームを選択。
- あるいはファンクションキーを使って:
  - F1: Zoom in

  - F2: Zoom out
  - F4 or simply click on the middle mouse button (without moving the mouse): Center the view around the cursor pointer position

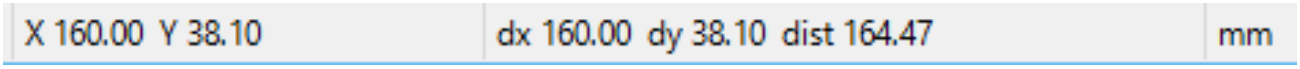
- ウィンドウのズーム:
  - Mouse wheel: Zoom in/out
  - Shift+Mouse wheel: Pan up/down
  - Ctrl+Mouse wheel: Pan left/right

2.6 カーソルの座標表示

表示単位は inch あるいは mm です。しかしながら、Eeschema は内部的には常に 1/1000 inch で扱っています。
ウィンドウの下部右側には以下の情報が表示されます:

- ズーム倍率
- カーソルの絶対位置
- カーソルの相対位置

相対座標値 (x, y) はスペースキーで (0, 0) へリセットされます。これは２点間の距離を測る時に役立ちます。



2.7 上部メニューバー

上部メニューバーでは、回路図やプログラム設定を開いたり、保存したり、ヘルプメニューを開いたりできます。

2.8 上部ツールバー

このツールバーから、Eeschema の主な機能へアクセスできます。
Eeschema が単独で実行されている場合、以下のツールが有効です。:

Eeschema がプロジェクトマネージャー (KiCad) から実行されている場合、以下のツールが有効です。:

プロジェクトを初期化するツールは無効です。このツールは プロジェクトマネージャーにあります。

	Create a new schematic (only in standalone mode).
	Open a schematic (only in standalone mode).
	Save complete (hierarchical) schematic.
	Select the sheet size and edit the title block.
	Open print dialog.
	Remove the selected elements during a block move.
	Copy selected elements to the clipboard during a block move.
	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 dialog to search components and texts in the schematic.
	Call the dialog to search and replace texts in the schematic.
	Zoom in and out.
	Refresh screen; zoom to fit.
	View and navigate the hierarchy tree.
	Leave the current sheet and go up in the hierarchy.
	Call component editor *Libedit* to view and modify libraries and component symbols.
	Display libraries (Viewlib).
	Annotate components.
	Electrical rules check (ERC), automatically validate electrical connections.

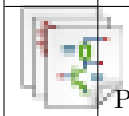
	Export a netlist (Pcbnew, SPICE, and other formats).
	Generate the BOM (Bill of Materials).
	Edit footprint.
	Call CvPcb to assign footprints to components.
	Call Pcbnew to perform a PCB layout.
	Back-import component footprints (selected using CvPcb) into the ”footprint” fields.

2.9 右ツールバー

このツールバーは次のツールを含んでいます:

- コンポーネント、ワイヤ、バス、ジャンクション、ラベル、テキストの配置
- 階層サブシートと接続シンボルの作成

	Cancel the active command or tool.
	Hierarchy navigation: 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 schematic).
	Display the component selector.
	Display the power symbol selector.
	Draw a wire.
	Draw a bus.
	Draw wire-to-bus entry points. These elements are only graphical and do not create a connection, thus they should not be used to connect wires together.
	Draw bus-to-bus entry points.
	Place a ”No Connect” flag. These are 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.
	Place a junction. This connects 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).
	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 or hierarchical labels.

	Place a global label. All global labels with the same name are connected, even between different sheets.
	Place a hierarchical label. This makes it possible to place a connection between a sheet and the parent sheet that contains it.
	Place a hierarchical subsheet. You must specify the file name for this subsheet.
	Import hierarchical labels from a subsheet. These hierarchical labels must already be placed in the subsheet. These are equivalent to pins on a component, and must be connected using wires.
	Place hierarchical label in a subsheet symbol. This is placed by name and does not require the label to already exist in the subsheet itself.
	Draw a line. These are only graphical and do not connect anything.
	Place textual comments. These are only graphical.
	Place a bitmap image.
	Delete selected element. If several superimposed elements are selected, the priority is given to the smallest (in the decreasing priorities: junction, ”No Connect”, 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 左ツールバー

このツールバーは表示オプションを管理します:

	Show/Hide the grid.
	Switch to inches.
	Switch to millimeters.
	Choose the cursor shape.

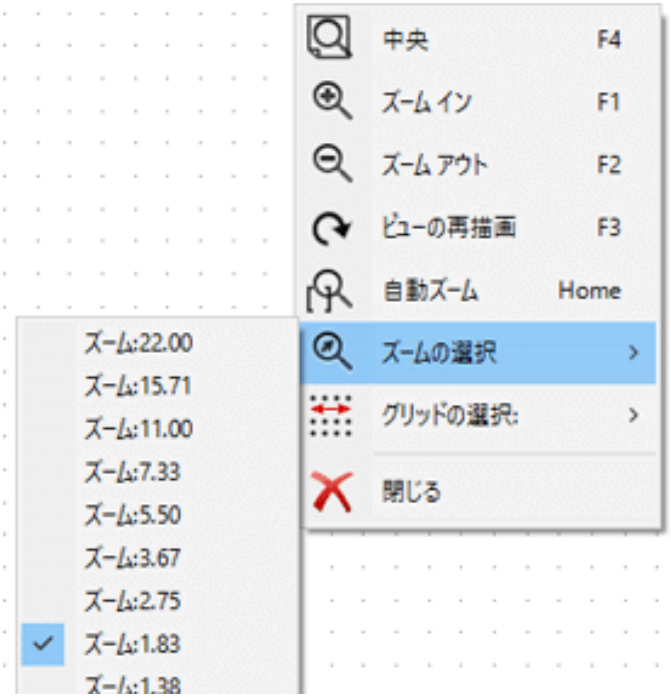
	Visibility of ”invisible” pins.
	Allowed orientation of wires and buses.

2.11 コンテキストメニューとクイックエディット

右クリックで選択要素に応じたコンテキストメニューを開き、下記の機能にアクセスします:

- ズーム倍率。
- グリッド調整。
- パラメータ編集。

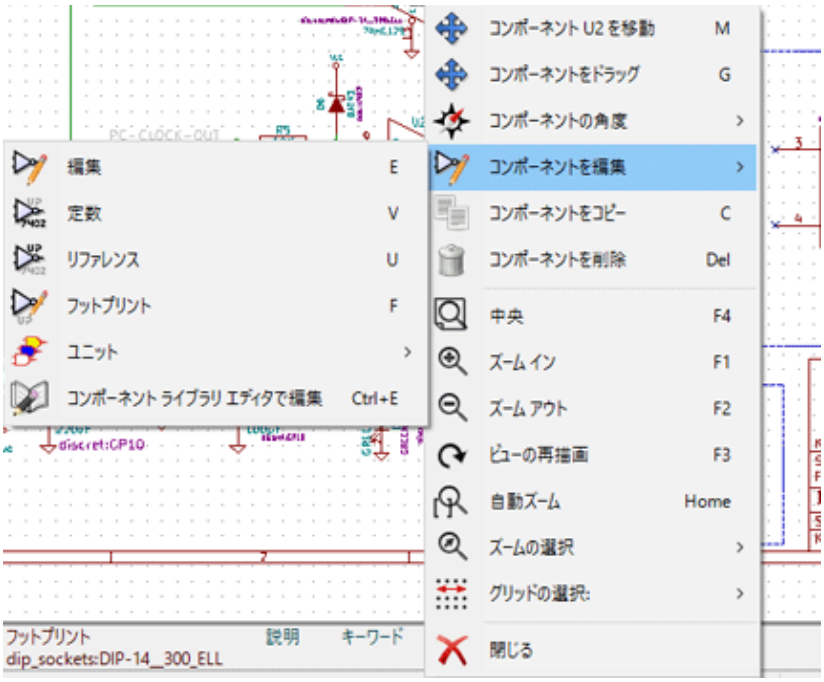
要素を選択しないでコンテキストメニューを呼び出す。



ラベルの編集。


ラベルを移動 M
ラベルをドラッグ G
ラベルをコピー C
ラベルを回転 R
ラベルを編集 E
ラベルを削除 Del
タイプの変更
中央 F4
ズーム イン F1
ズーム アウト F2
ビューの再描画 F3
自動ズーム Home
ズームの選択
グリッドの選択:
閉じる
階層ラベルに変更
テキストに変更
グローバル ラベルに変更

コンポーネントの編集。


コンポーネント U2 を移動 M
コンポーネントをドラッグ G
コンポーネントの角度
コンポーネントを編集
コンポーネントをコピー C
コンポーネントを削除 Del
中央 F4
ズーム イン F1
ズーム アウト F2
ビューの再描画 F3
自動ズーム Home
ズームの選択
グリッドの選択:
閉じる
編集 E
定数 V
リファレンス U
フットプリント F
ユニット
コンポーネント ライブラリ エディタで編集 Ctrl+E
フットプリント
dip_sockets:DIP-14_300_ELL
説明
キーワード

Chapter 3

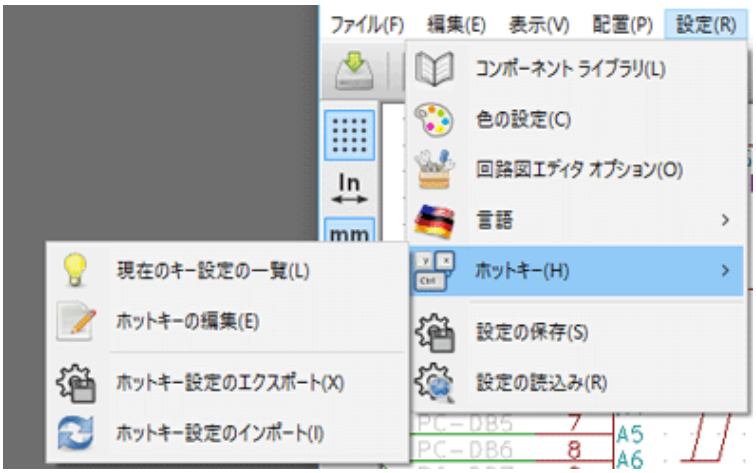
上部メニューバーの主なメニュー

3.1 ファイルメニュー

New Schematic Project	Clear current schematic and initialize a new one
Open Schematic Project	Load a schematic hierarchy
Open Recent	Open a list of recently opened files
Append Schematic Sheet	Insert the contents of another sheet into the current one
Save 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.
Page Settings	Configure page dimensions and title block.
Print	Print schematic hierarchy (See also chapter Plot and Print).
Plot	Export to PDF, PostScript, HPGL or SVG format (See chapter Plot and Print).
Close	Quit without saving.

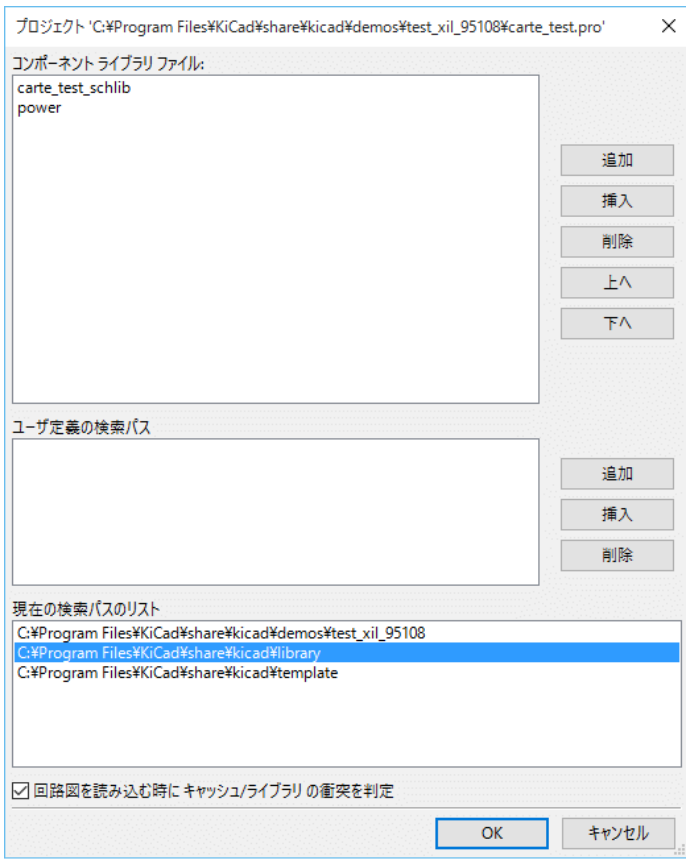
3.2 設定メニュー

3.2.1 設定



コンポーネントライブラリ	ライブラリとライブラリパスを選択。
色の設定	印刷、プロットの色を選択。
回路図エディタオプション	一般的なオプション (単位、グリッドサイズ、ファイル名等)。
言語	インターフェイスの言語を選択。
ホットキー	リスト、編集、エクスポート、インポートのホットキー設定。
設定の保存	.pro ファイルへプロジェクトの設定を保存。
設定の読込み	.pro ファイルからプロジェクトの設定を読込み。

3.2.2 設定メニュー / コンポーネントライブラリ



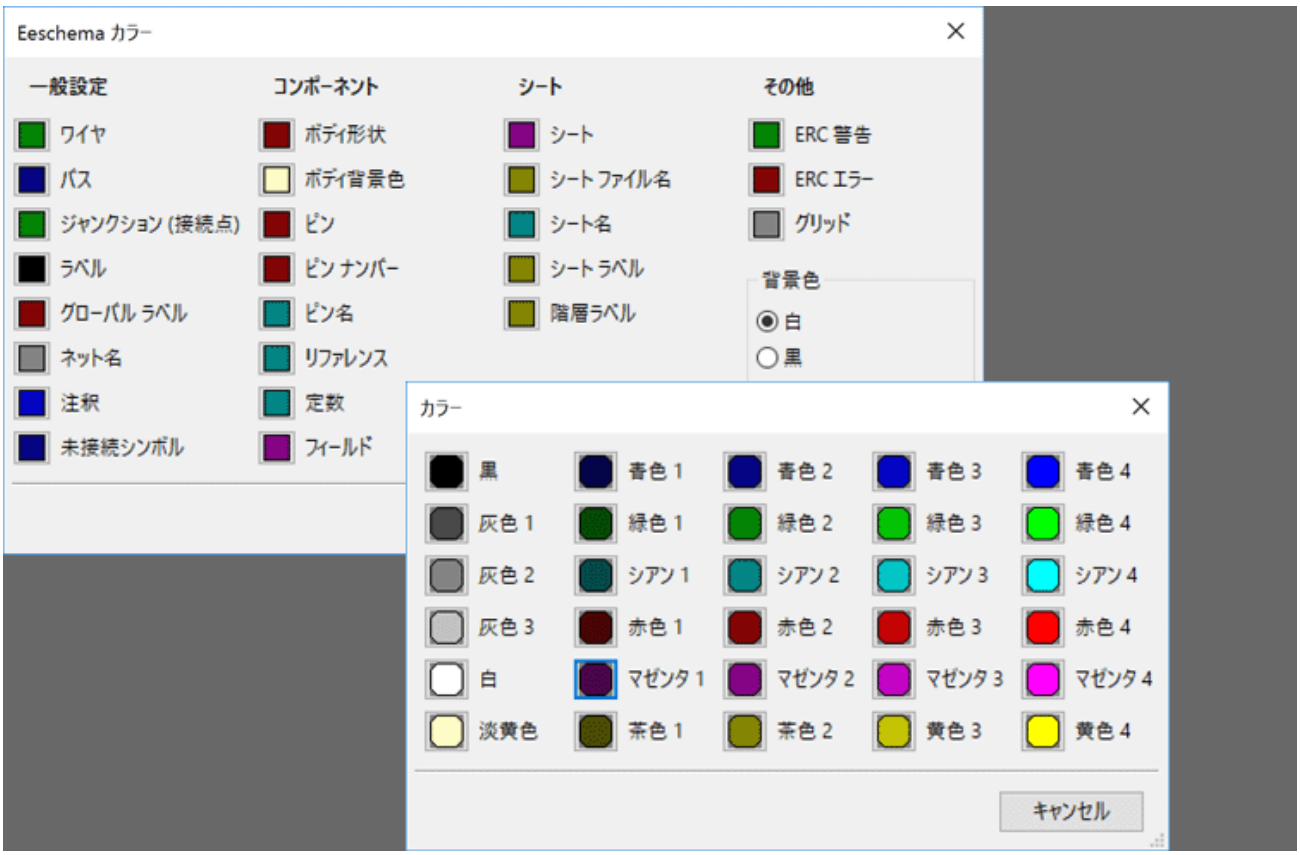
このダイアログは、コンポーネントライブラリと検索パスの設定に使用されます。設定パラメータは.pro ファイルに保存されます。異なるディレクトリでは、異なる設定ファイルを使用することも可能です。

Eeschema は次の優先度順に設定ファイルを検索します。:

1. 現在のディレクトリ内の設定ファイル (projectname.pro) 。
2. KiCad ディレクトリ内の kicad.pro という設定ファイル。このファイルはデフォルト設定として使用されます.
3. ファイルが見つからない場合のデフォルト値。少なくともロードするためのライブラリのリストを記入し、設定を保存する必要があります。

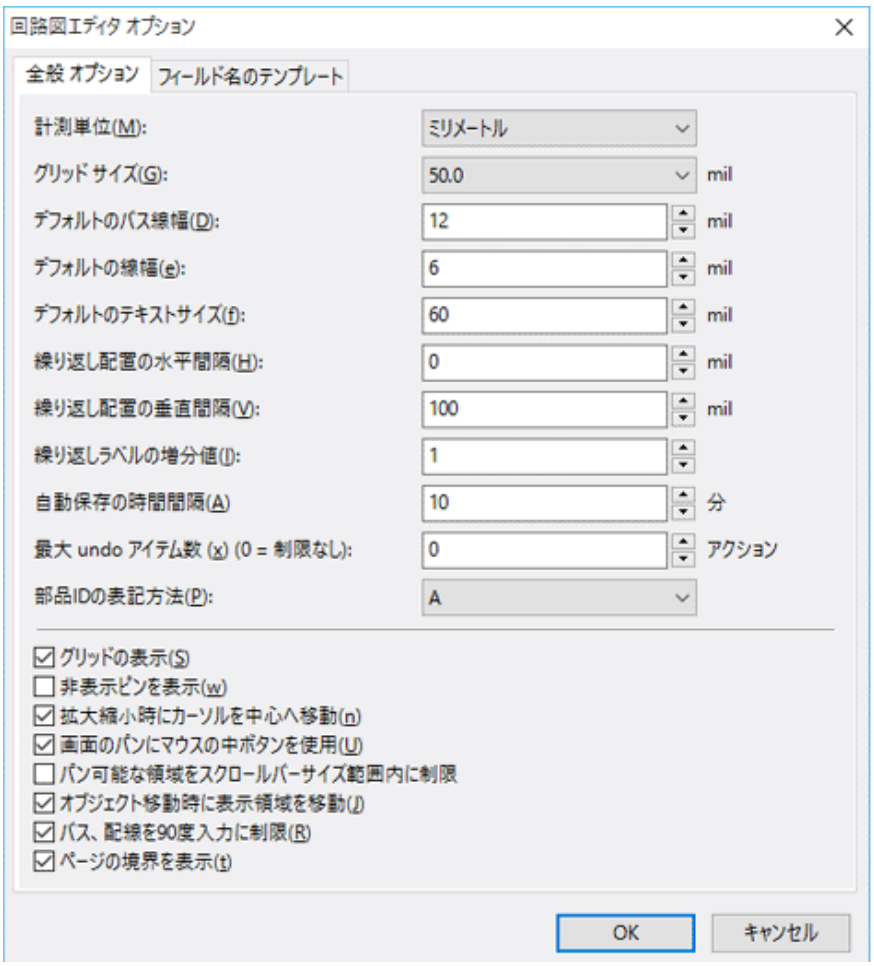
“回路図の読込み時にキャッシュ／ライブラリの衝突をチェック” ボックスはライブラリの衝突を救済する設定用に使われます。詳細は、キャッシュされたコンポーネントのレスキュー を参照のこと。

3.2.3 設定メニュー / 色の設定



様々な要素の描画色と背景色（黒または白のみ）の設定。

3.2.4 設定メニュー / 回路図エディタオプション



Measurement units:	Select the display and the cursor coordinate units (inches or millimeters).
Grid Size:	Grid size selection. **It is recommended to work with normal grid (0.050 inches or 1,27 mm)**. *Smaller grids are used for component building.*
Default bus width:	Pen size used to draw buses.
Default line width:	Pen size used to draw objects that do not have a specified pen size.
Default text size:	Text size used when creating new text items or labels
Repeat draw item horizontal displacement	increment 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 *Insert* key)
Repeat draw item vertical displacement	increment on Y axis during element duplication (usual value is 0.100 inches or 2,54 mm)
Repeat label increment:	Increment of label value during duplication of texts ending in a number, such as bus members (usual value 1 or -1).

Auto save time interval:	Time in minutes between saving backups.
Part id notation:	Style of suffix that is used to denote component parts (U1A, U1.A, U1-1, etc.)
Show Grid:	If checked: display grid.
Show hidden pins:	Display invisible (or *hidden*) pins, typically power pins. If checked, allows the display of power pins.
Do not center and warp cursor on zoom:	When zooming, keep the position and cursor where they are.
Use middle mouse button to pan	When enabled, the sheet can be dragged around using the middle mouse button.
Limit panning to scroll size	When enabled, the middle mouse button cannot move the sheet area outside the displayed area.
Pan while moving object	If checked, automatically shifts the window if the cursor leaves the window during drawing or moving.
Allow buses and wires to be placed in H or V orientation only	If checked, buses and wires can only be vertical or horizontal. Otherwise, buses and wires can be placed at any orientation.
Show page limits	If checked, shows the page boundaries on screen.

3.2.5 言語設定

デフォルトのモードで使用してください。他言語は主に開発目的で利用されます。

3.3 ヘルプメニュー

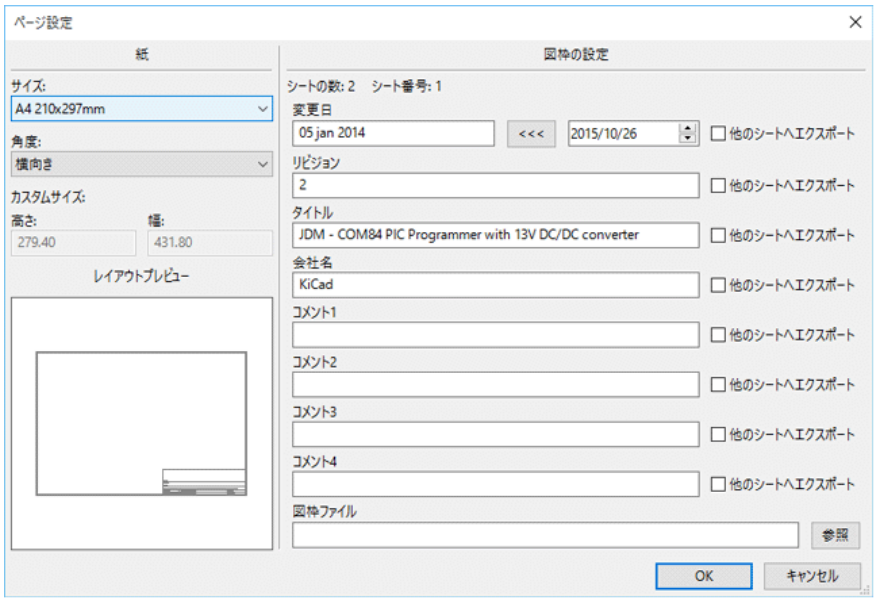
KiCad についての広範囲なチュートリアルについては、オンラインヘルプ（この文書）にアクセスします。ビルドと環境を明示してバグレポートを送る時は、“バージョン情報をコピー”を使って下さい。

Chapter 4

上部ツールバーの主なツール

4.1 ページ設定

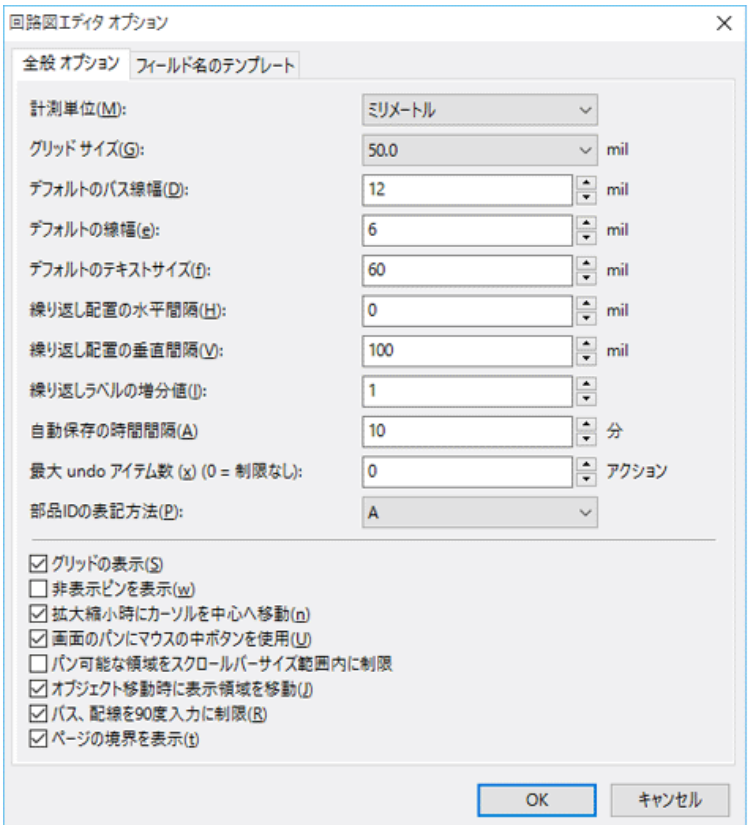
 アイコンにより、ページ設定にアクセスできます。ここでは用紙サイズと右下隅にある表題欄のテキストセクションを定義することができます。



シートの数やシート番号といったデータは、自動的に更新されます。左向き矢印のボタンを押すと今日の日付を“変更日”の日付に設定することができますが、自動的には更新されません。

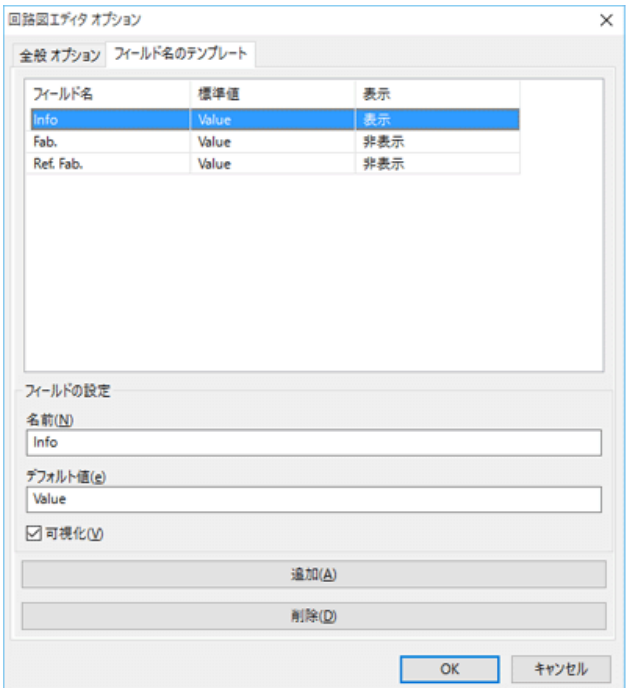
4.2 回路図エディタのオプション

4.2.1 全般オプション



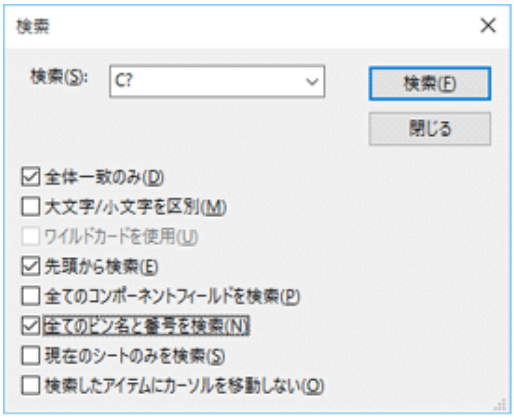
4.2.2 フィールド名のテンプレート

各々のコンポーネントでデフォルトで存在するカスタムフィールド（フィールドが空のままでも可）を定義できます。



4.3 検索ツール

 アイコンは、検索ツールへのアクセスに使われます。



現在のシートあるいは階層全体内にある、リファレンス、値、テキスト文字列を検索することができます。見つかると、関係するサブシートの見つかった要素の上にカーソルが移動します。

4.4 ネットリストツール

 アイコンで、ネットリストファイルを生成するツールを呼び出します。

ネットリストファイルは、階層全体（通常のオプション）の全ての接続の記述を作成します。

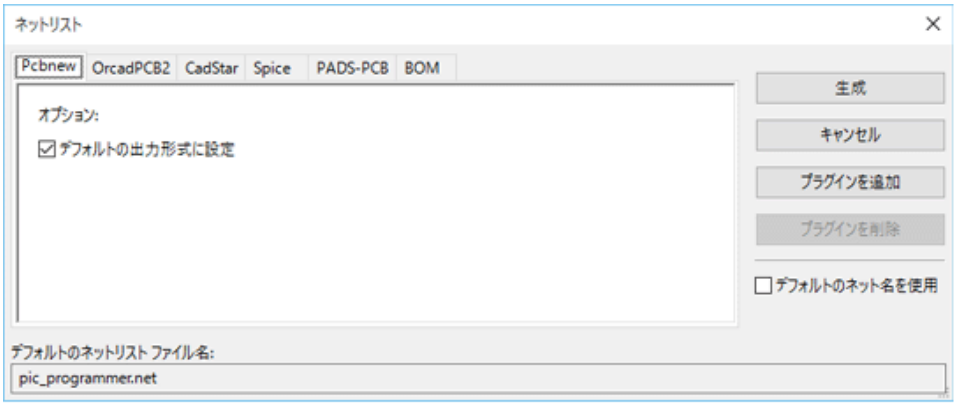
マルチシートの階層において、ローカルラベルは自身が属するシート内だけで通用します。したがって、シート３のラベル TOTO は、シート５のラベル TOTO とは接続されません（意図的にそれらを接続していない場合)。これは、シート番号（アノテーションコマンドによって更新される）がローカルラベルと関連付けられているからです。

注１：

Eeschema にはラベルの長さに制限はありませんが、生成されたネットリストを利用するソフトウェアには制限のある場合があります。

注２：

区切られた単語として表示されてしまうため、ラベルの中では空白文字（半角スペース）を使うべきではありません。それは Eeschema の制限ではなく、多くのネットリストフォーマットにおいて、ラベルは空白文字を含んでいないものと定義されているからです。



オプション：

デフォルトの出力形式に設定:

デフォルトのフォーマットとして Pcbnew を選択するためには、チェックを入れてください。

以下のような他のフォーマットも生成できます:

- Orcad PCB2
- CadStar
- Spice, シミュレータ用

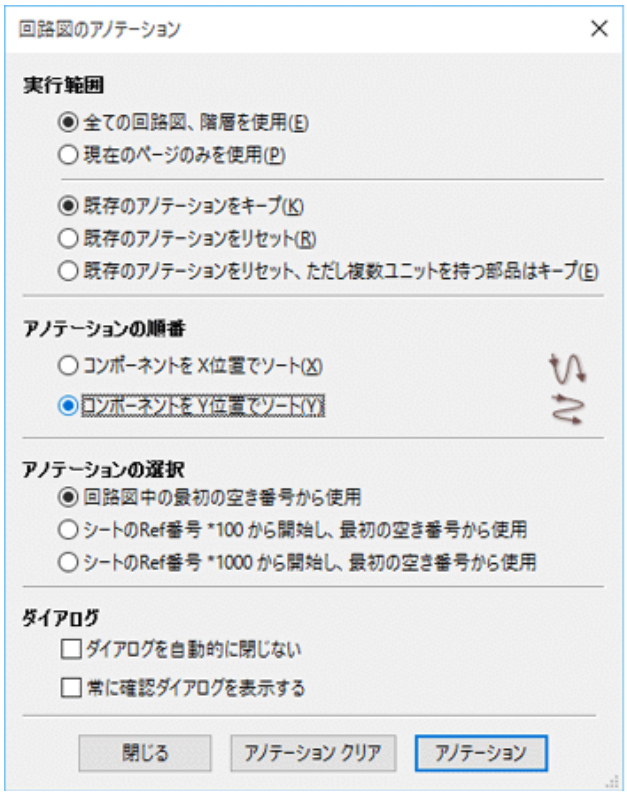
“プラグインの追加”を使うと、外部プラグインでネットリスト・フォーマット・リストを拡張できます（PadsPcb プラグインなど)。

4.5 アノテーションツール

アイコンからアノテーションツールを呼び出します。このツールは使用されている全てのコンポーネントに対して自動的にリファレンス（参照番号）の割り付けを行います。

多パーツコンポーネント（４ゲート入りの 7400 TTL など）には、マルチパート (multi-part) の接尾辞が割り当てられます。従って、U3 に指定された 7400 TTL は、U3A、U3B、U3C、U3D に分かれます。

無条件に全てのコンポーネントに、または未だアノテートされていない新規のコンポーネントだけに、アノテートできます。



実行範囲

1. 全ての回路図、階層を使用：全てのシートを再アノテート (通常の選択)
2. 現在のページでのみ使用：現在のシートのみ再アノテート（このオプションは特別な場合にのみ使用されます。例えば、現在のシートの抵抗のみを対象としたい場合など）
3. 既存のアノテーションをキープ：条件付きのアノテーション。新しいコンポーネントのみ再アノテート（通常の選択)。
4. 既存のアノテーションをリセット：無条件のアノテーション。全てのコンポーネントを再アノテートします（このオプションは重複したリファレンスがある場合に使われます)。
5. 既存のアノテーションをリセット、ただし複数ユニットを持つ部品はキープ：再アノテートの時、全ての多パーツコンポーネントのグループ (例えば、U2A, U2B) を維持します。

アノテーションの順番

コンポーネントの番号付けを行う順番を選択します。

アノテーションの選択

コンポーネント番号の選択方法を選択します。

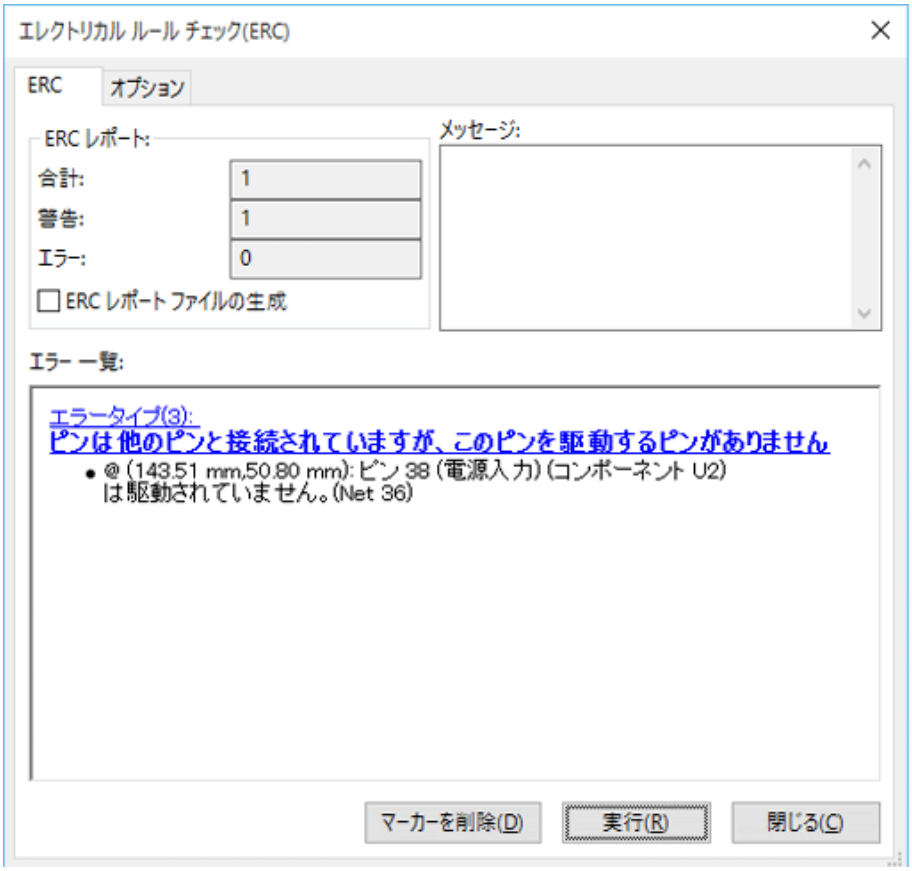
4.6 エレクトリカル・ルール・チェック (ERC) ツール

 アイコンで、エレクトリカル・ルール・チェック (ERC) ツールを呼出します。

このツールは設計の検証を行い、特に接続忘れや矛盾を検出するのに役立ちます。

一度 ERC を実行すると、Eeschema はラベルやピン上に問題を目立たせるマーカーを配置します。マーカー上で左クリックすると診断結果を表示します。エラーファイルを生成させることもできます。

4.6.1 ERC ダイアログ：ERC



エラーはエレクトリカル・ルール・チェック (ERC) ダイアログボックスに表示されます：

- 合計（エラーと警告の合計数）
- 警告（警告の発生数）
- エラー（エラーの発生数）

オプション：

- ERC レポートファイルの生成：ERC レポートファイルを生成するには、このオプションをチェックします。

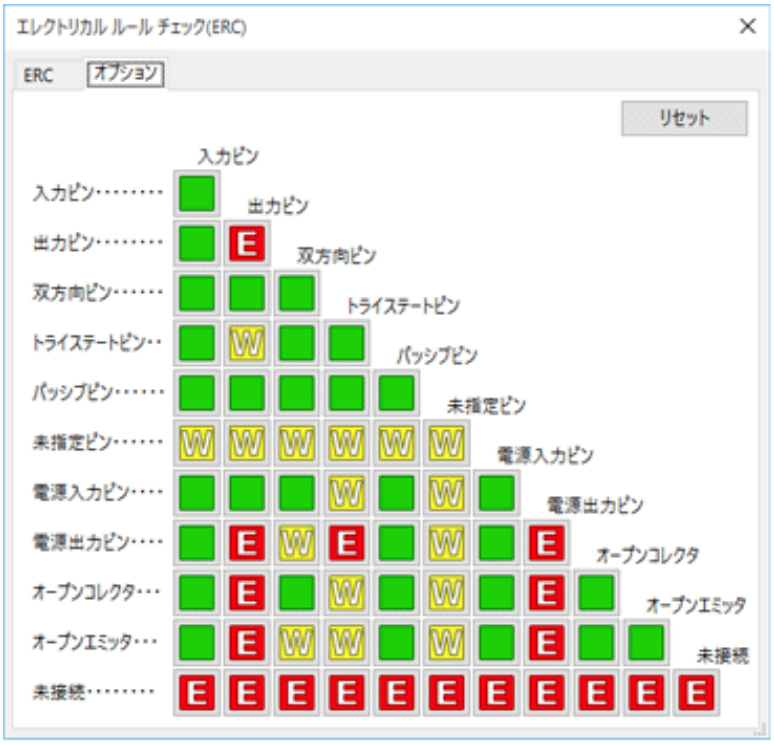
コマンド：

- マーカーの削除：全ての ERC エラー／警告マーカーを消去する
- 実行：エレクトリカル・ルール・チェックを実行する
- 閉じる：このダイアログボックスを閉じる

注：

- エラーメッセージをクリックすると、回路図の対応するマーカーにジャンプします。

4.6.2 ERC ダイアログ：オプション



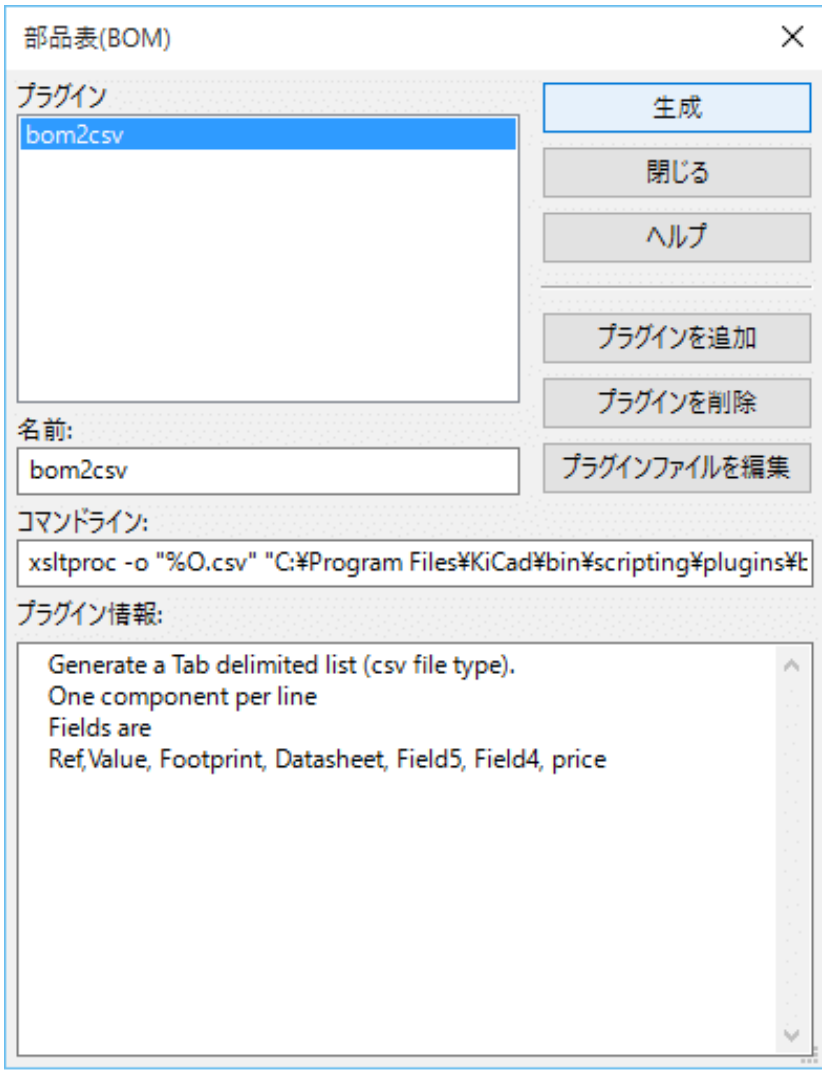
このタブページで、ピン間の接続ルールを設定できます。それぞれのケースに対して、3 つのオプションから選択できます。:

- エラーなし（No error：緑）
- 警告（Warning：黄）
- エラー（Error：赤）

マトリックスのそれぞれの四角上でクリックすることにより、内容を変更できます。

4.7 部品表 (BOM) ツール

アイコンで、部品表 (BOM) ツールを呼出します。このメニューから、コンポーネントと階層接続グローバルラベルの一覧表が生成できます。

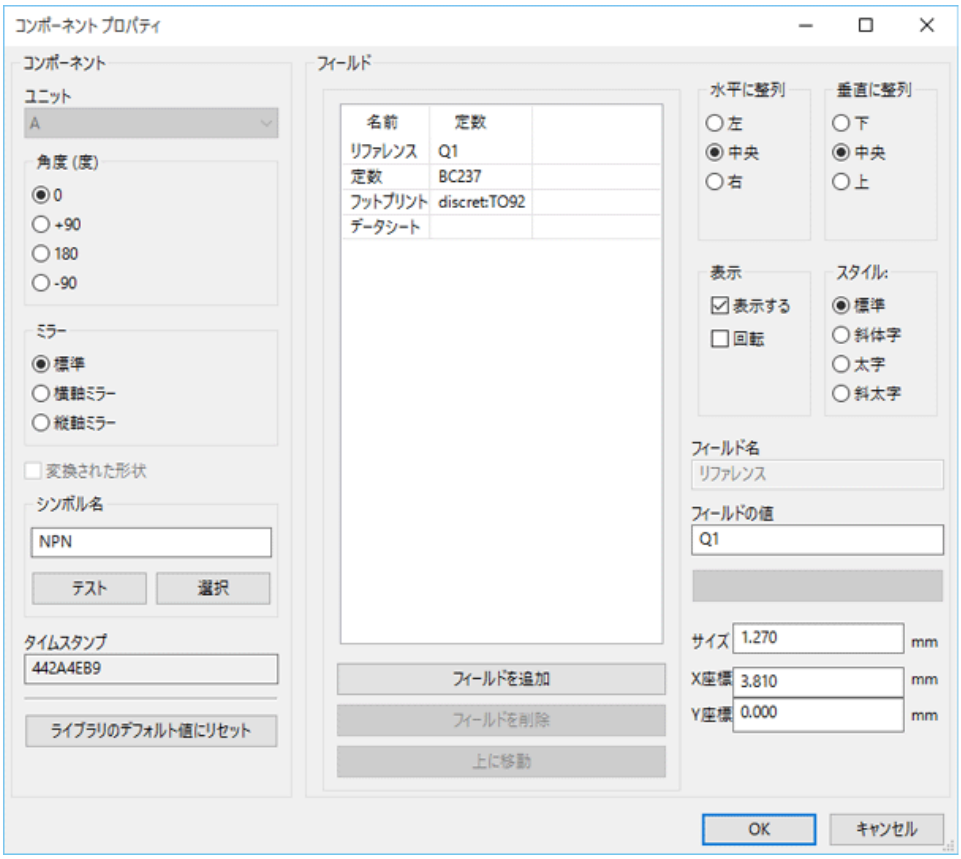


Eeschema の部品表 (BOM) ツールは外部プラグイン (通常、XSLT 又は Python) を利用します。いくつかが提供中で、KiCad プログラムのあるディレクトリへインストールされます。

部品表 (BOM) での使用に役立つコンポーネントのプロパティには、以下のものが挙げられます。:

- 定数-各部品で使用される固有の名前
- フットプリント-手入力、またはバックアノテートされたもの（下図参照）
- フィールド１-製造業者名（任意、追加が必要）
- フィールド２-製造業者の部品番号（任意、追加が必要）
- フィールド３-販売業者の部品番号（任意、追加が必要）

例:



4.8 フットプリント割当用インポート（バックアノテート）ツール

4.8.1 アクセス:

アイコンで、バックアノテートツールを呼出します。

このツールを使うと、PcbNew で変更されたフットプリントを Eeschema のフットプリントフィールドへ反映させること (imported back) ができます。

Chapter 5

回路図の作成と編集

5.1 はじめに

回路図は１枚のシートのみを使用しても作成可能ですが、規模の大きな回路図の場合は複数のシートで構成することもできます。

回路図が複数のシートから構成される場合を階層構造と呼び、構成する全てのシート（各シートはそれぞれ１つのファイルから成っています）が Eeschema のプロジェクトを構成することになります。階層構造を持つ回路図については 階層回路図 の章に記述があります。

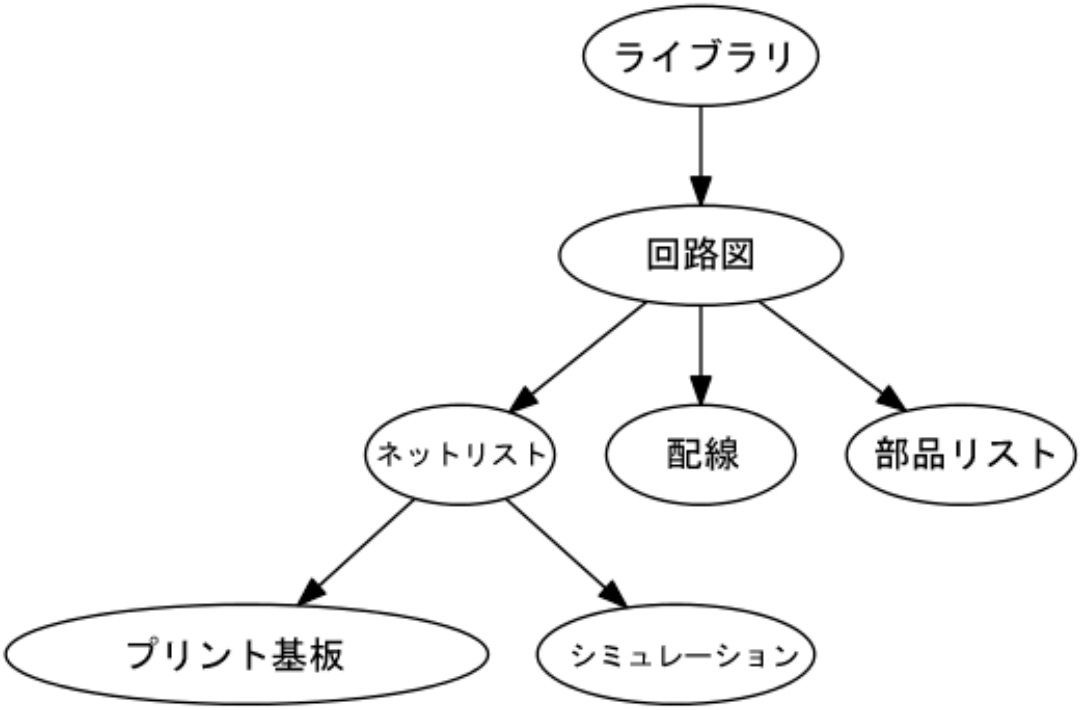
5.2 基本的な検討事項

Eeschema を使う回路図設計は、単なる回路図画像の描画に留まりません。この回路図設計は，開発フローのスタートとなります。:

- 回路図の誤りや欠落の検出。(ERC（エレクトリカル・ルール・チェック）による設計検証)
- 部品表（BOM：Bill Of Material）の自動生成。(カスタマイズされたネットリストと BOM （部品表）ファイルの生成)
- Pspice などの回路シミュレータのためのネットリスト生成。カスタマイズされたネットリストと BOM （部品表）ファイルの生成
- PcbNew を利用したプリント基板設計のためのネットリスト生成。カスタマイズされたネットリストと BOM （部品表）ファイルの生成

回路図は主にワイヤ、ラベル、ジャンクション、バス、電源から構成されます。また回路図を見易くするために、バスエントリ、コメント、破線などのグラフィック要素も配置できます。

5.3 開発フロー

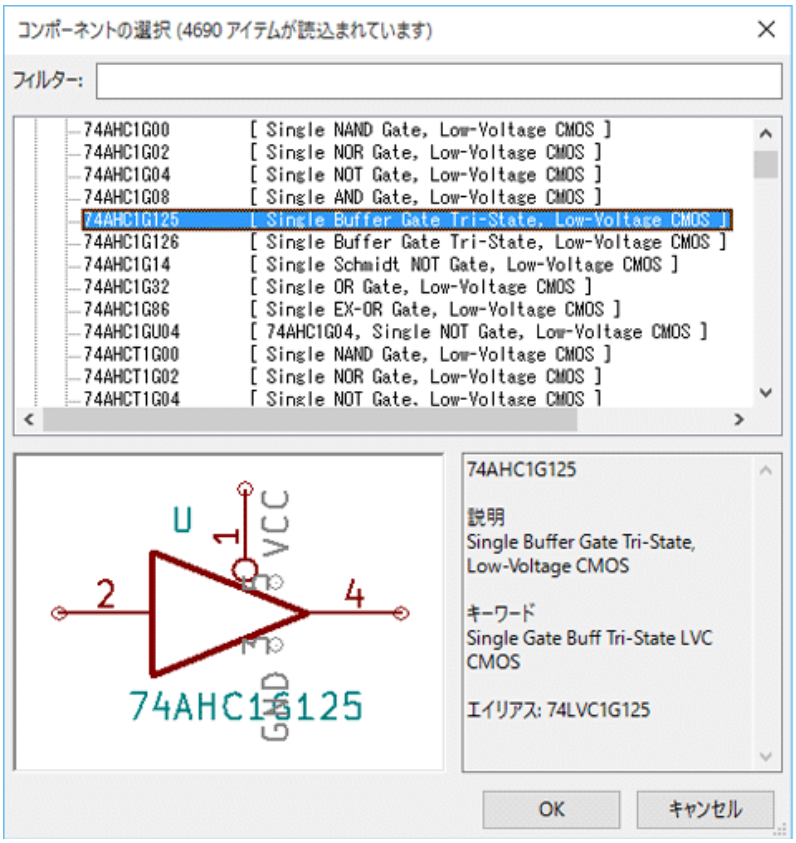


コンポーネントはコンポーネントライブラリから回路図へと配置されます。回路図が完成した後で、PcbNew が回路の接続情報とフットプリントをインポートできるように、ネットリストが作られます。

5.4 コンポーネントの配置と編集

5.4.1 コンポーネントの検索と配置

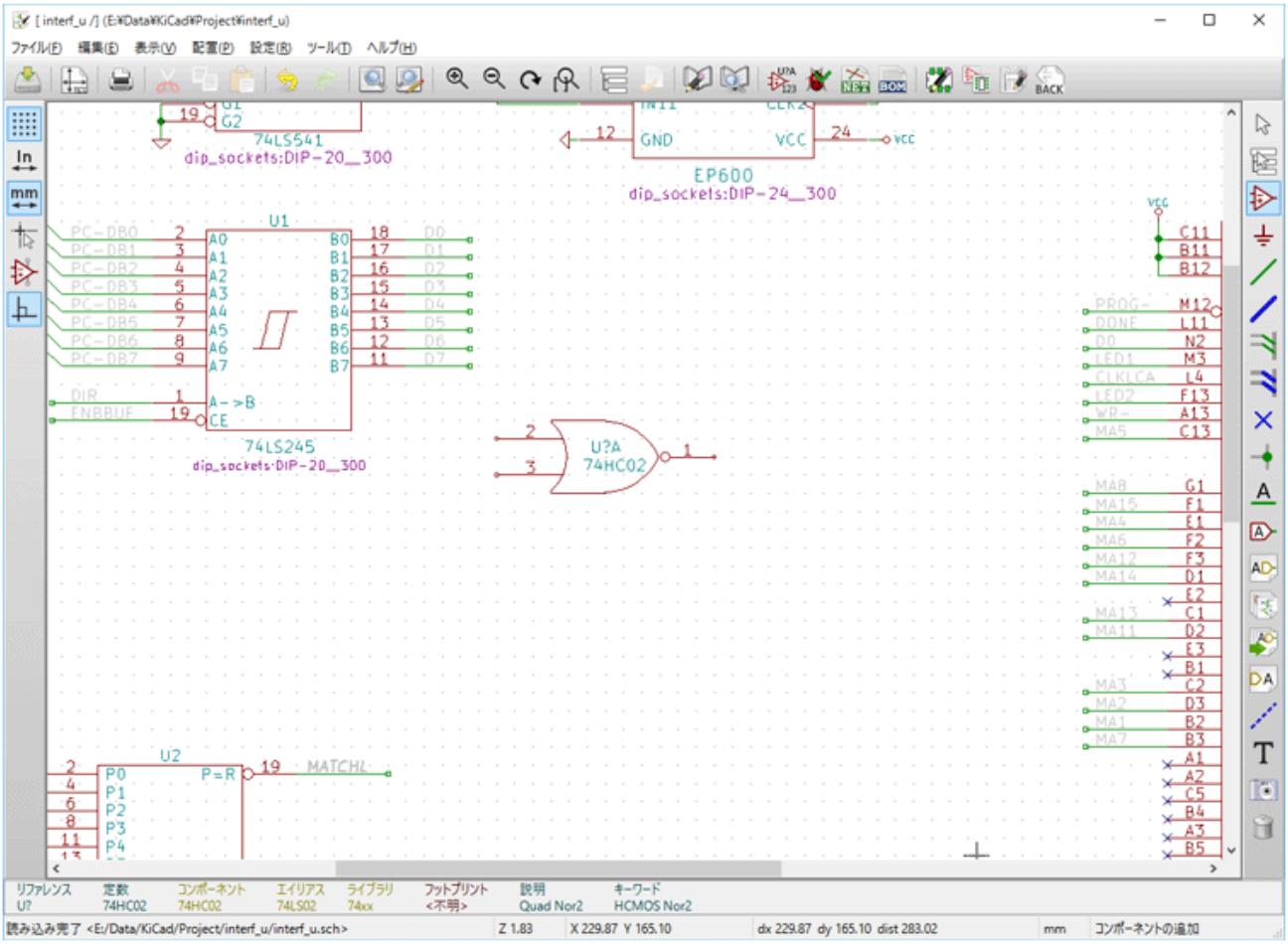

アイコンを使って、回路図にコンポーネントをロードします。ダイアログボックスでは、ロードするフットプリントの名前を指定することができます。



“コンポーネントの選択”ダイアログは、名前、キーワード、検索フィールドへの入力内容でコンポーネントをフィルタリングできます。

コンポーネントを回路図へ配置する前に、ホットキーや右クリックで表示されるコンテキストメニューを使って、コンポーネントの回転（90 度ごと）、横軸／縦軸でのミラー、フィールドの編集ができます。これらの変更は，部品を配置した後でも簡単に行うことができます。

配置中のコンポーネントは次のようになります。:



5.4.2 電源ポート（コンポーネント）

電源ポートもコンポーネントのひとつです（“power”ライブラリに分類されています）。よって、これまで説明したコンポーネントの配置と同じ手順で配置することができます。しかし、これらは頻繁に配置されるものなので、

“電源ポートの配置”ツールが用意されています。このツールは“power”ライブラリを探して直接参照します。

5.4.3 （配置された）コンポーネントの編集と変更

コンポーネントの編集には，以下の２種類があります:

- コンポーネント自身の変更：部品の位置、向き、複数ユニットを持つコンポーネントのユニット選択。
- コンポーネントフィールドの変更：リファレンス、定数、フットプリント、等。

コンポーネントが配置された直後に必要に応じてそれらの部品定数を設定します（特に抵抗やコンデンサなど）。しかし、コンポーネントを配置した直後に部品のリファレンスの割り当てを行ったり、あるいは 7400 のような複数ユニットを持つコンポーネント内の部品番号を設定したりすることは無駄になってしまうかもしれません。これ

らコンポーネントのリファレンスや部品番号は、アノテーション機能を使うことで後から自動的に割り振ることができます。

5.4.3.1 コンポーネントの変更

コンポーネントのプロパティを編集するには、マウスカーソルをコンポーネント上へ移動させ、以下のいずれかの操作をします。:

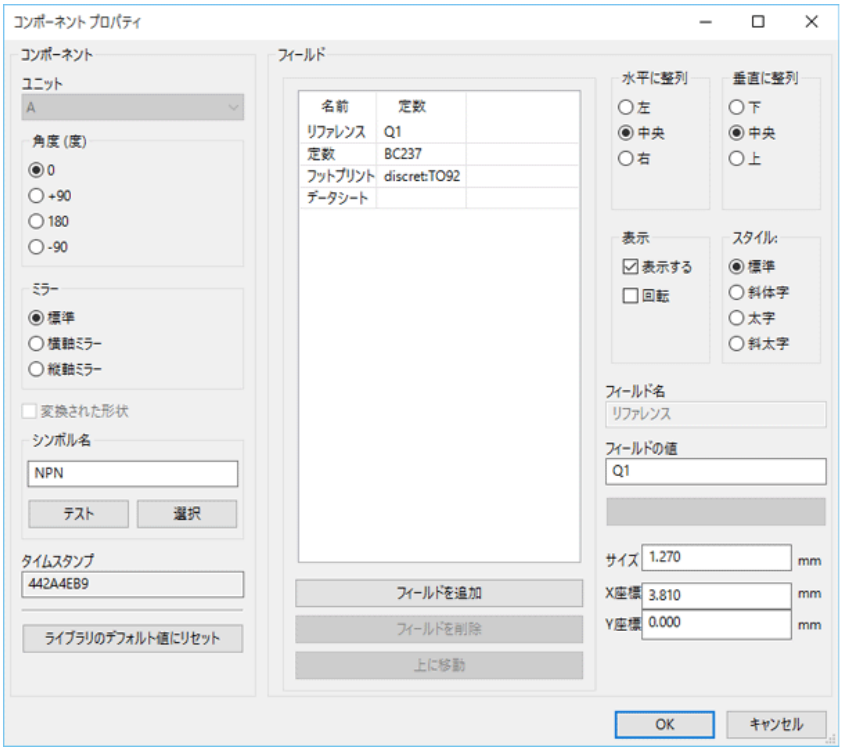
- コンポーネントをダブルクリックして、全てのプロパティを編集することができるダイアログボックス（コンポーネントプロパティ）を開く。
- 右クリックしてコンテキストメニューを開き、表示された編集コマンドを選択する: 移動、回転、編集、削除など。

5.4.3.2 テキストフィールドの編集

リファレンス、定数、位置、向き、テキストサイズ、フィールドの可視性を変更できます。:

- テキストフィールドをダブルクリックし編集する。
- 右クリックしてコンテキストメニューを開き、表示された編集コマンドを選択する: 移動、回転、編集、削除など。

他のオプションの変更や新たにフィールド項目を作成する場合は、コンポーネントをダブルクリックし、コンポーネントプロパティのダイアログを開きます。



それぞれのフィールドについて表示/非表示と回転（表示方向）を設定することができます。表示位置は常に普通に表示される（回転やミラーがない）コンポーネントに対するもので、コンポーネントのアンカー位置への相対座標で指定します。

“ライブラリのデフォルト値にリセット”ボタンは、コンポーネントを元の向きにセットし、オプション、サイズ、位置、各フィールドを初期化します。しかし、回路図情報を壊してしまう可能性があるため、テキストフィールド（リファレンス、定数、フットプリント等）の内容は変更されません。

5.5 ワイヤ、バス、ラベル、電源ポートの接続

5.5.1 はじめに

これら全ての描画要素は、画面右に縦表示されているツールバーに配置されています。

これらの要素を以下に示します:

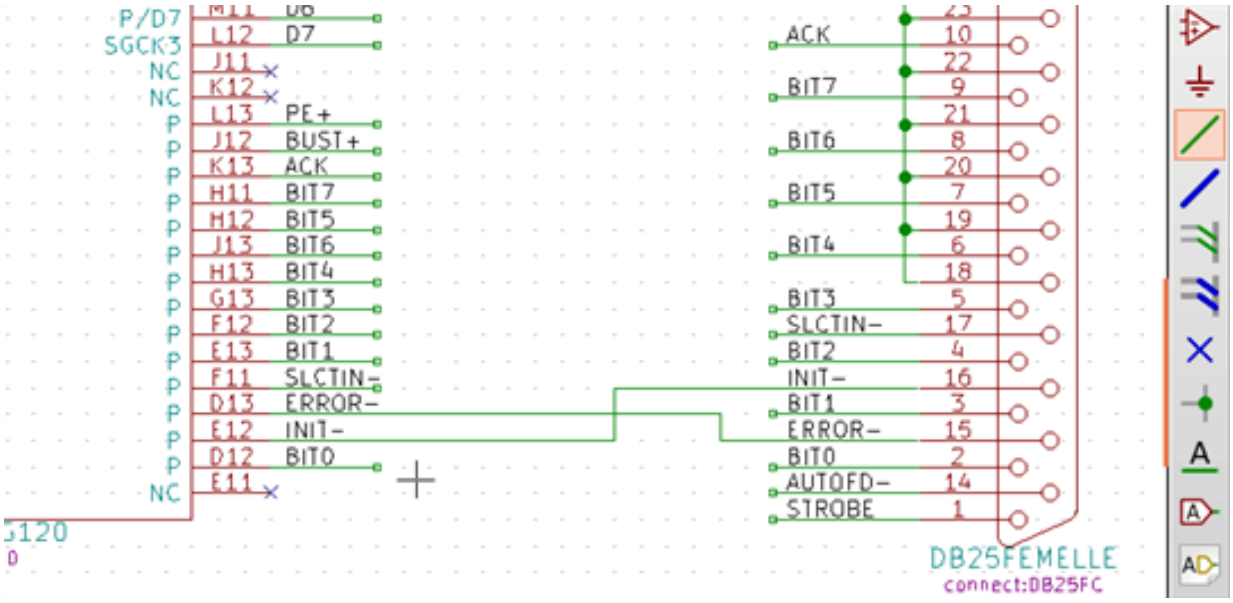
- ワイヤ: コンポーネント間の接続
- バス: バス配線をまとめラベルを用いて接続
- 図形ライン, ポリゴン: （画面を見やすくする）グラフィック表現用
- ジャンクション（接続点）: ワイヤやバスの交差点で接続
- ワイヤ - バスエントリ: バス配線とワイヤの接続。グラフィックのみ!
- ネット名（ローカルラベル）: 接続の作成やラベリング
- グローバルラベル: シート間での接続
- テキスト: コメントとアノテーション用
- “空き端子”フラグ: 接続する必要のないピンを終端。
- 階層シート, とその接続ピン

5.5.2 接続（ワイヤとラベル）

接続を確立する方法は 2 つあります:

- ピン間のワイヤ
- ラベル

以下の図はこれら 2 種類の接続方法を示します。



注 **1:**

ラベルが示す“接続”点は、ラベル一文字目の左下になります。この点は、未接続時には小さな四角で表示されます。

この点は、ワイヤに接しているか、ピンの接続位置に重なっていなければなりません。

注 **2:**

接続を確立するためには、ワイヤの端を他のセグメントかピンへ接続します。

もし配線とピンが重なりあった場合（ピンの終端へ接続されずにワイヤがピンを乗り越えた場合）、これらは接続されません。

注 **3:**

交差したワイヤは暗黙的に未接続となります。もし接続する必要がある場合は、ワイヤの交差点にジャンクション（接続点）シンボルの配置が必要になります。

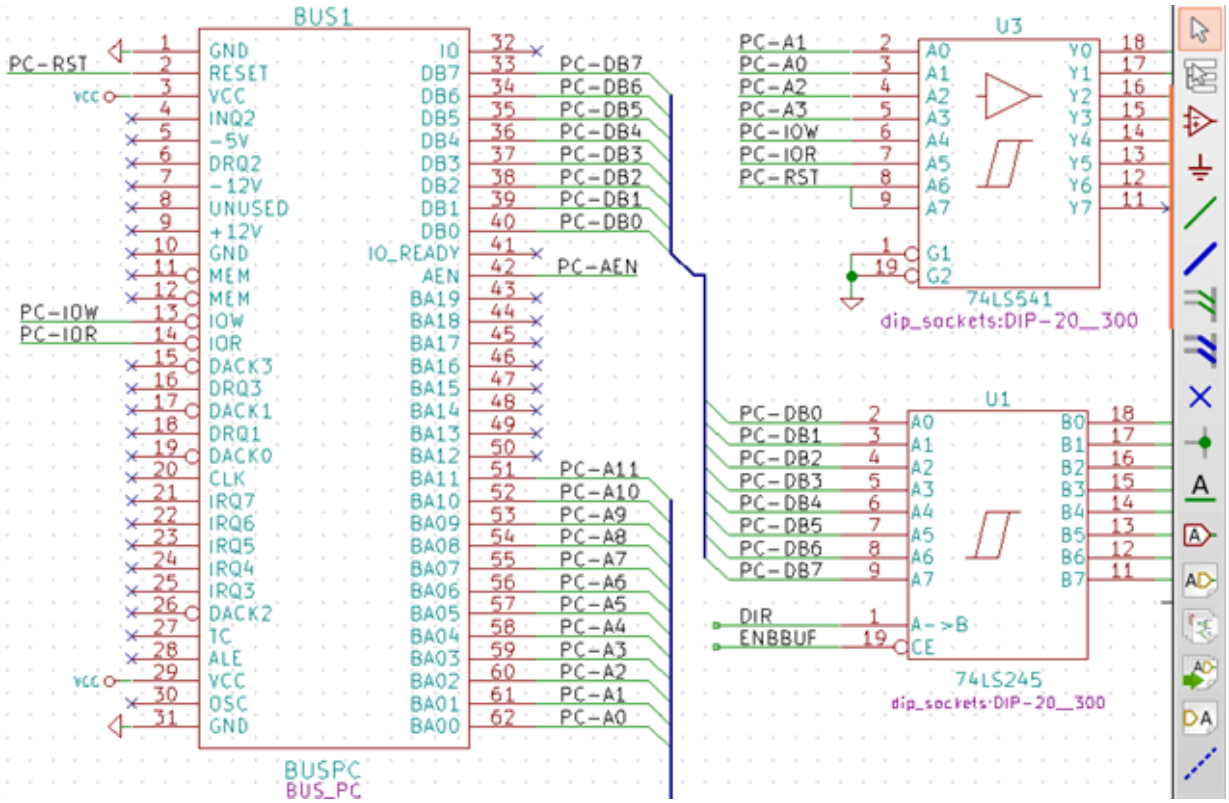
以前に示した図（ワイヤが DB25FEMALE 22, 21, 20, 19 ピンに接続されているもの）では，このジャンクション（接続点）シンボルを使った場合を示しています。

注 **4:**

１つのワイヤに２つの異なるラベルが配置されている場合、これらのラベルで示される両方の信号同士が接続されます。どちらか一方のラベルで接続されている全ての信号が互いに接続されます。

5.5.3 バス接続

以下に示す回路図では、多くのピンがバスへ接続されています。



5.5.3.1 バスのメンバ

回路図を見易くするため、信号の集合体としてバスを利用し、接頭語＋数値というフォーマットで名前が付けられます。上の図では、PCA0、PCA1、PCA2 は PCA バスのメンバとなります。

バスそのものを指す場合は、PCA[N..m] のように呼びます。この場合、N と m はバスの最初と最後のワイヤ番号になります。例えば、PCA バスに 0 から 19 までの 20 本のメンバがある場合、バスの呼び名は PCA[0..19] となります。しかしながら、PCA0、PCA1、PCA2…以外に WRITE、READ のような信号を含めて一つのバスにすることはできません。

5.5.3.2 バスのメンバ同士の接続

バスの同一メンバ間でピンを接続する場合は、ラベルによって接続しなければなりません。（バスは信号の集合体であるため）ピンはバスへ直接接続できません。Eeschema はこのような接続を無視します。

上に示したような例では、ピンに接続しているワイヤへ配置されたラベルによってピン間が接続されます。バスワイヤへのバスエントリ部（45 度曲がっているワイヤ部分）は外観上の見易さを目的としているだけで、Eeschema で回路図としての意味はありません。（論理的な接続である必要はありません）

もしコンポーネントのピン番号が昇順で並んでいるのであれば（メモリやマイクロプロセッサなどで良く見かけます）、以下の手順のように繰り返しコマンド (インサートキー) を用いることで非常に速く接続を行う事ができます。:

- 最初のラベルを配置します（例えば PCA0 ）
- 繰り返しコマンドを使用してメンバのラベルを配置します。Eeschema は理論的に次のピン位置に相当する縦方向に整列した次のラベル（PCA1、PCA2、…）を自動的に生成します。

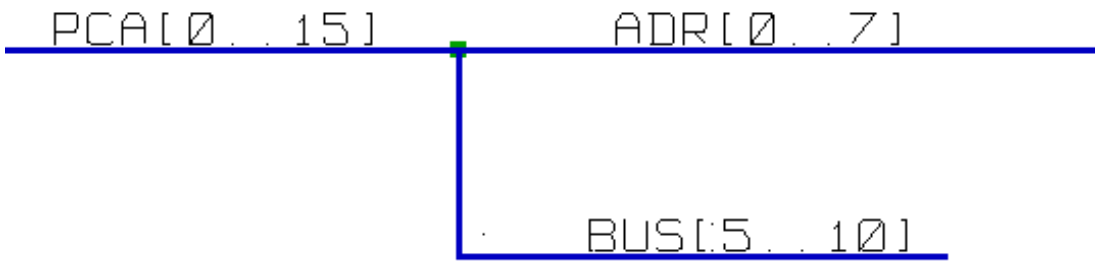
- 最初のラベルの下にワイヤを配置します。同様に繰り返しコマンドを利用し、ラベルの下へワイヤを配置していきます。
- 必要に応じて、同じ方法（最初のエントリを配置し，繰り返しコマンドを使用する）を用い、バスエントリを配置して下さい。

注意
メインメニュー“設定”→ “回路図エディタオプション”で、繰り返しに関するパラメータを設定することができます:

- アイテムを垂直方向にリピート（縦方向の間隔)。
- アイテムを水平方向にリピート（横方向の間隔)。
- ラベルのカウントアップリピート（2, 3,…のようなインクリメント (加算) ／またはデクリメント (減算)）

5.5.3.3 バスのシート間接続

階層構造になっている回路図において、シート間で異なる名前のバス同士を接続する必要がある場合があります。この接続の手順を以下に示します。



バス PCA[0..15]，ADR[0..7]，BUS[5..10] は互いに接続されています。(接続点に注意してください。縦方向のバスが横方向のバスの中央で接続されています。)

より正確に言うならば、バスを構成する対応するメンバ同士が接続されます。: 例えば、PCA0 と ADR0 が接続されています。(同様に、PCA1 と ADR1 …PCA7 と ADR7)

さらに、PCA5 と BUS5 と ADR5 も接続されてます。(PCA6 と BUS6 と ADR6，PCA7 と BUS7 と ADR7 も同様です)

PCA8 と BUS8 も接続されることになります。(PCA9 と BUS9，PCA10 と BUS10 も同様です)

5.5.4 電源ポートの接続

コンポーネントに電源ピンがある場合も、他の信号と同様に接続する必要があります。

ゲートやフリップフロップのようなコンポーネントは見えていない電源ピンを持っています。これらの扱いは少し厄介です。難点は以下の２つです:

- 電源ピンが非表示であるため、ワイヤを接続できない。
- 電源ピンの名前が解らない。

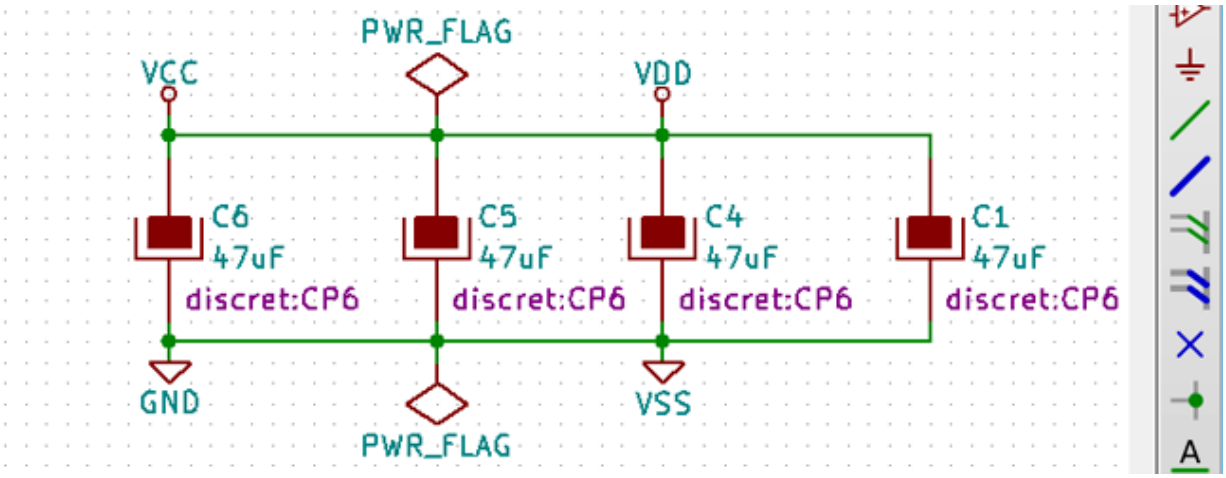
これら電源ピンの設定を可視に変更し接続することはあまり良い方法とは言えません。これをしてしまうと，回路図が読みにくくなってしまい、また普通の回路図の慣習に反することになってしまいます。

注意
これらの非表示となっている電源ピンを表示させたい場合は、メインメニュー“設定”→“回路図エディタオプション”をたどるか、左ツールバー（オプションツールバー）の  アイコンをクリックして、“非表示ピンの表示”オプションをチェックします。

Eeschema は、同じ名前の全ての非表示の電源ピンを自動的に接続します。異なる名前の非表示電源ピン（例えば、TTL デバイスの“GND”と MOS デバイスの“VSS”）を接続することが必要になることもあるでしょう。; このような場合に電源ポートシンボルを使います。

ラベルを利用して電源ピンを接続することは推奨されません。ラベルは“局所的 (local)”な接続機能しか無く、非表示の電源ピンは接続されません。

以下の図は、電源ポートの接続例です。



この例では、GND と VSS、VCC と VDD が接続されています。

２つの PWR_FLAG シンボルがあります。これは、２つの電源ポート VCC と GND が本当に電源 (出力) へと接続されていることを示します。これら２つのフラグがない場合、ERC は *Warning: power port not powered* という診断結果を出力します。

これらすべてのシンボルは、コンポーネントとして“power”ライブラリに収められています。

5.5.5 “空き端子”フラグ

このシンボルは、ERC チェック時に不要な警告を出さないようにするために役立ちます。ERC で接続忘れの空きピンを確実に見つけることができます。

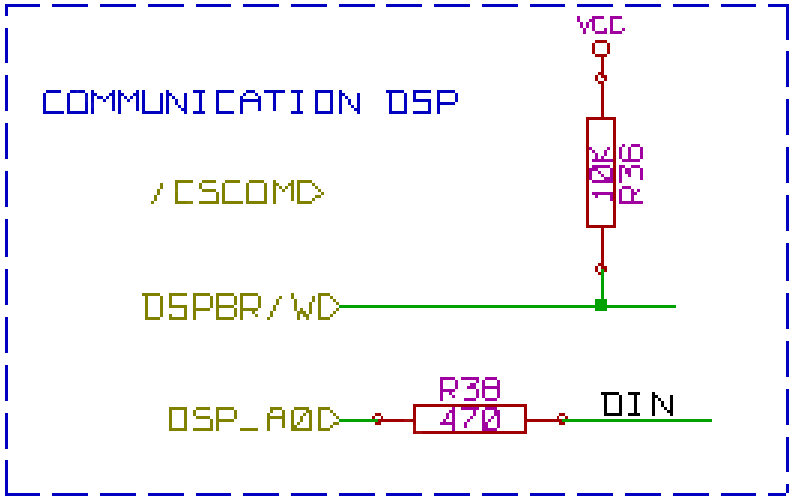
ピンを無接続の空き端子としたい場合、空き端子フラグ（ツール  ）を配置することが必要です。このシンボルは生成するネットリストに影響を与えません。

5.6 回路図作成に関する補足

5.6.1 テキストコメント

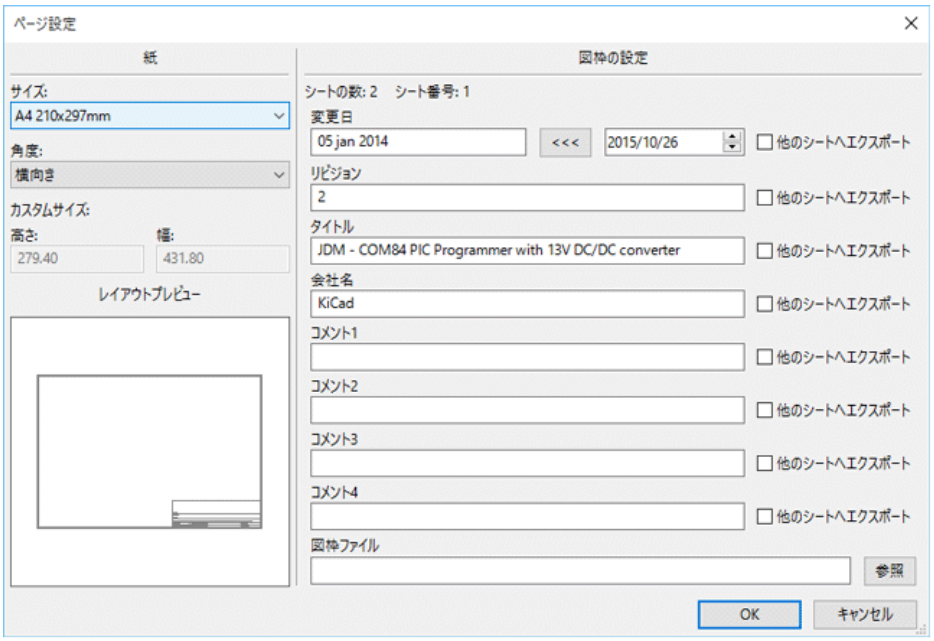
図形テキストのような注釈（コメント）を配置することは（回路図を理解するために）有用です。“テキストの配置”ツール（  ）と“図形ラインかポリゴンを配置”ツール（  ）は、素子同士の接続を行うラベルやワイヤとは異なり、このような用途を意図しています。

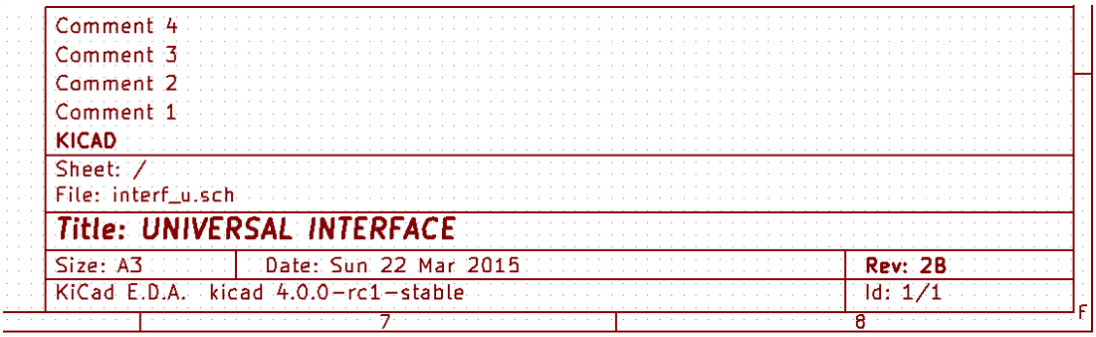
図形コメントの例を以下に示します。



5.6.2 シートの表題欄（タイトルブロック）

表題欄は、  “ページの設定”ツールで編集することができます。



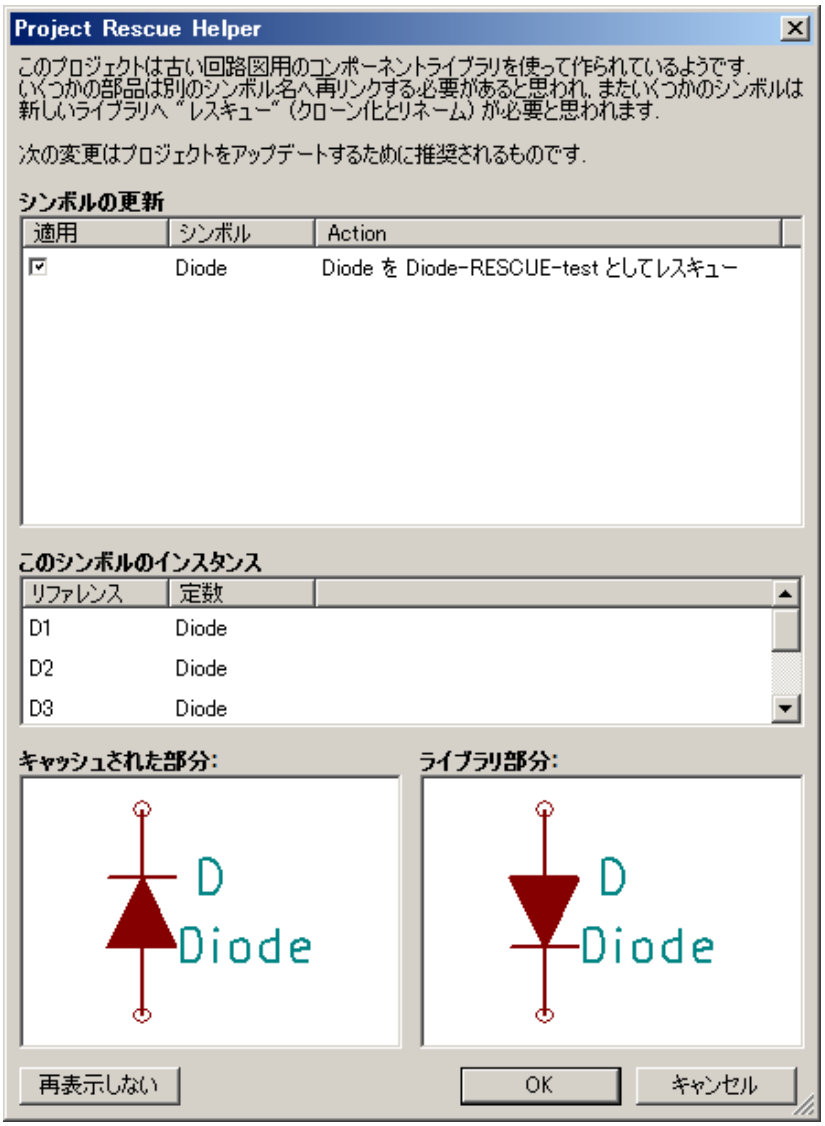


シート数Y、シート番号X（Sheet X/Y）は自動更新されます。

5.7 キャッシュされたコンポーネントのレスキュー

デフォルトでは、Eeschema は設定されたパスに従ってライブラリからコンポーネントを読み込みます。これは古いプロジェクトを読み込む時に問題を引き起こします。: ライブラリにあるコンポーネントがプロジェクトで使われた時より後に変更されていた場合、プロジェクトのコンポーネントは自動的に新しいバージョンへと置き換えられます。新しいバージョンが互換性を失っていたり、違う方向を向いているようなことがあると、回路図の破損につながります。

しかしながら、プロジェクトの保存時にはキャッシュされたライブラリも一緒に保存されます。これにより、プロジェクトは全てのライブラリがなくても配布できます。キャッシュとシステムライブラリの両方に存在するコンポーネントを使ったプロジェクトを読み込むと、Eeschema は衝突検出のためにライブラリを調べるでしょう。発見された衝突は次のダイアログに一覧表示されます。:



この例では、元々のプロジェクトはカソードが上向きのダイオードを使っていますが、現在のライブラリはカソードが下向きのものを含んでいます。この変更はプロジェクトを台無しにします！ここで OK ボタンを押すと、古いコンポーネントのシンボルは特別な“レスキュー”ライブラリへと保存され、このシンボルを使う全てのコンポーネントは名前の衝突を避けるために変更 (rename) されます。

もし Cancel ボタンが押されると、レスキューは作成されず、Eeschema はデフォルトで全て新しいコンポーネントを読み込むでしょう。変更されなかった場合でも、前に戻ってレスキュー機能を再度実行できます。: ツールメニューから“キャッシュされたコンポーネントのレスキュー”を選んで、ダイアログを再び呼び出します。

このダイアログを表示させたくない場合は、“次回から表示しない”ボタンを押してください。デフォルトで何もせずに新しいコンポーネントを読み込むようになります。このオプションは、コンポーネントライブラリの設定で元に戻せます。(“回路図の読込み時にキャッシュ／ライブラリの衝突をチェック”ボックスをチェックする)

Chapter 6

階層回路図

6.1 はじめに

シート数が２～３枚で済まないようなプロジェクトでは、階層的表現を用いるのが一般的によい解決策となります。この種のプロジェクトを管理したい場合、次のことが必要になるでしょう:

- 大きなサイズのシートを使用する。その場合、印刷と取り扱いの問題が生じます。
- シートを数枚使用する。これは階層構造に至ります。

完全な回路図は、ルートシートと呼ばれるメインの回路図シートおよび階層を構成するサブシートから構成されます。さらに、設計を個別のシートにうまく分割すると可読性が改善されます。

ルートシートからは、全てのサブシートを辿ることができなければなりません。上部ツールバーのアイコン  で統合された“階層ナビゲーター”を呼び出すことにより，Eeschema では階層回路図の管理が非常に簡単に行えます。

階層には２種類が同時に存在し得ます：１つ目は、すでに開いて使用しているような一般的に使用されているものです。２つ目は、回路図で使われるコンポーネントの実体を表すもので、実際にはコンポーネントの内部構造を記述した回路図に対応します。

この２つ目のタイプはよく集積回路 (IC) の開発で使用されます。この場合には作成中の回路図で機能ライブラリを使用しなければならないからです。

Eeschema は現在、この第２のケースに対応していません。

階層は次のようなものです:

- 単一：任意のシートを一度だけ使用する。
- 複合：任意のシートを２回以上使用する（複数のインスタンス)。
- 平面 (Flat) ：単一の階層であるが、シート間の接続は記述されない。

Eeschema はこれらの階層を全て扱うことが可能です。

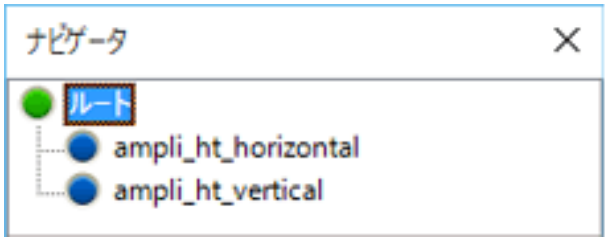
階層回路図の作成は簡単です。階層全体はルート回路図から始まるように管理され、ただ一つの回路図しかないように見えます。

次の２つの重要なステップを理解する必要があります：

- サブシートの作成方法。
- サブシート間の電気的な接続方法。

6.2 階層内のナビゲーション

上部ツールバーの  “回路図の階層ナビゲータを表示”で呼び出される階層ナビゲータツールを使用するとサブシート間のナビゲーションが非常に簡単に行えます。



シート名を左ダブルクリックすると、そのシートに移動できます。

右ツールバーにある  “階層の上下移動”ツールにより、ルートシートあるいはサブシートに素早く移動可能です。ツールを選択後に以下の操作を行います：

- シート内でシート名を左クリックすると、そのシートに移動します。
- サブシートでシート名以外の場所を左クリックすると、上の階層のシートに移動します。

6.3 ローカルラベル、階層ラベル、グローバルラベル

6.3.1 プロパティ

ローカルラベル（  ツール）は、あるシート内のみで接続される信号です。階層ラベル（  ツール）は、あるシート内のみで接続される信号であると同時に親シートに配置された階層ピンに接続されています。

グローバルラベル（  ツール）は階層全体に渡って信号を接続しています。非表示の電源ピン（*power in* および *power out* タイプ）も全階層に渡って互いに接続されているので、グローバルラベルに似ています。

注意
（単一または複合）階層内では、階層ラベルとグローバルラベルのどちらか、または両方を使用可能です。

6.4 階層のヘッドライン（見出し）作成

次のことをする必要があります:

- “シートシンボル”という階層シンボルをルートシート内に配置します。
- ナビゲーターを使用して新規回路図（サブシート）に入り、他の回路図と同様に作成します。
- 新しく作成した回路図（サブシート）にグローバルラベル (HLabels) を配置して２つの回路図間に電気的接続を作成します。また、シートラベル (SheetLabels) という同じ名前を持つラベルをルートシートに配置します。これらのシートラベルはルートシートのシートシンボルや標準的なコンポーネントピンのような他の回路図要素に接続されます。

6.5 シートシンボル

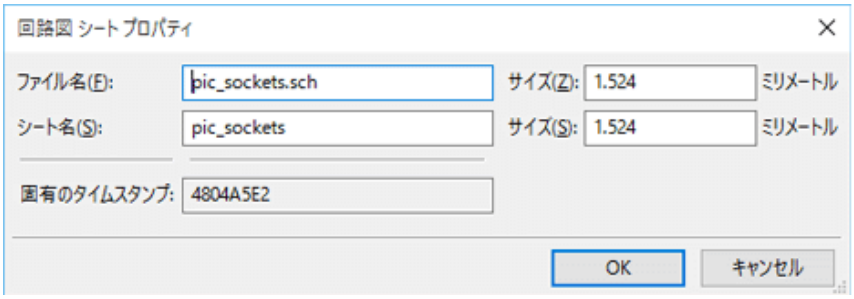
対角上の２点を指定して作成した矩形でサブシートを表します。

この矩形のサイズは、サブシート内のグローバルラベル (HLabels) に対応した特定のラベルや階層ピンを後で配置可能なものでなければなりません。これらのラベルは通常のコンポーネントピンに似ています。

“階層シートの作成”ツールを選択します。

左クリックして矩形の左上角を配置します。矩形が十分な大きさとなったら再度左クリックして右下角を配置します。

この時、(階層ナビゲーターを使用し、対応する回路図に移動するために）このサブシートのファイル名とシート名の入力が要求されます。



少なくともファイル名の入力が必要です。シート名がない場合、ファイル名がシート名として使用されます（この方法はよく行われています)。

6.6 接続 - 階層ピン

作成したシンボル用の接続点（階層ピン）をここで作成します。

これらの接続点は通常のコンポーネントピンと似ていますが、１つの接続点だけで複数の信号からなるバスを接続できます。

次のような２つ方法があります:

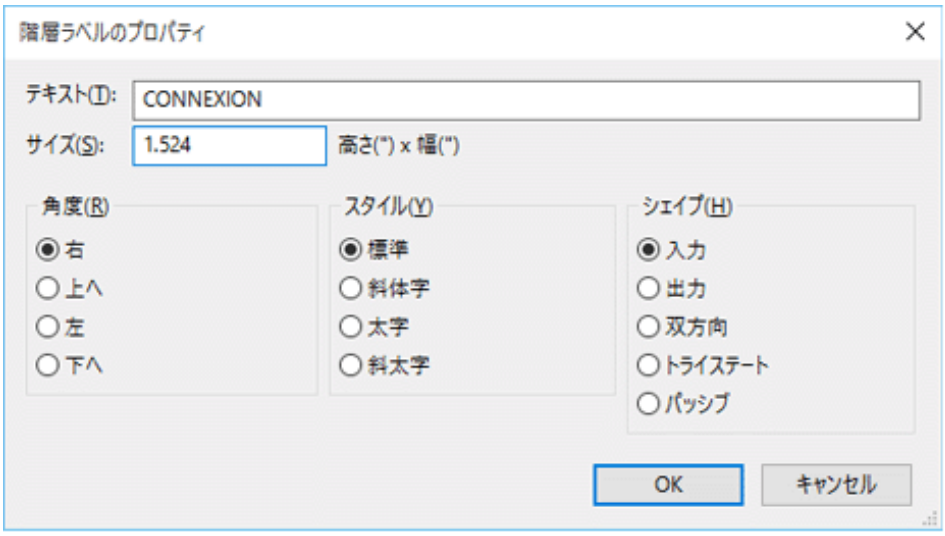
- 必要なピンをサブシート作成前に配置（手動による配置）。
- 必要なピンおよびグローバルラベルをサブシート作成後に配置（半自動配置）。

特に好ましいのは２つ目の方法です。

手動配置:

-  ツールを選択します。
- ピンを配置したい階層シンボルをクリックします。

以下は“CONNEXION”と言う名前の階層ピンを作成する例です。



階層ラベルのプロパティ（右クリックしてコンテキストメニューの“階層ラベルの編集”を選択）を編集して，グラフィカルな属性とサイズを定義できます。後で行うことも可能です。

様々なピンシンボルが使用可能です:

- 入力（Input）
- 出力（Output）
- 双方向（Bidirectional）
- トライステート（Tri-State）
- パッシブ（Passive）

これらのピンシンボルは単なるグラフィカルな強調で、それ以外の役割はありません。

自動配置:

-  “シート内の対応する階層ラベルからインポートされた階層ピンを入力”ツールを選択します。

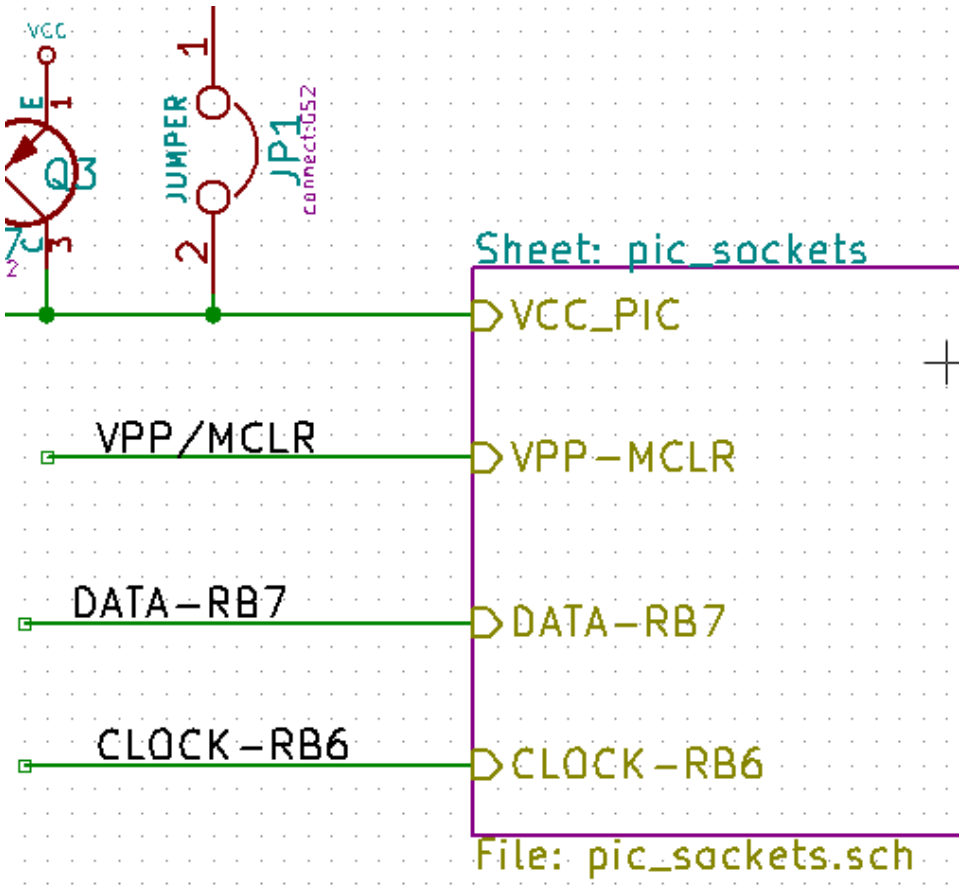
- 階層シンボルをクリックして、そこからグローバルラベルに対応するピンをインポートして対応する回路図に配置します。新しいグローバルラベルが存在する場合、つまり配置済みのピンに対応したものでないなら、階層ピンが現れます。
- このピンを配置したい場所でクリックします。

必要なすべてのピンはエラーなく速やかに配置することが可能です。それらの向きや方向は対応するグローバルラベルと一致しています。

6.7 接続 - 階層ラベル

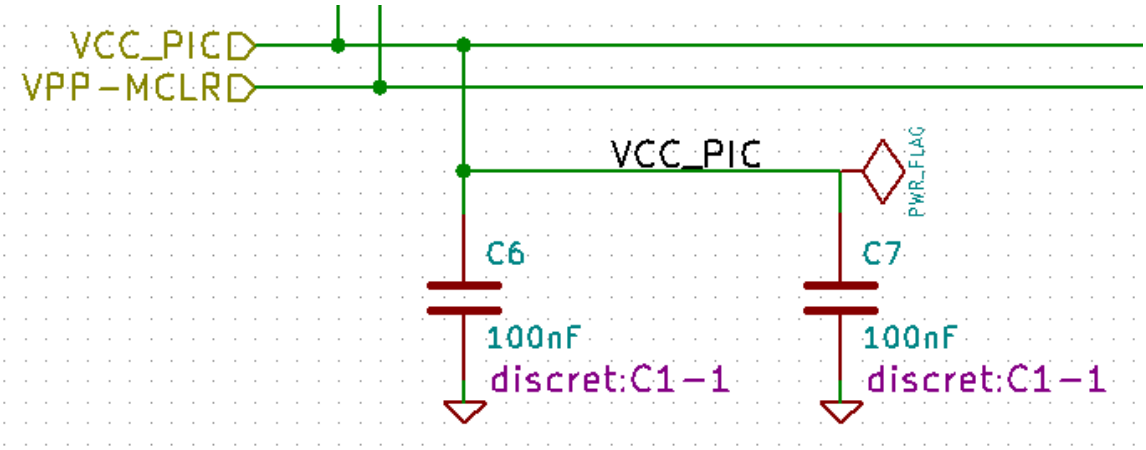
作成したシートシンボルの各ピンはサブシート内の階層ラベルと一致していなければなりません。階層ラベルはラベルと似ていますが、サブシートおよびルートシート間の接続を行います。２つの相補的なラベル（ピンと HLabel ）のグラフィカルな表示は似ています。階層ラベルの作成は、  “階層のラベル入力”ツールで行います。

以下はルートシートの例です。



ピン VCC_PIC がコネクタ JP1 に接続されていることに注意して下さい。

サブシート内での対応する接続は次のようになります:



２つの階層シート間を接続する２つの対応する階層ラベルがあるのがさらにわかります。

注意
既述の構文 (Bus [N. .m]) に従って、２つのバスを接続する階層ラベルおよび階層ピンを使うことが可能です。

6.7.1 （単純な）ラベル、階層ラベル、グローバルラベル、非表示電源ピン

ワイヤによる接続以外に、接続を行う様々な方法について説明します。

6.7.1.1 （単純な）ラベル

単純なラベルはローカルな接続に使います。つまり、接続は配置されている回路図シートだけ制限されます。これは次の理由によります:

- 各シートにはシート番号が存在する。
- このシート番号はラベルに関連付けられる。

そのため、シート番号３にラベル“TOTO”を配置した場合、実際のラベルは“TOTO_3”となります。シート番号１（ルートシート）にラベル“TOTO”を配置した場合、実際には“TOTO_3”ではなく“TOTO_1”というラベルを配置したことになります。これはシートが１つしかない場合でも常にこうなります。

6.7.1.2 階層ラベル

単純なラベルで言えることは、階層ラベルにも当てはまります。

このため、同一シート内で HLabel の“TOTO”はローカルラベル“TOTO”に接続されていると見なされますが、別のシートの HLabel あるいは“TOTO”というラベルには接続されません。

しかし、HLabel はルートシートに配置された階層シンボル内の対応するシートラベル・シンボルに接続されていると見なされます。

6.7.1.3 非表示電源ピン

非表示の電源ピンは、同一名であるなら互いに接続されているものと見做されます。このため、“Invisible Power Pins“として宣言されている VCC という名前の全ての電源ピンは、それが置かれているどのシートでも互いに接続され、同電位 VCC を形成します。

このことは，あるサブシートに VCC ラベルを配置した場合、そのラベルが VCC ピンには接続されないということを意味します。それは、このラベルが実際には VCC_n であるからです。ここで n とはシート番号です。

この VCC ラベルを同電位 VCC に実際に接続したいなら、VCC 電源ポートによって明示的に非表示電源ピンへ接続する必要があります。

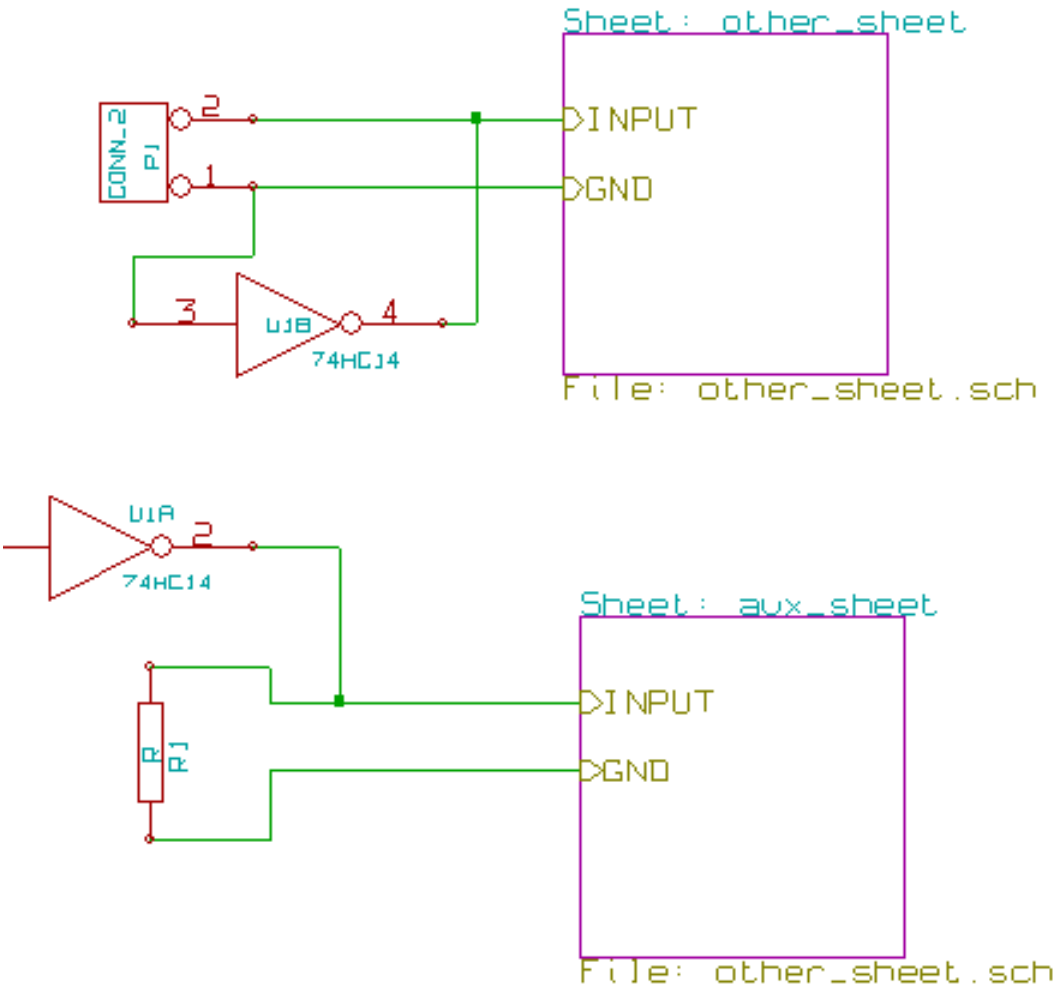
6.7.2 グローバルラベル

同一名のグローバルラベルは、全ての階層に渡って互いに接続されています。

(vcc …のような電源ラベルはグローバルラベルです)

6.8 複合階層

一例を示します。同じ回路図が２回使用されています（２つのインスタンス)。２つのシートのファイル名が同じなので (“other_sheet.sch”) 、２つのシートは同じ回路図を共有します。しかし、シート名は異なっていなければなりません。

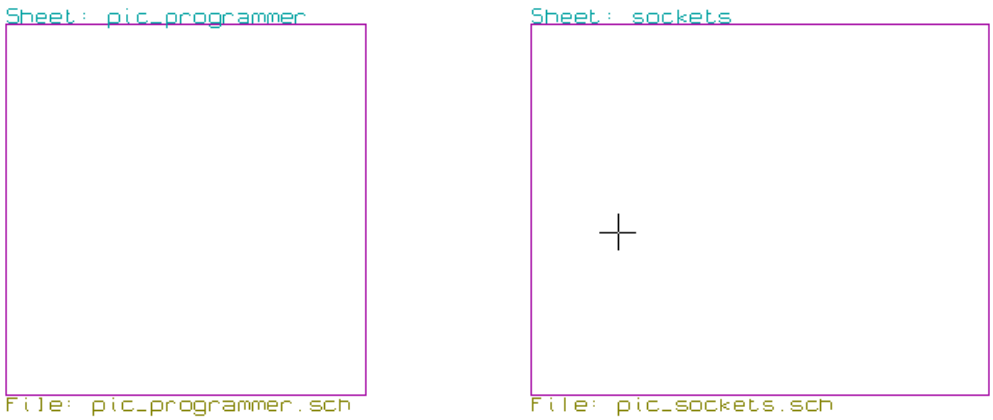


6.9 平面階層

シート間の接続を作らずに（平面階層 (flat hierarchy) ）、シートを多数使うプロジェクトの作成が可能です。それには次のルールを順守して下さい:

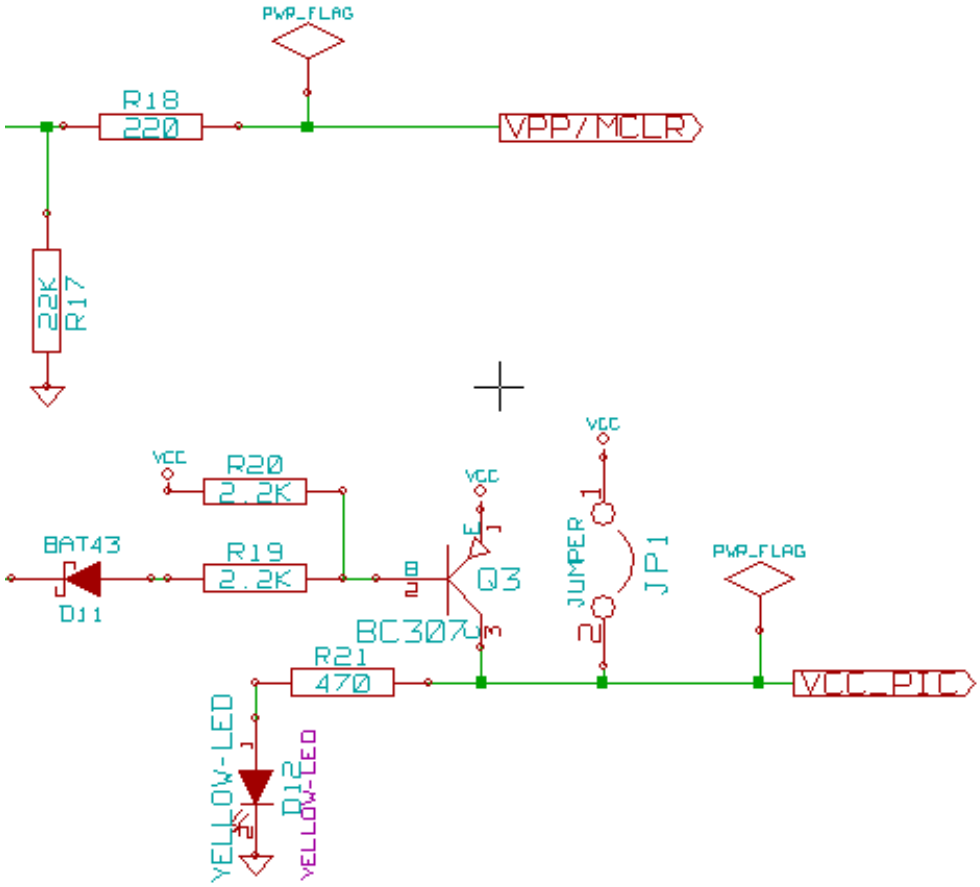
- ルートシートを作成し、他のすべてのシートをそれに含めます。ルートシートはシート間のリンクとして機能します。
- 明示的な接続はまったく必要ありません。
- シート間のすべての接続には、階層ラベルではなくグローバルラベルを使用します。

ルートシートの例を以下に示します。

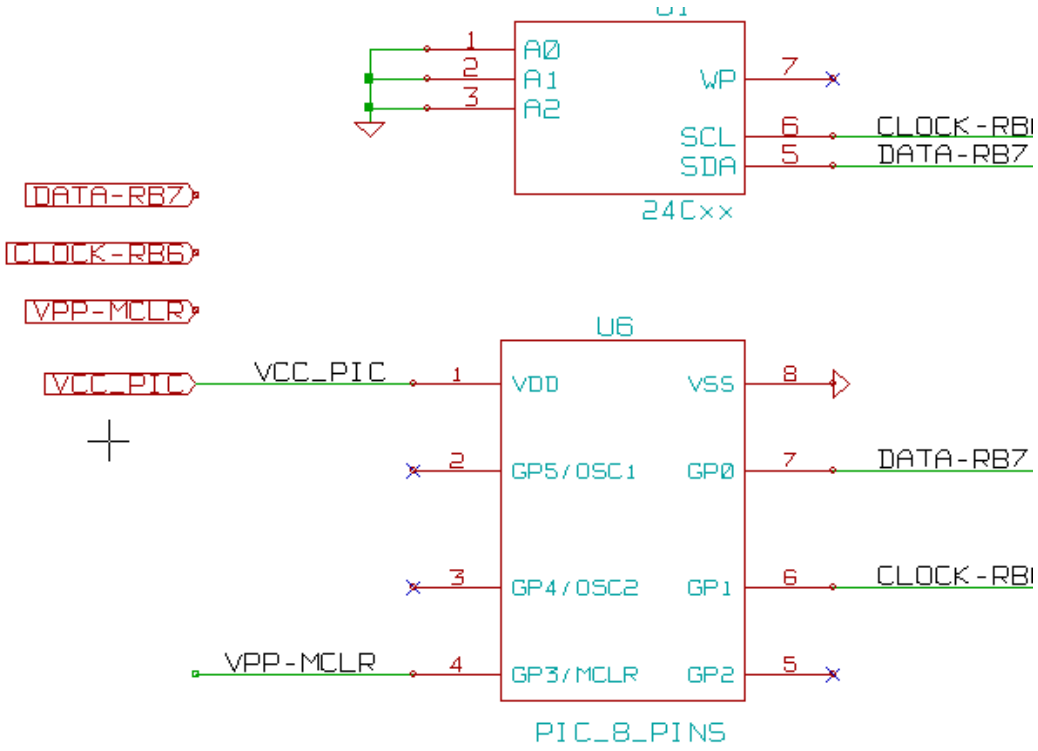


２ページあり、それらはグローバルラベルで接続されています。

こちらが pic_programmer.sch です。



こちらが pic_sockets.sch です。



グローバルラベルに注目してください。

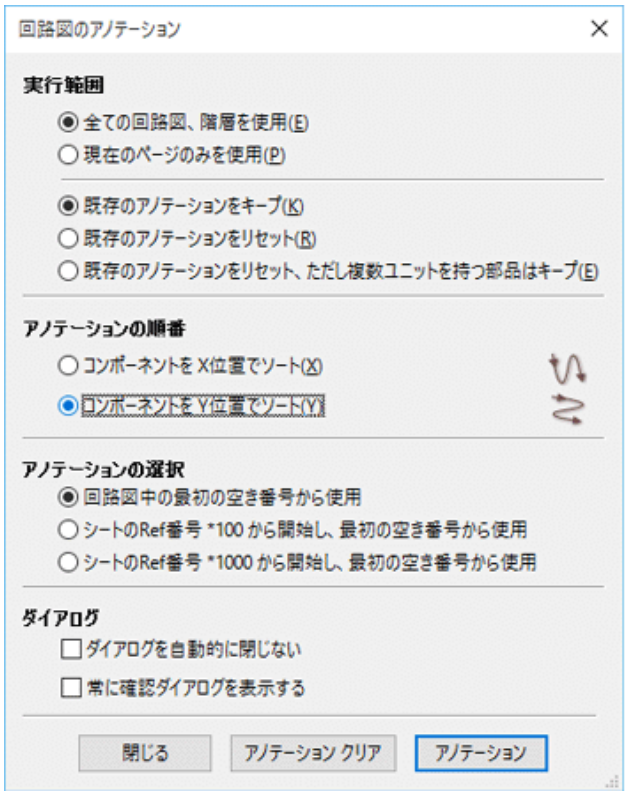
Chapter 7

自動アノテーション

7.1 はじめに

“回路図のアノテーション”ツールを用いることで、回路図中のコンポーネントに自動的にリファレンス（参照番号）を割り当てることができます。複数ユニットを持つコンポーネントについては，使用パッケージ数が最小となるように部品番号を割り当てます。アノテーションツールは、アイコン 

をクリックすることで利用できます。以下に“回路図のアノテーション”のメインウィンドウを示します。



次のように、さまざまな利用方法があります。:

- 全てのコンポーネントをアノテート（「既存のアノテーションをリセット（Ｒ）」を選択）

- 既存の複数ユニットを持つコンポーネント内の順番を入れ替えないで、全てのコンポーネントをアノテート。(「既存のアノテーションをリセット、ただし複数ユニットを持つ部品はキープ（Ｅ)」を選択)。
- 新しく追加したコンポーネントのみをアノテート（「既存のアノテーションをキープ（Ｋ)」を選択)。
- 全階層をアノテート（「全ての回路図，階層を使用（Ｅ)」を選択)。
- 現在のシートのみをアノテート（「現在のページのみ使用（Ｐ)」を選択)。

’既存のアノテーションをリセット、ただし複数ユニットを持つ部品はキープ’ オプションは、複数ユニットを持つコンポーネント内の全ての関連性を保存します。これは、もし U2A と U2B があったなら、それぞれ U1A と U1B へと番号が振られることはあっても、U1A と U2A、または U2B と U2A とはならないということです。これは、ピングループを確実に維持したい時、配置するのに都合がいいように定義した子部品がある時、に役立ちます。

「アノテーションの順番」オプションでは、それぞれのシート内での番号の振り方を指定することができます。

特別な場合を除いて、以前のアノテーション結果を変更しない場合は、プロジェクト全体（全てのシート）と新しいコンポーネントが自動アノテーションの対象となります。

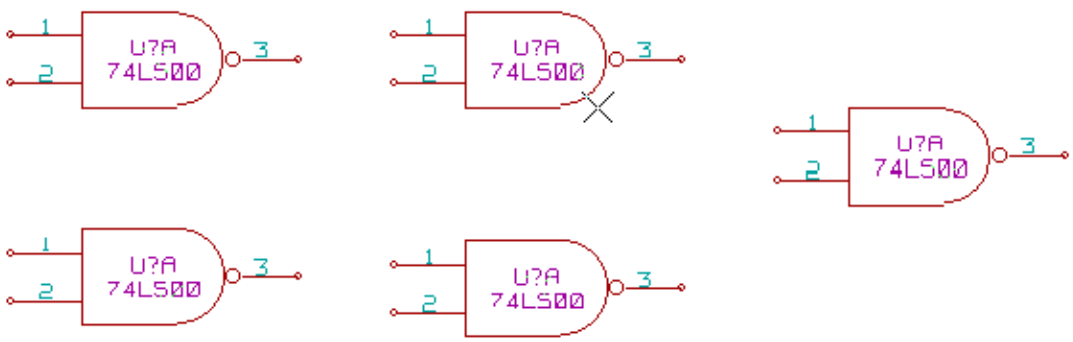
「アノテーションの選択」オプションでは、リファレンス（参照番号）の計算方法を指定します。:

- 回路図中の最初の空き番号から使用する：コンポーネントは（各部品記号につき）1 からリファレンスが振られます。前回のアノテーションをキープする場合は、使われていない番号から利用されます。
- シートの Ref 番号を *100 から開始し、最初の空き番号から使用する：シート１では 101 から，シート２では 201 からリファレンスが振られます。それぞれの部品記号（ＵやＲ）が１シート内で 99 を超えてしまった場合は継続して以降の番号が振られ、例えばシート２では 200 番台の最初の空き番号からリファレンスが振られます。
- シートの Ref 番号を *1000 から開始し、最初の空き番号から使用する：シート１では 1001 からリファレンスが振られ、シート２では 2001 からリファレンスが振られていきます。

7.2 例

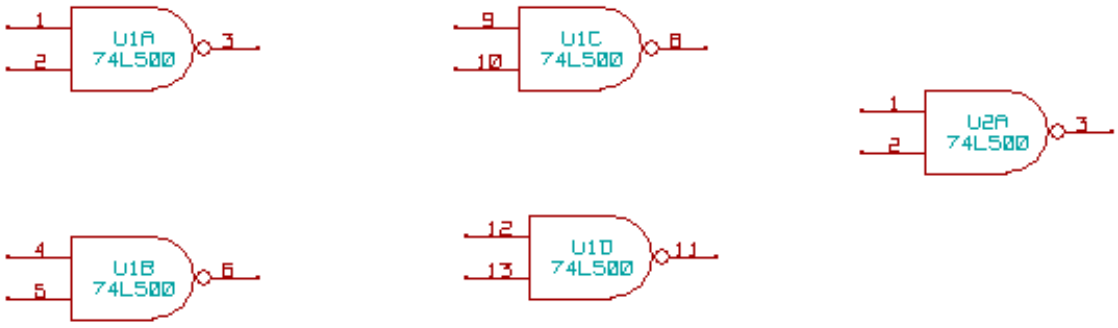
7.2.1 アノテーションの順序

部品配置後、未だリファレンスが振られていない５つの素子を例とします。

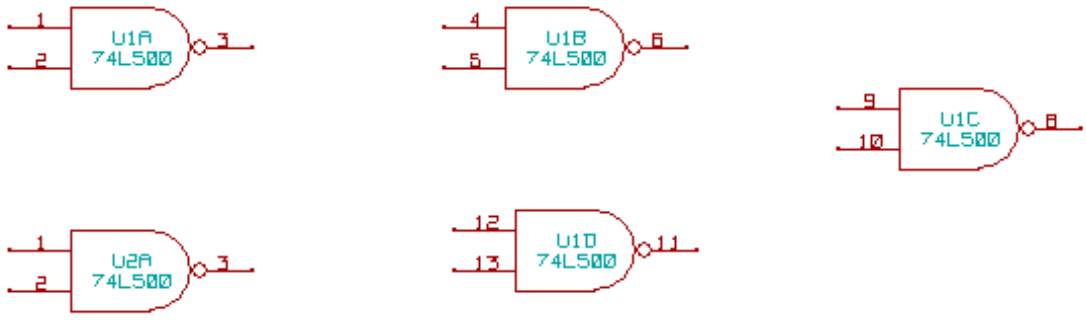


アノテーションを実行すると、以下のような結果が得られます。

コンポーネントをＸ位置でソートした場合：



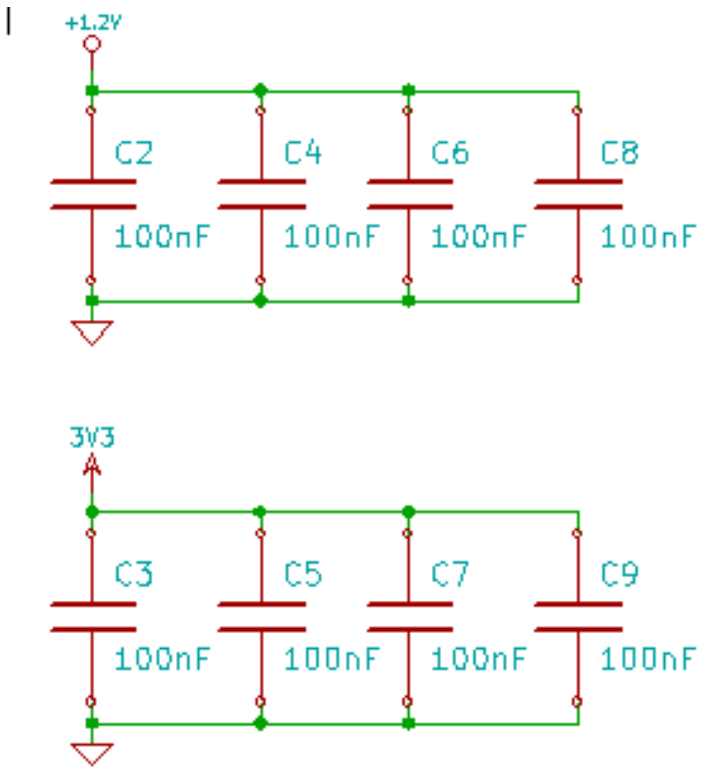
コンポーネントをＹ位置でソートした場合。



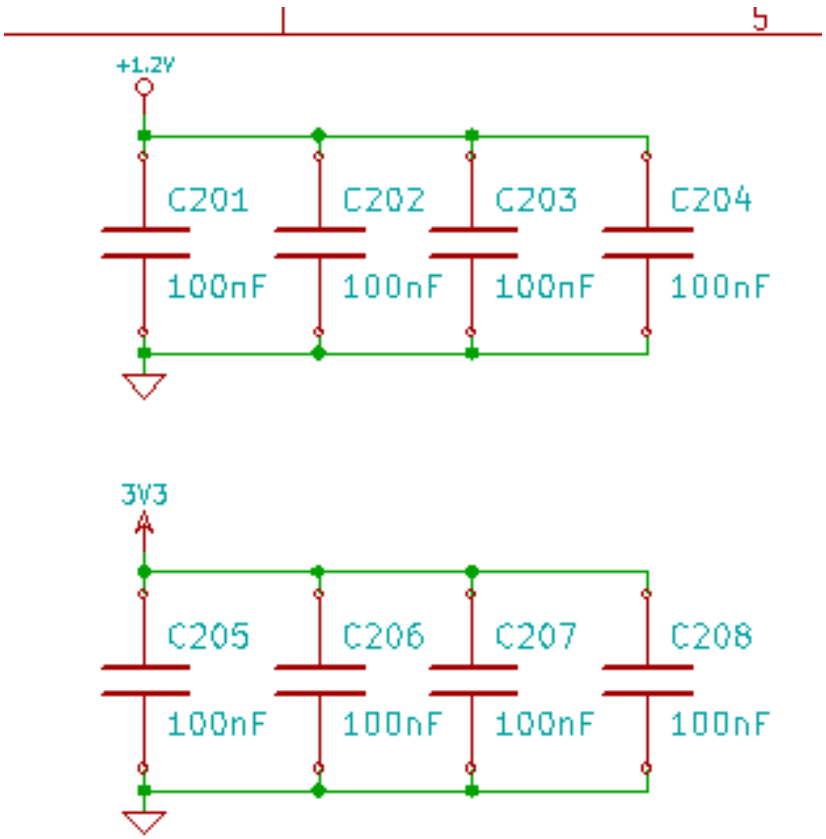
アノテーションにより、74LS00 の４つのゲートが U1 パッケージにまとめられ、５番目のゲートは次の U2 へと分類されました。

7.2.2 アノテーションの選択

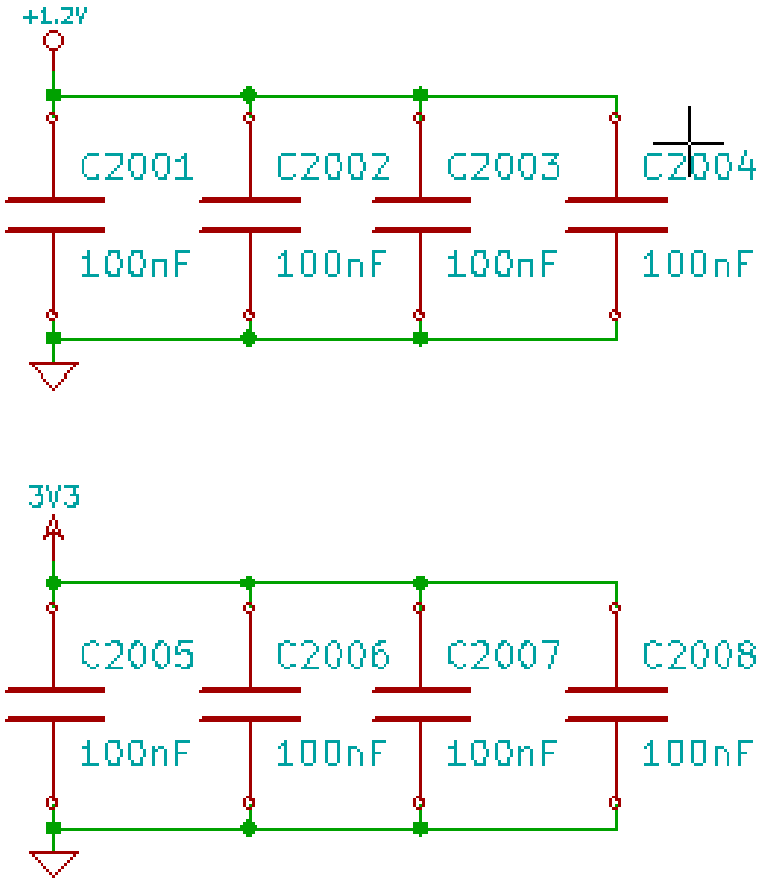
下記の図は、部品をシート２に配置し、「回路図中の最初の空き番号から使用する」オプションを利用してアノテーションを行ったものです。



「シートの Ref 番号を *100 から開始し、最初の空き番号から使用する」オプションを利用しアノテーションを行うと、下図のようになります。



「シートの Ref 番号を *1000 から開始し、最初の空き番号から使用する」オプションを利用した場合は、下図のようになります。



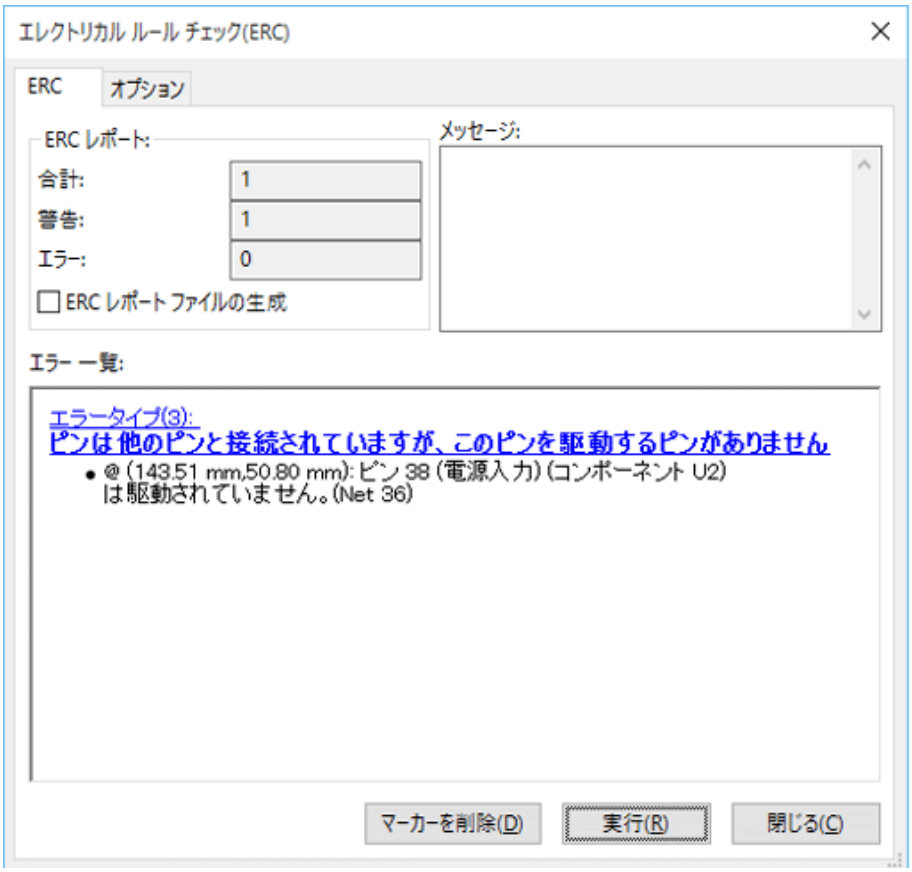
Chapter 8

ERC（エレクトリカル・ルール・チェック）による設計検証

8.1 はじめに

ERC （エレクトリカル・ルール・チェック）ツールは回路図の自動チェックを実行します。ERC は、未接続ピン、未接続の階層シンボル、出力のショートなどのようなシート内のすべてのエラーをチェックします。当然ながら、自動チェックは絶対確実なものではありませんし、設計エラーを検出可能なソフトウェアは１００%完全ではありません。しかし、そのようなチェックは多くの見落としや小さな間違いを検出可能なので非常に便利です。

次の工程へと進む前に、検出された全てのエラーをチェックして正常な状態となるよう修正する必要があります。ERC の質はライブラリ作成中に、ピンの電気的なプロパティをどれだけ細かく指定したかに依存します。ERC の出力は“エラー”または“警告”として報告されます。



8.2 ERC の使用法

アイコン  をクリックすると ERC を開始します。

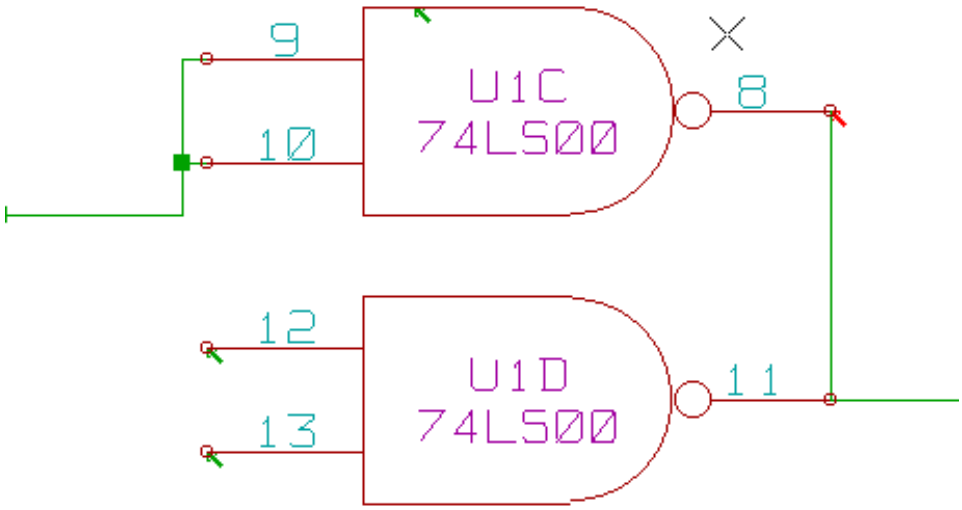
マーカーは、ERC エラーを出力した回路図の要素（ピンまたはラベル）上に配置されます。

注意

- “エレクトリカル・ルール・チェック (ERC)” ダイアログ内でエラーメッセージをクリックすると、回路図内の対応するマーカーに移動できます。
- 回路図中に表示されたマーカーを右クリックすることで、診断結果のメッセージへアクセスすることができます。

また、ダイアログからエラーのマーカーを削除できます。(一括で全て削除されます)

8.3 ERC の例



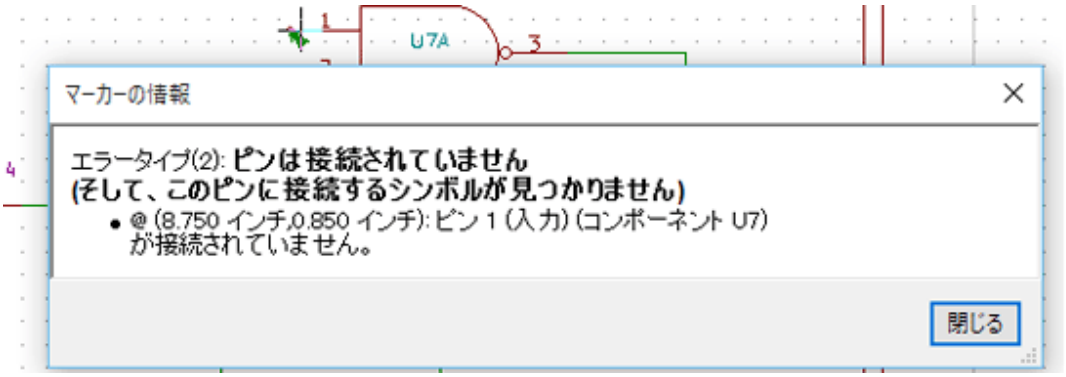
エラーが４つ見られます：

- ２本の出力が誤接続されています（赤の矢印)。
- 入力が２本未接続のままです（緑の矢印)。
- 非表示電源ポートのエラーで，電源フラグがありません（上部に緑の矢印)。

8.4 診断結果の表示

マーカーを右クリックし、コンテキストメニューを表示します。

ここで、“マーカーエラー情報”をクリックするとエラーの内容が“マーカーの情報”ウィンドウに表示されます。

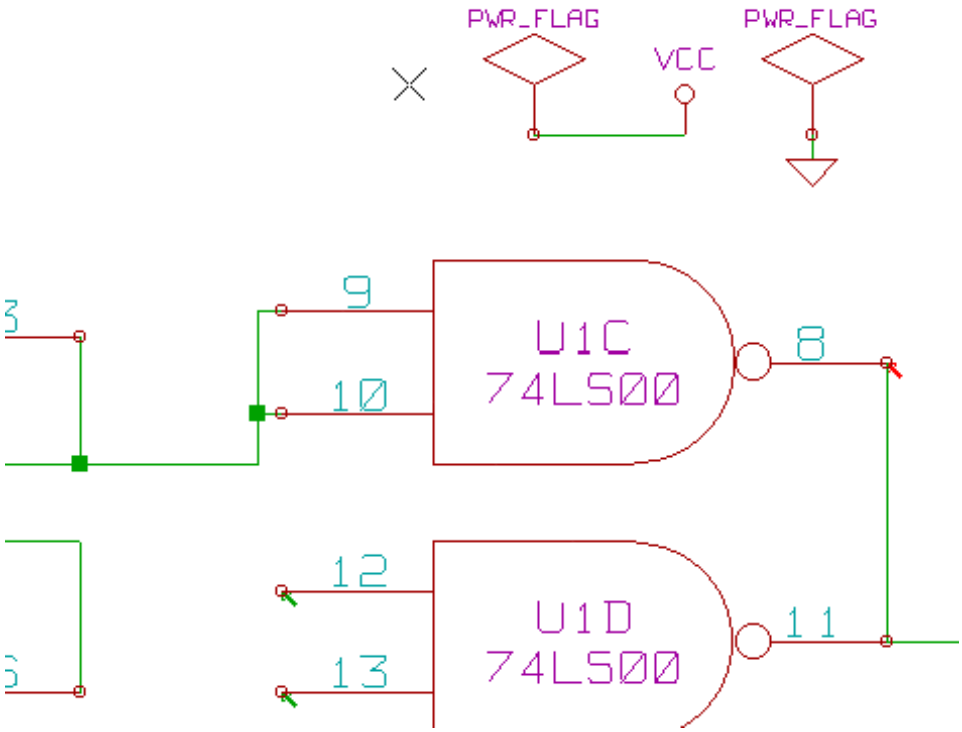


8.5 電源ピンと電源フラグ

電源ピンにエラーまたは警告を出すのは一般的です。たとえ、すべて正常のように思われるとしてもです。上の例を参照して下さい。大抵の設計では電源はコネクタによって供給されますが、そのコネクタは（電源出力として宣言されているレギュレータ出力のような）電源ではありません。

このため ERC は、電源出力ピンを検出してこの配線を操作するということはせず、電源で駆動されていないものと判断します。

この警告を避けるには、そのような電源ポートに“PWR_FLAG”を配置しなければなりません．次の例を参照して下さい。:



このようにすると、エラーマーカーが消えます。

大抵の場合、PWR_FLAG は GND に接続されていなければなりません。それは普通、レギュレータは電源出力として宣言された出力を持ちますが、グラウンドピンは電源出力ではなく（通常の属性は電源入力)、その結果、グラウンドは PWR_FLAG がなければ電源に接続されたことにはならないからです。

8.6 ルールの設定

オプションパネルで接続ルールを設定し、エラーおよび警告チェックのための電気的条件を定義します。

マトリクス内の変更したい四角をクリックすると，ノーマル，警告，エラーの選択が順番に切り替わり、ルールの変更が可能です。

8.7 ERC レポートファイル

オプションの ERC レポートの作成にチェックを付けると、ERC レポートファイルの生成と保存が可能です。ERC レポートのファイル拡張子は、.erc です。ERC レポートファイルの例を示します。

```
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
```

Chapter 9

ネットリストの作成

9.1 概要

ネットリストはコンポーネント間の電気的接続を記述したファイルです。ネットリストのファイルには，次のものが含まれます:

- コンポーネントのリスト。
- 等電位リストと呼ばれる、コンポーネント間の接続のリスト。

いろいろな異なるネットリストのフォーマットが存在します。コンポーネントのリストと等電位リストが２つの別々のファイルであることもあります。回路図入力 (capture) ソフトウェアにとって、このネットリストはなくてはならないものです。それはネットリストが、次のような他の電子系 CAD ソフトウェアとのリンクとなるからです。:

- PCB ソフトウェア。(基板設計ソフト)
- 回路および PCB シミュレータ。
- CPLD （および他のプログラマブル IC の）コンパイラ。

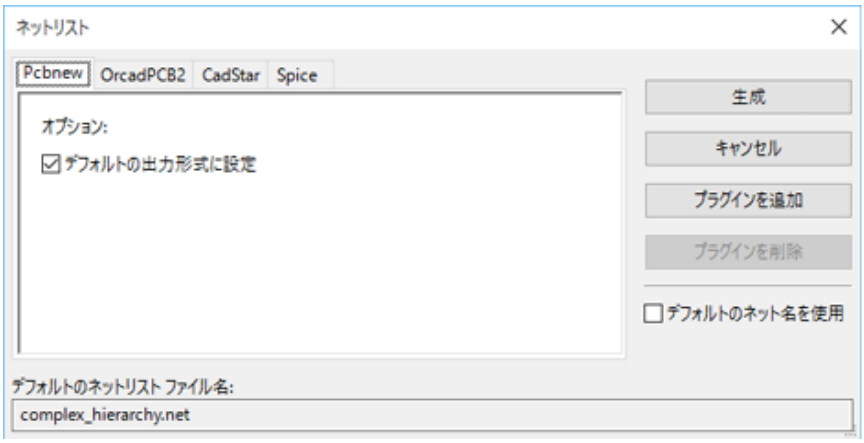
Eeschema はいくつかのネットリストのフォーマットをサポートしています。

- Pcbnew フォーマット（プリント配線)。
- OrCAD PCB2 フォーマット（プリント配線)。
- CADSTAR フォーマット（プリント配線)。
- 様々なシミュレータ用 Spice フォーマット（Spice フォーマットは他のシミュレータにも使用されます)。

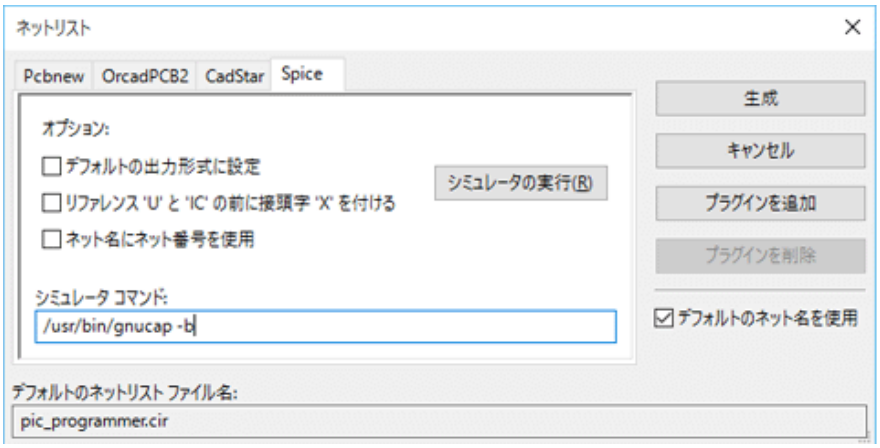
9.2 ネットリストフォーマット

ツールを選択し、ネットリスト作成ダイアログボックスを開きます。

Pcbnew を選択



Spice を選択

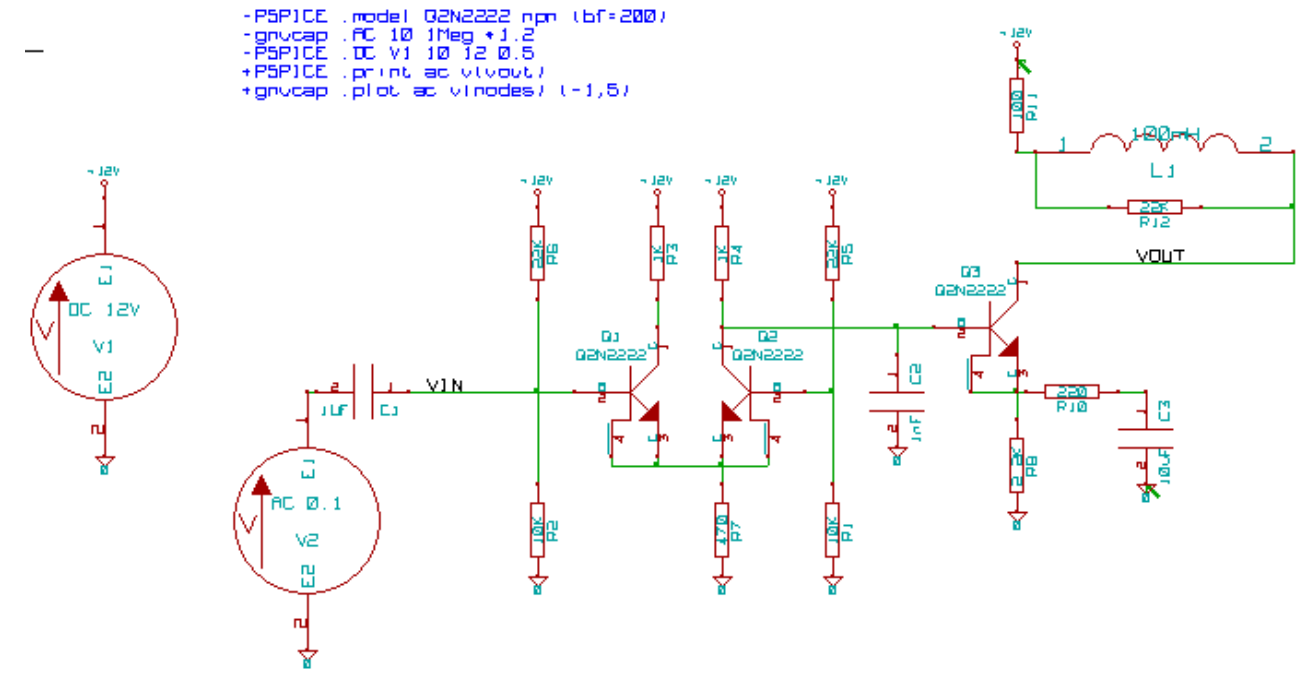


それぞれのタブで希望するフォーマットを選択できます。Spice フォーマットでは、等電位の名称（その方が読みやすい）か、またはネット番号（Spice の古いバージョンは番号のみ受け付ける）のどちらかでネットリストを生成することが可能です。ネットリストボタンをクリックすると、ネットリストファイルの名前を入力するよう促されます。

注意
大きなプロジェクトでは、ネットリストの生成に数分かかることがあります。

9.3 ネットリストの例

PSpice ライブラリを使用した回路設計は以下を参照して下さい。:



Pcbnew ネットリストファイルの例です。:

```
# Eeschema Netlist Version 1.0 generee 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}
(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
```

PSpice フォーマットでは，ネットリストは次のようになります。:

```
* 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 ネットリストについての注釈

9.4.1 ネットリスト名の注意事項

ネットリストを使用する多くのソフトウェアツールは，コンポーネント名，ピン名，等電位名（equipotentials）あるいは他の名前に空白文字（半角スペース (space) ）の使用を認めません。特に、ラベルあるいはコンポーネントやそのピンの名前と数値欄に空白を使用しないで下さい。

同様に、英数字以外の特殊文字の使用は問題を生じる可能性があります。この制限は Eeschema には無関係ですが、ネットリストを使用する他のソフトウェアが解釈できなくなるネットリスト・フォーマットとなることに注意して下さい。

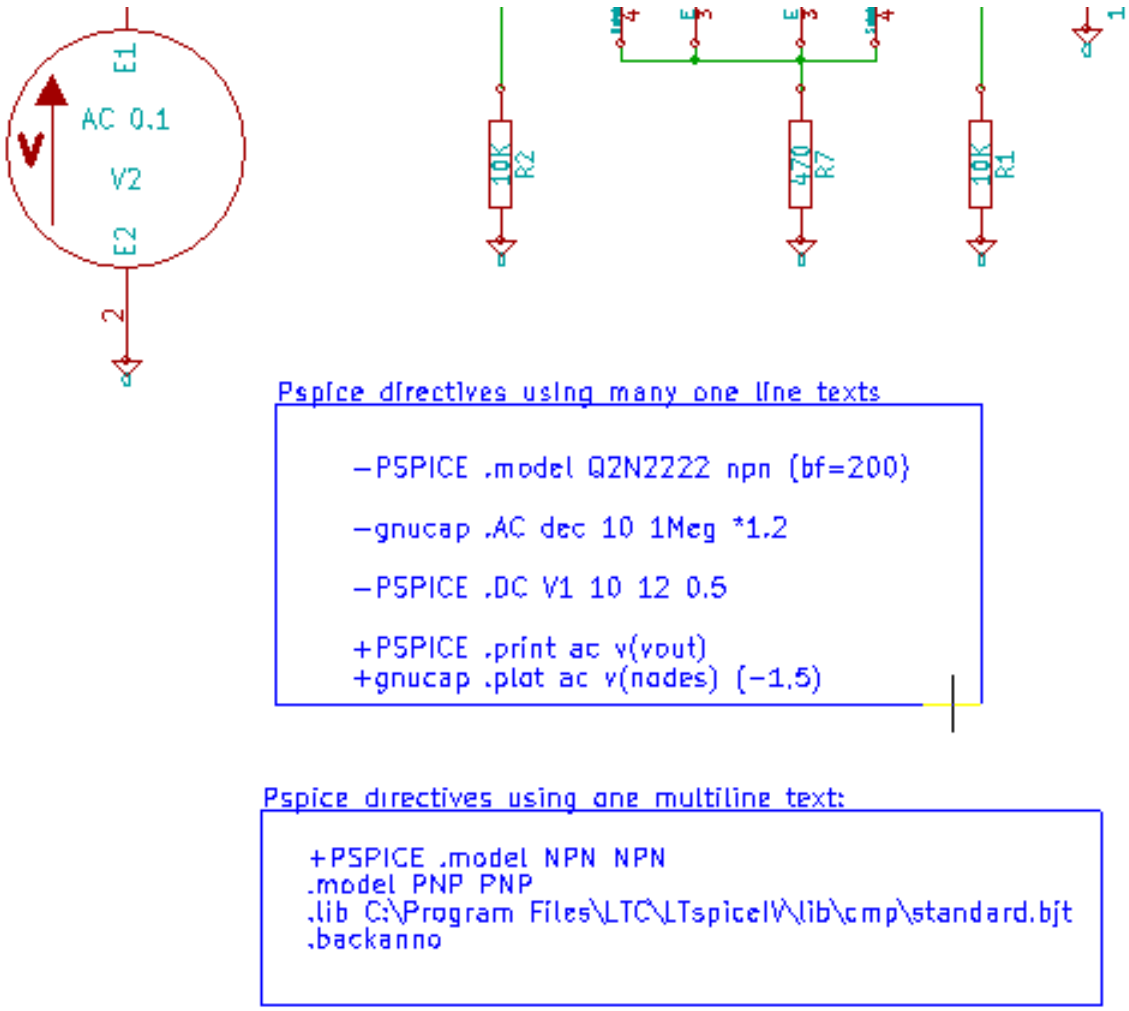
9.4.2 PSPICE ネットリスト

Pspice シミュレーターの場合、ネットリストの中にコマンド行 (.PROBE, .AC など) をいくつか含める必要があります。

回路図に含まれる **-pspice** または **-gnucap** のキーワードで始まるテキスト行は、ネットリストの先頭に（キーワードがない状態で）挿入されます。

回路図に含まれる **+pspice** または **+gnucap** のキーワードで始まるテキスト行は、ネットリストの最後に（キーワードがない状態で）挿入されます。

１行テキストを数回使用した例と、複数行テキストを１つ使用した例です。



例えば、次のようなテキストを入力する場合（ラベルを使用しないこと !）:

```
-PSPICE .PROBE
```

PROBE の行はネットリストに挿入されます。前述の例ではこの方法でネットリストの先頭に３行、末尾に２行挿入されました。

複数行テキストを使用している場合、**+pspice** または **+gnucap** のキーワードは１度だけ必要です:

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

４行生成されます:

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

また Pspice の場合、等電位の GND は０（ゼロ）という名前にしなければならないことに注意して下さい。

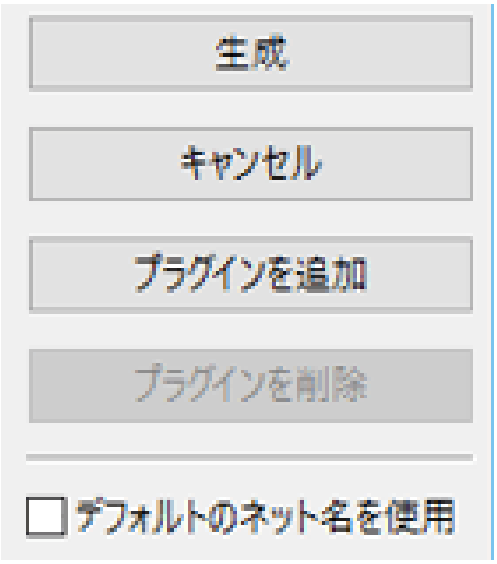
9.5 他のフォーマット

他のネットリスト・フォーマットの場合には、プラグインの形でネットリストコンバーターを追加することが可能です。Eeschema はそれらのコンバーターを自動的に起動します。コンバーターの解説と例は１４章にあります。

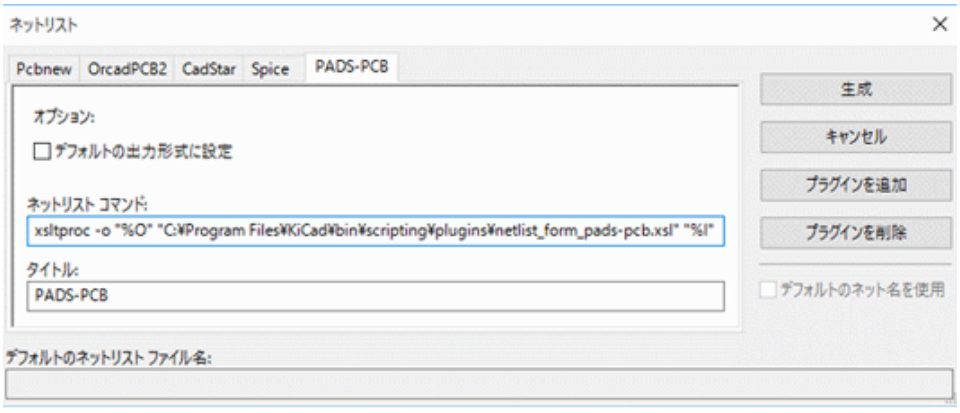
コンバーターはテキストファイル（xsl フォーマット）ですが，Python のような他の言語を使用することもできます。xsl フォーマットを使用する場合、ツール（xsltproc.exe あるいは xsltproc ）は Eeschema が生成した中間ファイルとコンバーターファイルを読み込んで、出力ファイルを生成します。この場合、コンバーターファイル（シートスタイル）は非常に小さく記述が容易です。

9.5.1 ダイアログウィンドウの初期設定

プラグイン追加ボタンで、新規ネットリスト・プラグインを追加することが可能です。



PadsPcb プラグインのセットアップウィンドウです。:



セットアップには以下が必要です:

- タイトル（例えば、ネットリストフォーマットの名前)。
- 起動するプラグイン。

ネットリスト生成時に、以下のことが行われます:

1. Eeschema は中間ファイル *.tmp を生成します。例えば、test.tmp とします。
2. Eeschema はプラグインを実行し、test.tmp を読み込んで test.net を生成します。

9.5.2 コマンドライン・フォーマット

xsltproc.exe を.xsl ファイルの変換ツールとして、ファイル netlist_form_pads-pcb.xsl をコンバーターのシートスタイルとして使用する例です:

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

各部の意味は次の通りです:

f:/kicad/bin/xsltproc.exe	A tool to read and convert xsl file
-o %O.net	Output file: %O will define the output file.
f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl	File name converter (a sheet style, xsl format).
%I	Will be replaced by the intermediate file created by Eeschema (*.tmp).

test.sch という名前の回路図の場合，実際のコマンドラインは次の通りです:

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

9.5.3 コンバーターとシートスタイル（プラグイン）

これは非常に単純なソフトウェアです。なぜなら、その目的が入力テキストファイル（中間テキストファイル）を別のテキストファイルに変換するだけだからです。さらに、中間テキストファイルから BOM リストの生成が可能です。

xsltproc を変換ツールとして使用すると、シートスタイルのみが生成されます。

9.5.4 中間ネットリスト・ファイル・フォーマット

xslproc についてのさらに多くの説明、中間ファイルフォーマットの記述内容、各コンバーターの場合のシートスタイルの例は１４章を参照して下さい。

Chapter 10

プロットと印刷

10.1 はじめに

上部メニューバーの“ファイル”から“印刷 (N)”と“プロット (P)”の両コマンドが実行可能です。

プロットでサポートしている出力フォーマットは、PostScript、PDF、SVG、DXF、HPGL です。自分のプリンターで直接印刷することも可能です。

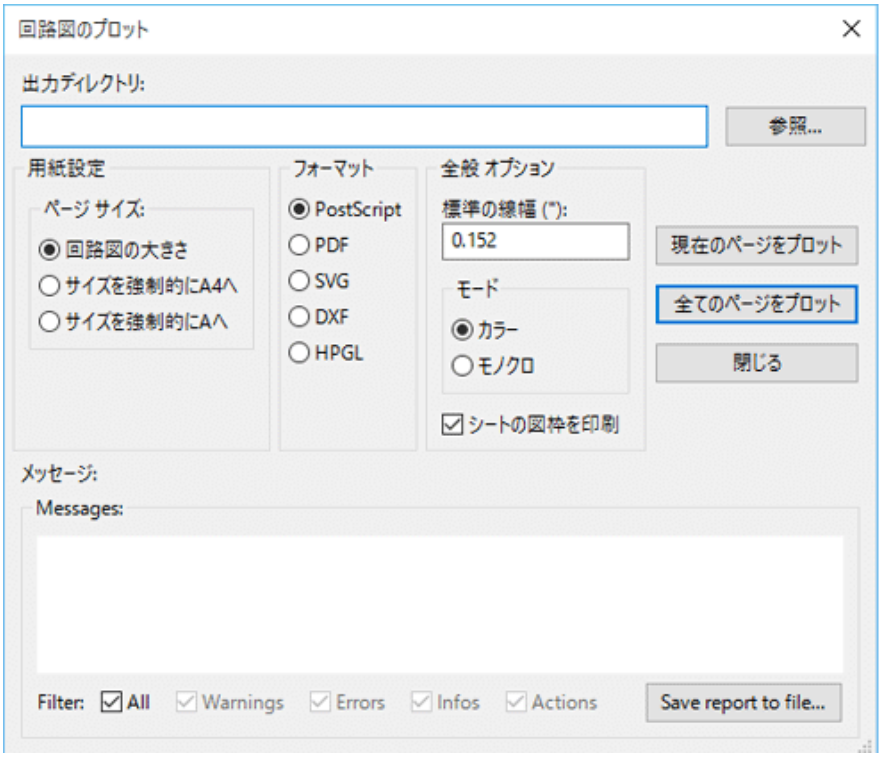
10.2 プロットの共通コマンド

現在のページをプロット
: 現在のシートのみプロットします。

全てのページをプロット
: 全階層をプロットします（各シート毎に印刷ファイルを 1 つ生成する）。

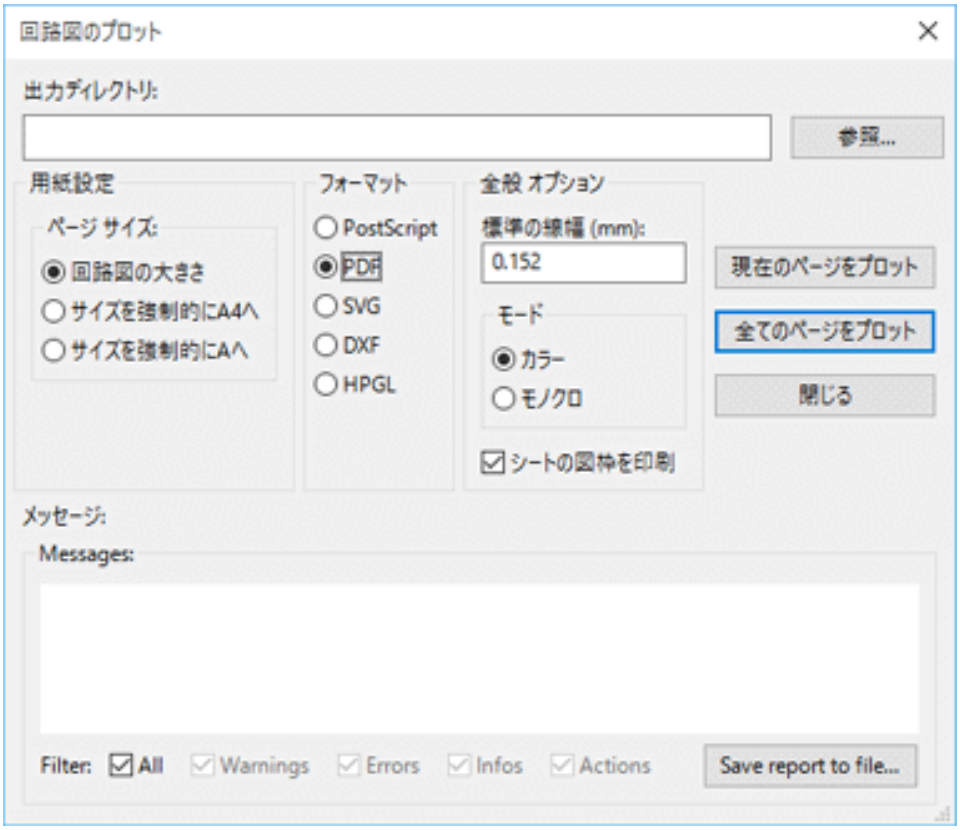
10.3 Postscript のプロット

“回路図のプロット” 内のフォーマットで PostScript オプションを指定します。



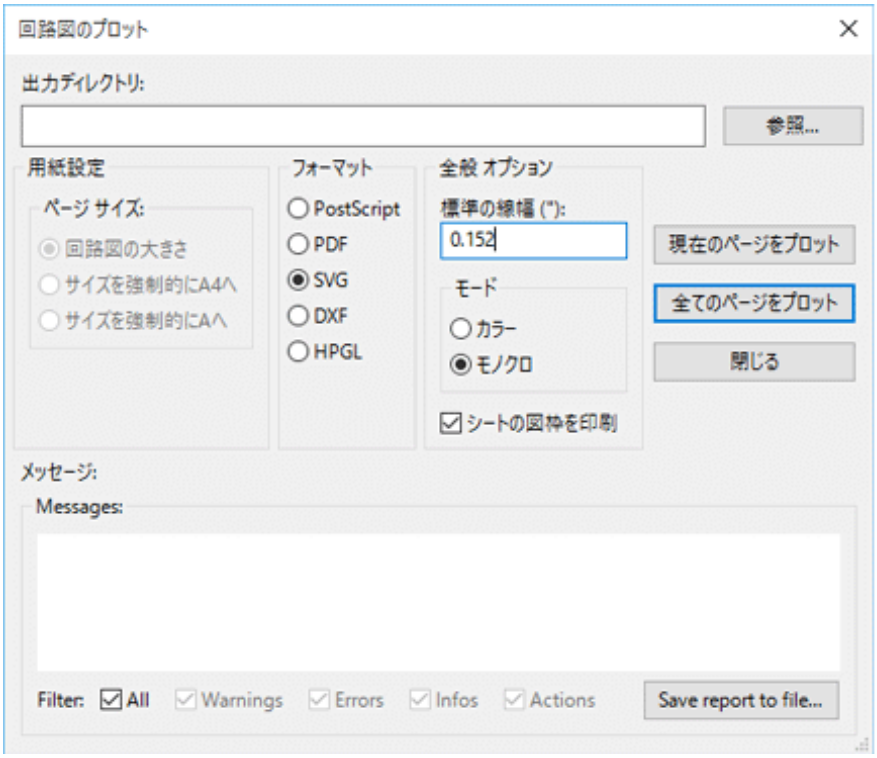
ファイル名はシート名に拡張子.ps を付加したものになります。“シートの図枠を印刷” オプションを無効にすることが可能です。これは文書編集ソフトウェアで図を挿入するためにしばしば使用されるカプセル化ポストスクリプトファイル（.eps フォーマット）を生成する場合に便利です。メッセージウィンドウは生成されたファイル名を表示します。

10.4 PDF のプロット



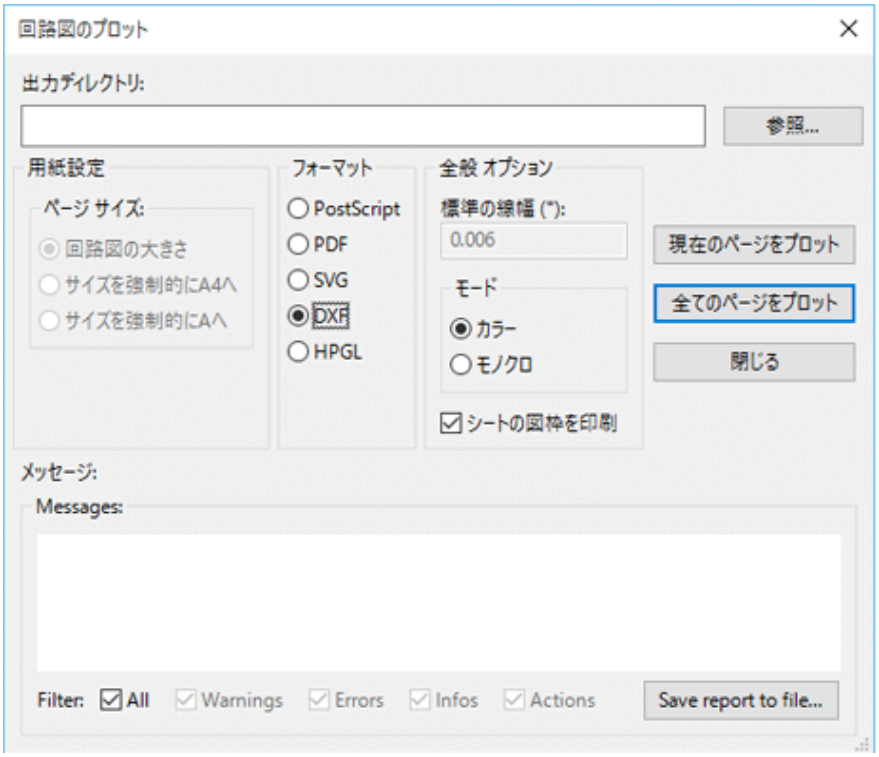
“回路図のプロット” 内のフォーマットで PDF オプションを指定します。ファイル名はシート名に拡張子.pdf を付加したものになります。

10.5 SVG のプロット



“回路図のプロット” 内のフォーマットで SVG オプションを指定します。ファイル名はシート名に拡張子.svg を付加したものになります。

10.6 DXF のプロット



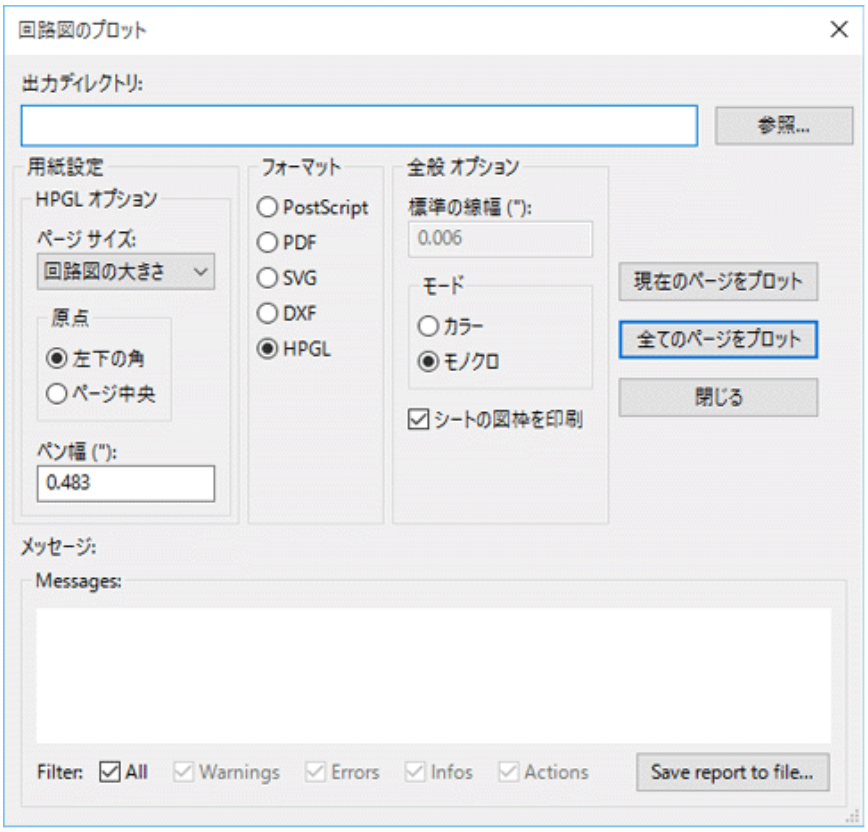
“回路図のプロット” 内のフォーマットで DXF オプションを指定します。ファイル名はシート名に拡張子.dxf を付加したものになります。

10.7 HPGL のプロット

“回路図のプロット” 内のフォーマットで HPGL オプションを指定します。このフォーマットでは、以下を定義可能です。

- ページサイズ。
- 原点。
- ペン幅 (mm 単位)。

以下は “回路図のプロット” ダイアログです。(上部メニューバーから、ファイル → プロット → プロットを選択します。):



出力ファイル名はシート名に拡張子.plt を付加したものです。

10.7.1 シートサイズ選択

通常は“回路図の大きさ”が選択されています。この場合、ページ設定で定義されているサイズが使用され、その時のスケールは 1 になります。異なるサイズ（A4 ～A0 あるいは A ～E ）を選択すると、スケールが自動的に調整されてページにフィットします。

10.7.2 オフセット調整

全ての標準的な寸法に対して、可能な限り正確に中央に描画するようにオフセットを調整できます。プロッターは原点がシートの中央か左下角にあるので、適切にプロットするためにはオフセットを挿入する必要があります。

一般的に言えることですが、

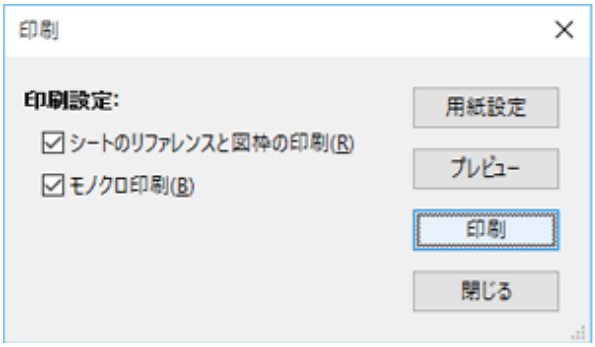
- シートの中央を原点とするプロッターの場合、オフセットは負の値で、シート寸法の 1/2 に設定しなければなりません。
- シートの左下角を原点とするプロッターの場合、オフセットは 0 に設定しなければなりません。

オフセットを設定するには次のことを行います。

- シートサイズを選択。
- 原点を選択。
- (“左下の角”または“ページ中央”を選択)

10.8 紙面印刷

上部メニューバーから“ファイル”→ “印刷 (N)”を選ぶか、上部ツールバーのアイコン  により、印刷プレビューや普通のプリンターでの印刷ができます。



“シートのリファレンスと図枠の印刷”オプションは、シートのリファレンスおよびタイトルブロックの有効／無効を切り替えます。

“モノクロ印刷”オプションは印刷をモノクロ印刷に最適化します。通常、このオプションはモノクロのレーザープリンターを使用する場合に必要です。それは、カラーがハーフトーンで印刷されて非常に読みにくくなることがよくあるからです。

Chapter 11

コンポーネント・ライブラリ・エディタ

11.1 コンポーネント・ライブラリに関する一般情報

コンポーネントは、図形表現、電気的接続、フィールド定義を含んだ図面要素です。回路図で使用するコンポーネントはすべてコンポーネントライブラリに保存されています。Eeschema は、ライブラリの作成、ライブラリへのコンポーネントの追加／削除、ライブラリ間でのコンポーネントの移動、ファイルへのコンポーネントのエクスポート、ファイルからのコンポーネントのインポートを行うためのコンポーネントライブラリ編集ツールを提供します。コンポーネント・ライブラリ・エディタはコンポーネント・ライブラリ・ファイルを管理するシンプルな方法を提供します。

11.2 コンポーネント・ライブラリの概要

コンポーネントライブラリは一つ以上のコンポーネントから構成されます。一般的に、コンポーネントは機能、種類、または製造者によってグループ分けされます。

コンポーネントは以下のものから構成されます。:

- グラフィカルな型式 (design) （ライン、円、テキストフィールド)。
- ピンは、図形的なプロパティ（線（通常のピン)、クロックピン、反転ピン、Low レベルアクティブピンなど）と ERC（エレクトリカル・ルール・チェック）ツールが使用する電気的なプロパティ（入力、出力、双方向など）の両方を持っています。
- リファレンス、値、PCB 設計用のフットプリント名などのようなテキストフィールド。
- エイリアスは、7400 のような通常のコンポーネントと共に、74LS00、74HC00、7437 のようなその全ての派生品で使用されます。これら全てのエイリアスは同じライブラリコンポーネントを共有します。

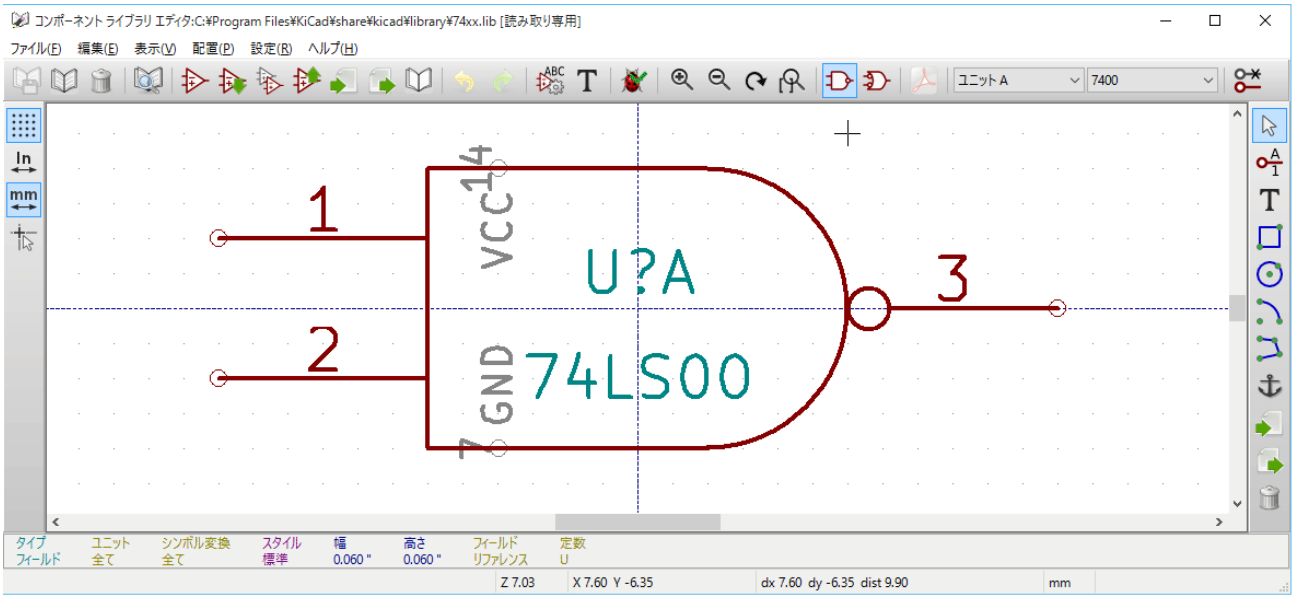
適切なコンポーネント設計には以下のことが要求されます。:

- コンポーネントは一つ以上のパーツで構成されるように定義する。

- コンポーネントがド・モルガン表現として知られる代替ボディスタイルを持つように定義する。
- ライン、矩形、円、ポリゴンおよびテキストを使用して回路図シンボル（記号）を設計する。
- グラフィック要素、名前、ピン数、電気的プロパティ（入力、出力、３ステート、電源ポートなど）を慎重に定義して、ピンを追加する。
- 他のコンポーネントの型式とピン配置が同じである場合、エイリアスを追加する。あるいは他のコンポーネントから新たにコンポーネントを作成した場合、エイリアスを削除する。
- PCB 設計ソフトウェアが使用するフットプリント名のような補助的フィールドを追加し、それらの可視性を定義する。
- データシートへの www リンクや説明文などを付加してコンポーネントに記録する。
- 適切な (desired) ライブラリにそれを保存する。

11.3 コンポーネント・ライブラリ・エディタの概要

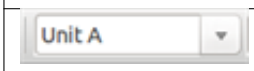
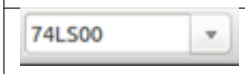
コンポーネント・ライブラリ・エディタのメインウィンドウを以下に示します。よく使う機能を手早く使えるようにするための３つのツールバーとコンポーネントの閲覧／編集エリアで構成されます。ツールバーからは全てのコマンドを使うことができませんが、メニューからは使用できます。



11.3.1 メイン・ツールバー

基本的にメインウィンドウの最上部にあるメインツールバー（以下に示します）は、ライブラリ管理ツール、取り消し／やり直しコマンド、ズームコマンドとコンポーネントプロパティから成ります。

	Save the currently selected library. The button will be disabled if no library is currently selected or no changes to the currently selected library have been made.
	Select the library to edit.
	Delete a component from the currently selected library or any library defined by the project if no library is currently selected.
	Open the component library browser to select the library and component to edit.
	Create a new component.
	Load component from currently selected library for editing.
	Create a new component from the currently loaded component.
	Save the current component changes in memory. The library file is not changed.
	Import one component from a file.
	Export the current component to a file.
	Create a new library file containing the current component. Note: new libraries are not automatically added to the project.
	Undo last edit.
	Redo last undo.
	Edit the current component properties.
	Edit the fields of current component.
	Test the current component for design errors.
	Zoom in.
	Zoom out.
	Refresh display.
	Zoom to fit component in display.

	Select the normal body style. The button is disabled if the current component does not have an alternate body style.
	Select the alternate body style. The button is disabled if the current component does not have an alternate body style.
	Show the associated documentation. The button will be disabled if no documentation is defined for the current component.
	Select the unit to display. The drop down control will be disabled if the current component is not derived from multiple units.
	Select the alias. The drop down control will be disabled if the current component does not have any aliases.
	Pin editing: independent editing for pin shape and position for components with multiple units and alternate symbols.
	Show pin table.

11.3.2 エレメント・ツールバー

基本的にメインウィンドウの右側に位置する縦のツールバーで、コンポーネント設計で必要とされる全ての要素を画面へ配置できます。ツールバーの各ボタンの定義は下表のとおりです。

	Select tool. Right-clicking with the select tool opens the context menu for the object under the cursor. Left-clicking with the select tool displays the attributes of the object under the cursor in the message panel at the bottom of the main window. Double-left-clicking with the select tool will open the properties dialog for the object under the cursor.
	Pin tool. Left-click to add a new pin.
	Graphical text tool. Left-click to add a new graphical text item.
	Rectangle tool. Left-click to begin drawing the first corner of a graphical rectangle. Left-click again to place the opposite corner of the rectangle.
	Circle tool. Left-click to begin drawing a new graphical circle from the center. Left-click again to define the radius of the cicle.
	Arc tool. Left-click to begin drawing a new graphical arc item from the center. Left-click again to define the first arc end point. Left-click again to define the second arc end point.

	Polygon tool. Left-click to begin drawing a new graphical polygon item in the current component. Left-click for each addition polygon line. Double-left-click to complete the polygon.
	Anchor tool. Left-click to set the anchor position of the component.
	Import a component from a file.
	Export the current component to a file.
	Delete tool. Left-click to delete an object from the current component.

11.3.3 オプション・ツールバー

基本的にメインウィンドウの左側に位置する縦のツールバーで、エディタの描画オプションのいくつかを設定できます。ツールバーの各ボタンの定義は下表のとおりです。

	Toggle grid visibility on and off.
	Set units to inches.
	Set units to millimeters.
	Toggle full screen cursor on and off.

11.4 ライブラリの選択および保守

メイン・ツールバーのアイコン  で現在のライブラリを選択します。クリックすると使用可能な全てのライブラリが表示され、作業対象のライブラリを選択することができます。コンポーネントの読込み／保存は、このライブラリ（現在のライブラリ）に対して行われます。コンポーネントのライブラリ名は、自身の定数フィールドの値となります。

注意

- ライブラリに含まれるコンポーネントの読込み／保存を行うためには、Eeschema でライブラリを読み込まなければなりません。
- 現在のライブラリに対して行った変更は、作業終了後にメインツールバーの  をクリックして、ハードディスクへ保存することが可能です。
- コンポーネントを現在のライブラリから削除するには  をクリックします。

11.4.1 コンポーネントの選択および保存

コンポーネントの編集時には、実際にハードディスクのライブラリ内のコンポーネントに対して作業しているのではなく、ローカルのメモリ内にあるそのコピーに対して作業しています。従って、どのような編集作業も容易に元に戻すことが可能です。また、コンポーネントはローカルのライブラリから読み込んだり、あるいは既存のコンポーネントから作成することもできます。

11.4.1.1 コンポーネントの選択

メインツールバーのアイコン  をクリックすると、利用可能なコンポーネントのリストが表示されます。そのコンポーネントは選択および読み込みが可能です。

注意
コンポーネントのエイリアスを選択すると、選択したエイリアスに代わって実際に読み込んだコンポーネントの名前が常にウィンドウのタイトルに表示されます。（メインのコンポーネントが読み込まれます。）コンポーネントのエイリアスのリストは常に各コンポーネントと共に読み込まれ、編集することができます。 

から現在のコンポーネントのエイリアスを選択することで、新しいコンポーネントを作ることができます。リストの最初の項目はルートコンポーネントです。

注意

もう一つの方法として、エクスポートコマンド  で前回保存したコンポーネントを、インポートコマンド  により読み込むことができます。

11.4.1.2 コンポーネントの保存

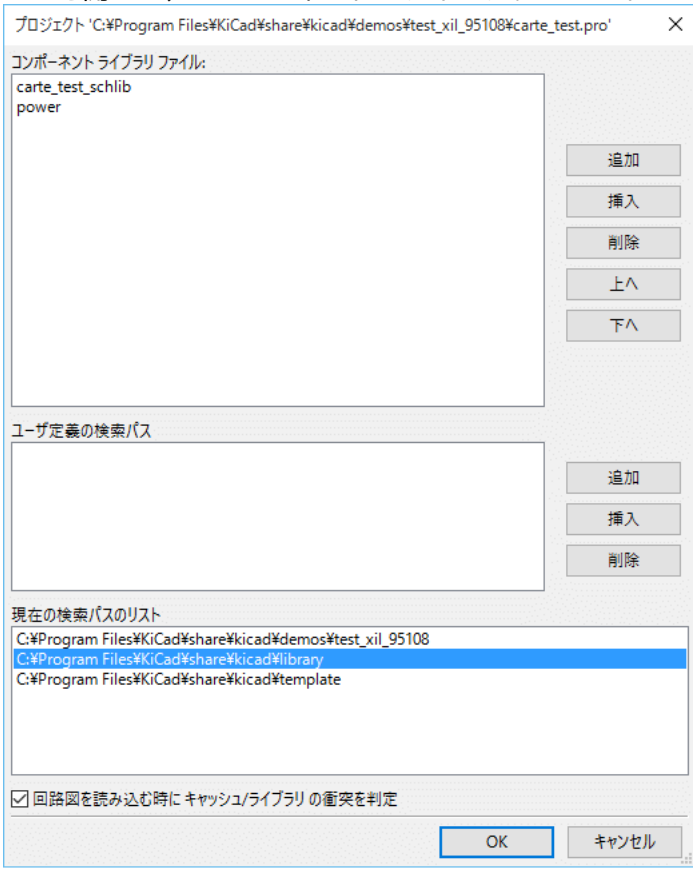
変更後、コンポーネントを現在のライブラリまたは新規ライブラリへ保存することが可能です。あるいはバックアップファイルにエクスポートことも可能です。

現在のライブラリに保存するには、更新コマンド  を使用します。更新コマンドはローカルメモリ内にコンポーネントを保存するだけであるということを覚えておいて下さい。

コンポーネントを永続的に保存したい場合は、保存アイコン  を使用しなければなりません。それは実際のハードディスク上のライブラリファイルに変更を加えます。

このコンポーネントで新規ライブラリを作成したい場合、NewLib コマンド  を使用して下さい。またその時には、新規ライブラリ名が必要となります。

注意
新しいライブラリは自動的に現在のプロジェクトへは追加されません。
回路図で使用したい新しいライブラリはすべて Eeschema の“設定”→“コンポーネントライブラリ”でダイアログを開いて、コンポーネント・ライブラリ・ファイルのリストへ追加しなければなりません。



エクスポートコマンド  を使用して、そのコンポーネントだけを含むファイルを作成することが可能です。このファイルはコンポーネントを一つだけ含む標準ライブラリファイルになります。また、このファイルは別のライブラリへコンポーネントをインポートするのに使うことができます。実際、NewLib コマンドとエクスポートコマンドは基本的に同じものです。

11.4.1.3 ライブラリ間のコンポーネントの移動

コピー元のライブラリからコピー先のライブラリに簡単にコンポーネントをコピーすることが可能です。それには次のコマンドを使用します:

- アイコン  をクリックしてコピー元のライブラリを選択します。
-  ボタンで移動するコンポーネントを読み込みます。(そのコンポーネントが表示されます。)
-  ボタンでコピー先のライブラリを選択します。
-  ボタンで現在のコンポーネントをローカルメモリに保存します。
-  ボタンでコンポーネントをハードディスク（コピー先のライブラリ）に保存する。

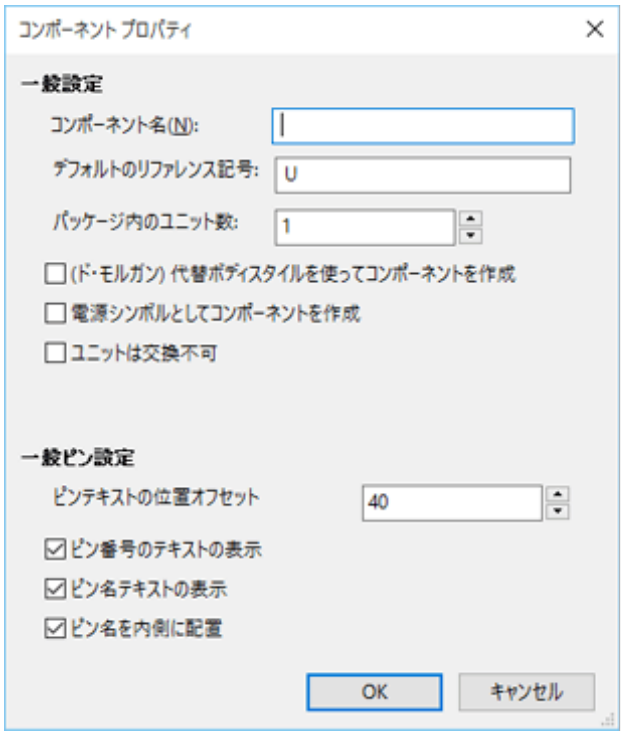
11.4.1.4 コンポーネントの編集の取り消し

あるコンポーネントに対して作業している時、編集中のコンポーネントというのは、ライブラリ内にある実際のコンポーネントの作業コピーです。このため、ローカルのメモリ内にそのコンポーネントを保存しない限り、再読み込みをすることで行ったすべての変更を取り消すことができます。ローカルのメモリ内に既に保存していたとしても、ハードディスク上のライブラリファイルに保存していない場合には、Eeschema を終了／再起動して、全ての変更を元に戻すことが常に可能です。

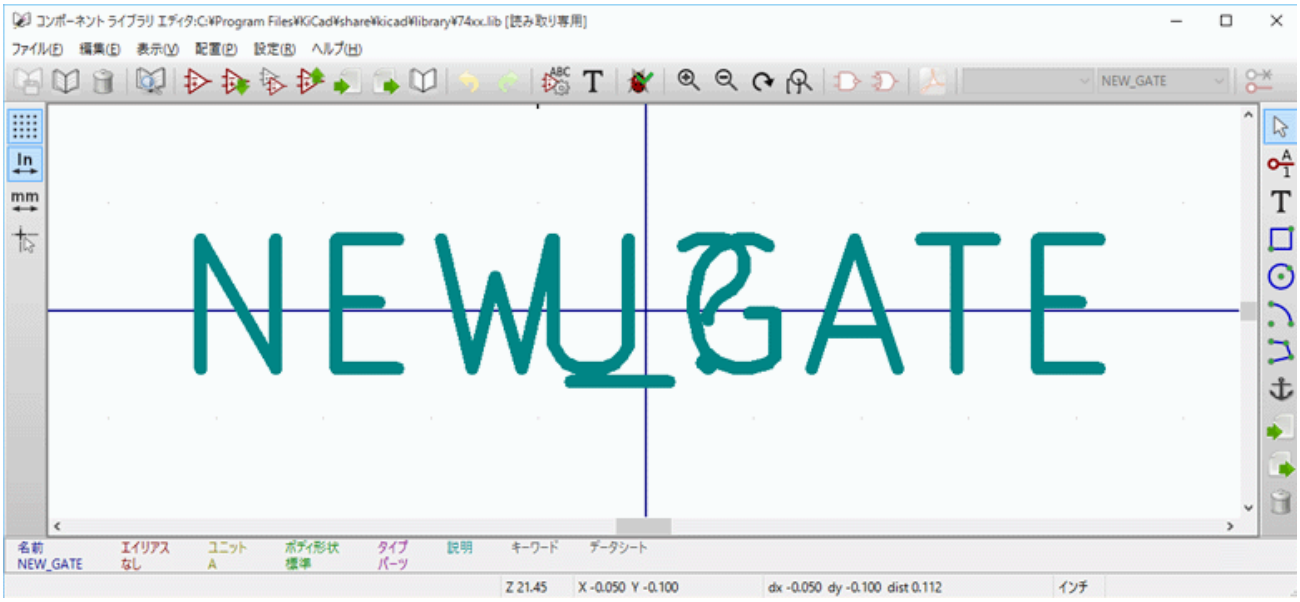
11.5 ライブラリコンポーネントの作成

11.5.1 新規コンポーネントの作成

 ボタンの NewPart コマンドを使用して新規コンポーネントの作成が可能です。コンポーネント名（名前は回路図エディタで定数フィールドのデフォルト値として使用される)、リファレンス（U、IC、R …)、パッケージ内のパーツ数（例えば、標準コンポーネントの 7400 は１つのパッケージ内に４つのパーツを持っている)、(ド・モルガンと呼ぶこともある）代替ボディスタイルが存在するかどうかの入力が要求されます。もしリファレンスフィールドが空白のままであると、デフォルトの“U “になります。これらのプロパティはすべて後で設定することが可能ですが、コンポーネントの作成時に設定した方がいいでしょう。



新しくコンポーネントを作成すると、コンポーネント・ライブラリ・エディタの画面上に次のように現れます。



11.5.2 他のコンポーネントからコンポーネントを作成

作成したいと思っているコンポーネントが KiCad のライブラリにあるものと似ているということがしばしばあります。この場合、既存のコンポーネントを読み込んで変更するのがごく普通です。

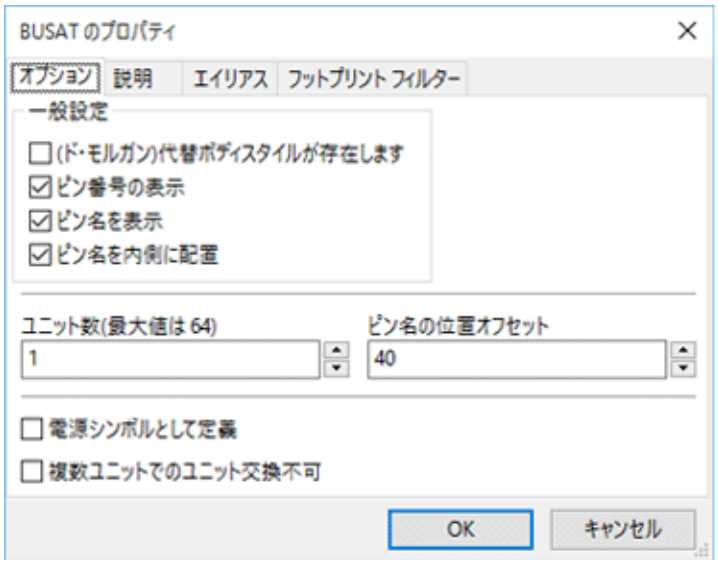
- 出発点として使うコンポーネントを読み込みます。
- アイコン  をクリックするかまたは名前を右クリックしてコンポーネント名のテキストを編集し、その名前

を変更します。もし、あなたが変更しようとした名前が既に使われていた場合、そのコンポーネントを上書きして新しく作成するコンポーネントと交換するかどうか確認を求められます。

- 元の型となったコンポーネントにエイリアスがある場合、新規コンポーネントから衝突するエイリアスを削除するよう促されます。その答えが NO の場合、新規コンポーネントの作成は中止されます。コンポーネントライブラリは重複した名前またはエイリアスを持つことができません。
- 必要に応じて新規コンポーネントを編集します。
-  ボタンでメモリに新規コンポーネントを保存するか、または  ボタンで新規ライブラリに保存します。あるいは他の既存のライブラリに新規コンポーネントを保存したい場合には、コマンド  で保存したいライブラリを選択して、新規コンポーネントを保存します。
-  ボタンの “ディスクにライブラリを保存” コマンドでハードディスクにライブラリファイルを保存します。

11.5.3 コンポーネントプロパティの編集

コンポーネントの新規作成時や既存のコンポーネントから継承してコンポーネントを作成する時には、コンポーネントのプロパティは注意深く追加すべきです。コンポーネントのプロパティを変更するには、“コンポーネントプロパティの編集” コマンド  をクリックします。次にコンポーネントプロパティのダイアログを示します。

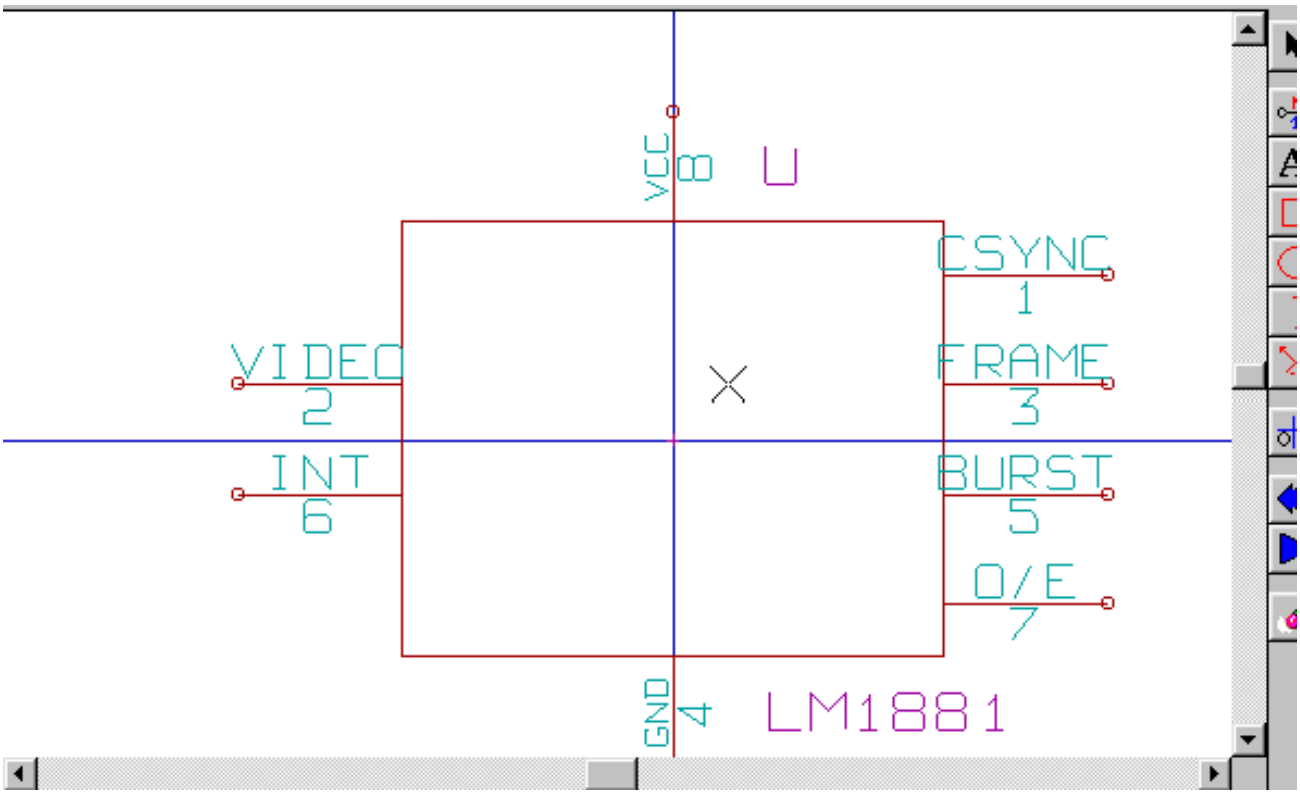


パッケージあたりのパーツ数を正しく設定することは大変重要です。またコンポーネントが代替シンボル表現を定義するパラメータを正確に持つなら、ピンが編集または作成される時、各パーツに相当するピンは正しく作成されるでしょう。ピンの作成／編集後にパッケージあたりのパーツ数を増やすと、新しいパーツへのピンとシンボルの追加という余計な仕事をする破目になるでしょう。例えそうだとしても、これらのパラメータの変更はいつでも可能です。

グラフィックオプション” ピン番号の表示” と” ピン名を表示” は、ピン番号とピン名のテキストの可視性を定義します。対応するオプションがチェックされている場合、そのテキストが表示されます。オプション” ピン名を内

側に配置”はピン名の位置を定義します。オプションがチェックされている場合、そのテキストはコンポーネントアウトラインの内側に表示されます。この場合、”ピン名の位置オフセット”プロパティは内側方向へのテキストの変位量を定義します。値は (1/1000 インチ単位で) 30 〜40 が妥当です。

以下の例は、”ピン名を内側に配置”オプションにチェックをつけない状態のコンポーネントを示しています。ピン名とピン番号の位置に注意して下さい。

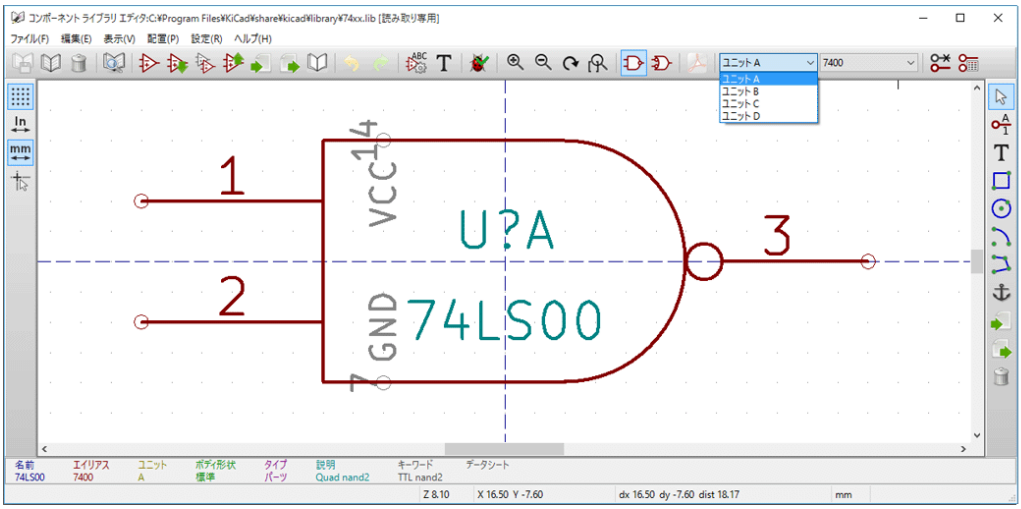


11.5.4 代替シンボルをもつコンポーネント

もしコンポーネントが一つ以上の代替シンボル表現を持つなら、それらを編集するためにコンポーネントの別のシンボルを選択しなければならないでしょう。通常のシンボルを編集するには、  をクリックします。

代替シンボルを編集するには、  をクリックします。編集対象のユニットを選ぶには 

を使って、ドロップダウンリストから選択します。次の画面に例を示します。



11.6 グラフィック要素

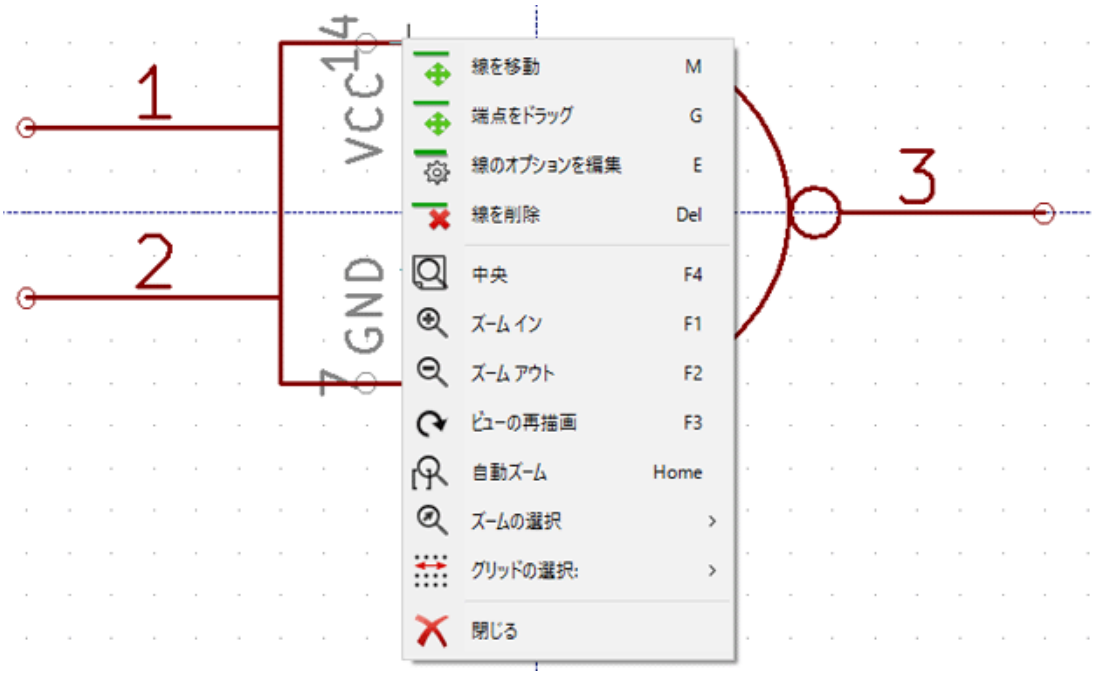
グラフィック要素はコンポーネントの図形シンボルを表し、電気的接続の情報を含みません。この設計には次のツールが使用可能です。:

- 始点と終点で定義されたラインとポリゴン。
- 対角線にある２つのコーナーで定義された矩形。
- 中心と半径で定義された円。
- 始点と終点および中心で定義された円弧。(円弧は 0° から 180° まで描画できます。)

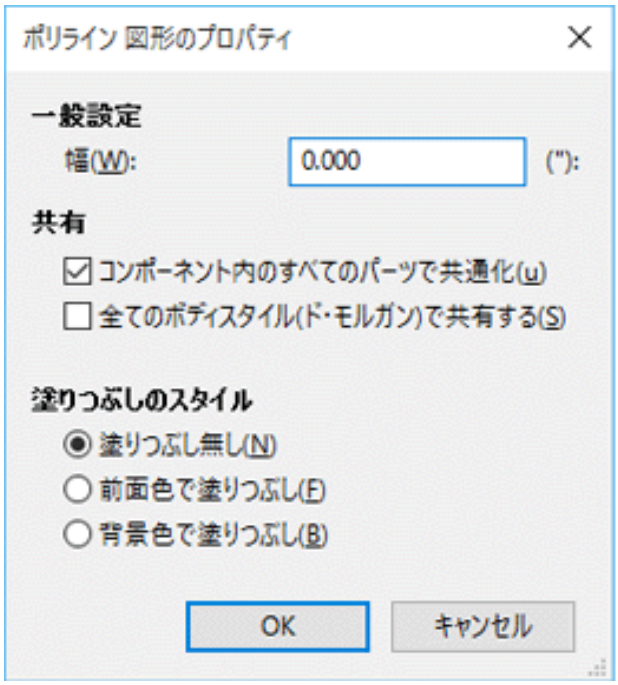
メインウィンドウの右側にあるエレメント・ツールバーで、コンポーネントの図形シンボル設計に必要な全てのグラフィック要素を配置できます。

11.6.1 グラフィック要素のメンバーシップ。

個々のグラフィック要素（ライン、円弧、円など）は、全てのパーツで共通／全てのボディスタイルで共通／パーツ毎／ボディスタイル毎に定義することが可能です。要素オプションは、選択した要素を右クリックしてコンテキストメニューを表示することで簡単に設定できます。次にラインのコンテキストメニューの例を示します。



またこの要素を左ダブルクリックすると要素のプロパティを変更できます。次に多角形のプロパティダイアログの例を示します。



グラフィック要素のプロパティは次の通りです。:

- “一般設定”の“幅 (W)”は、現在表示しているパーツにおけるライン要素の幅を定義します。
- “共有”の”コンポーネントの全てのパーツで共通”設定は、パッケージに一つ以上のパーツを持つコンポーネントの各パーツ全てへグラフィック要素を描く、または現在のパーツにのみグラフィック要素を描く、のどちらかを定義します。

- “共有”の”全てのボディスタイル (ド・モルガン) で共有する (S)”設定は、代替ボディスタイルを持つコンポーネントの各ボディスタイルへ全てグラフィック要素を描く、または現在のボディスタイルにのみグラフィック要素を描く、のどちらかを定義します。
- “塗つぶしのスタイル”は、グラフィック要素で定義されるシンボルに対して、“塗つぶし無し (N)”／“前面色で塗りつぶし (F)”／“背景色で塗りつぶし (B)”を何れかを選択します。

11.6.2 グラフィックタイプのテキスト要素

アイコン T により、グラフィックタイプのテキストを作成がすることができます。グラフィックタイプのテキストはコンポーネントが反転 (mirrored) しても影響されず常に読むことができます。グラフィックタイプのテキストはフィールドではないことに注意しましょう。

11.7 多パーツコンポーネントと代替ボディスタイル

コンポーネントは２つのシンボル表現（標準シンボルとド・モルガンと呼ばれる代替シンボル）を持つことができ、パッケージ（例えば論理ゲート）毎に１つ以上のパーツを持っています。いくつかのコンポーネントは、異なったシンボル表現と異なるピン定義を持つパーツを１つのパッケージ内に複数持っています。

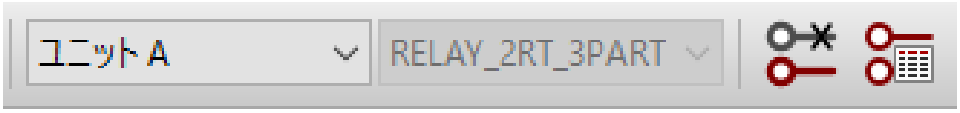
例として、３つの別のパーツ（コイル、スイッチ 1、スイッチ 2）として表現することが可能な２つのスイッチを持つリレーを考えてみましょう。多パーツコンポーネントと代替ボディスタイルを持つコンポーネントの設計はとても柔軟に行えます。ピンまたはシンボルは、それぞれのパーツに共通または特定のパーツのみ、もしくは両方のボディスタイルに共通または特定のボディスタイルのみ、に適用できます。

デフォルトでは、ピンはそれぞれのパーツのそれぞれのシンボル表現に固有です。それは、ピンの数は各パーツで異なり、シンボル表現に依存した形状となるからです。ピンが各パーツまたは各シンボル表現で共通の場合、全てのパーツとシンボル表現に対して一度作成するだけで済みます (これは電源ピンの場合には一般的です)。これは、それぞれのパーツで共通であるようなボディスタイルのグラフィックとテキストの場合も同様です（しかし、普通は各シンボル表現で固有です)。

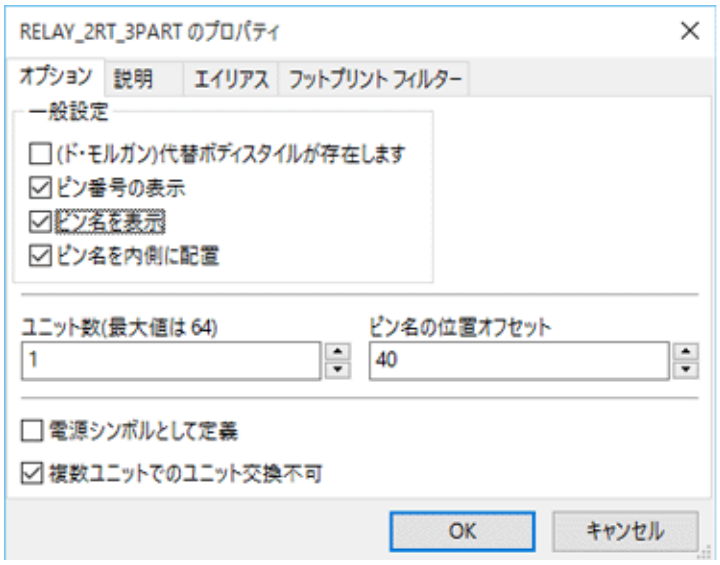
11.7.1 異なったシンボル表現を持つ多パーツコンポーネントの例:

これはパッケージ毎に３つのパーツ（スイッチ 1、スイッチ 2、コイル）で定義されるリレーの例です。:

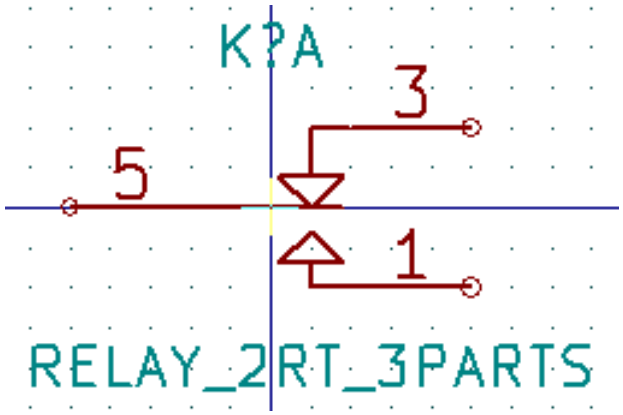
オプション: ピンはリンクしていない。別のパーツのいかなるピンとも同期せず、各パーツごとに単独でピンを追加/編集できる。



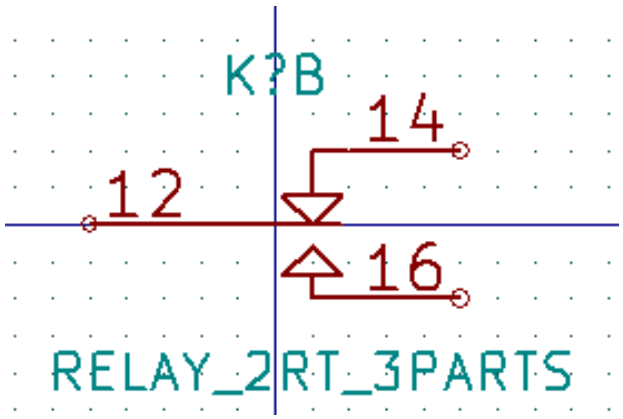
選択されている全てのパーツは入れ替えできない。



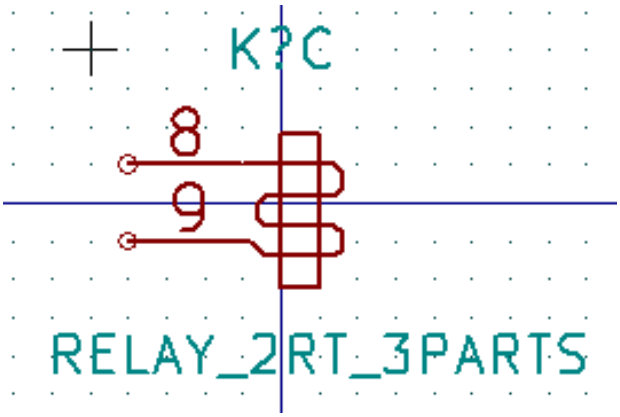
パーツ 1



パーツ 2



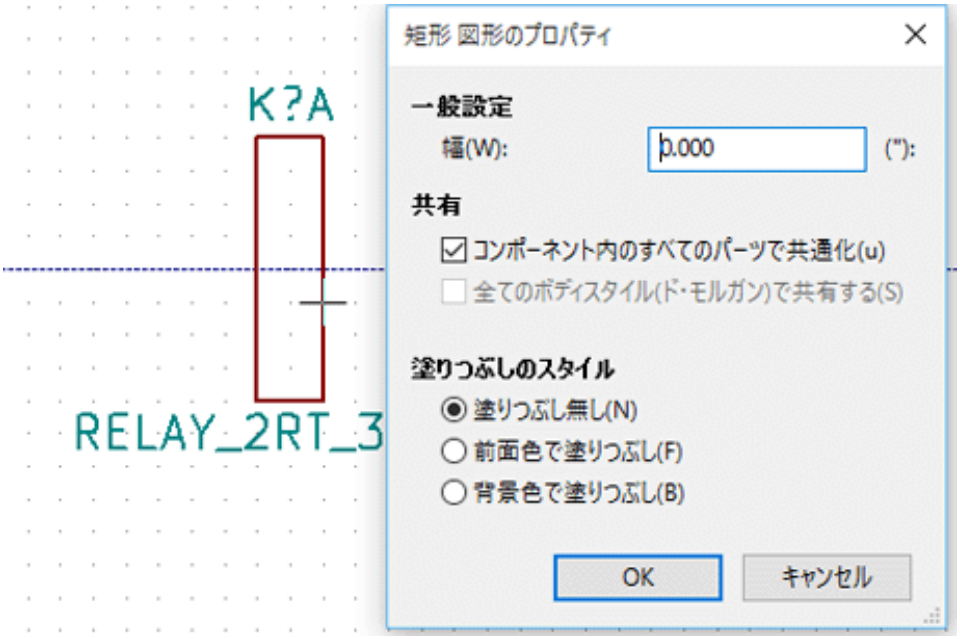
パーツ 3



同じシンボルとピン配置を持たないので、ユニット 1 とユニット 2 は入れ替えできません。

11.7.1.1 グラフィックシンボル要素

次にグラフィックシンボル要素のプロパティを示します。上のリレーの例ですが、３つのパーツは異なったシンボル表現を持っています。そのため、各パーツは個別に作成され、グラフィックシンボル要素は”コンポーネントの全てのパーツで共通”（のプロパティ）を無効としなければなりません。



11.8 ピンの作成および編集

アイコン  を左クリックしてピンの作成や挿入が可能です。全てのピンのプロパティ編集はピンを左ダブルクリックして行います。もう一つの方法としては、ピンを右クリックしてピンのコンテキストメニューを開きます。ピンは慎重に作成しなければなりません。それは、どのようなエラーも基板（PCB）設計に影響するからです。すでに配置済みの任意のピンは再編集、削除または移動が可能です。

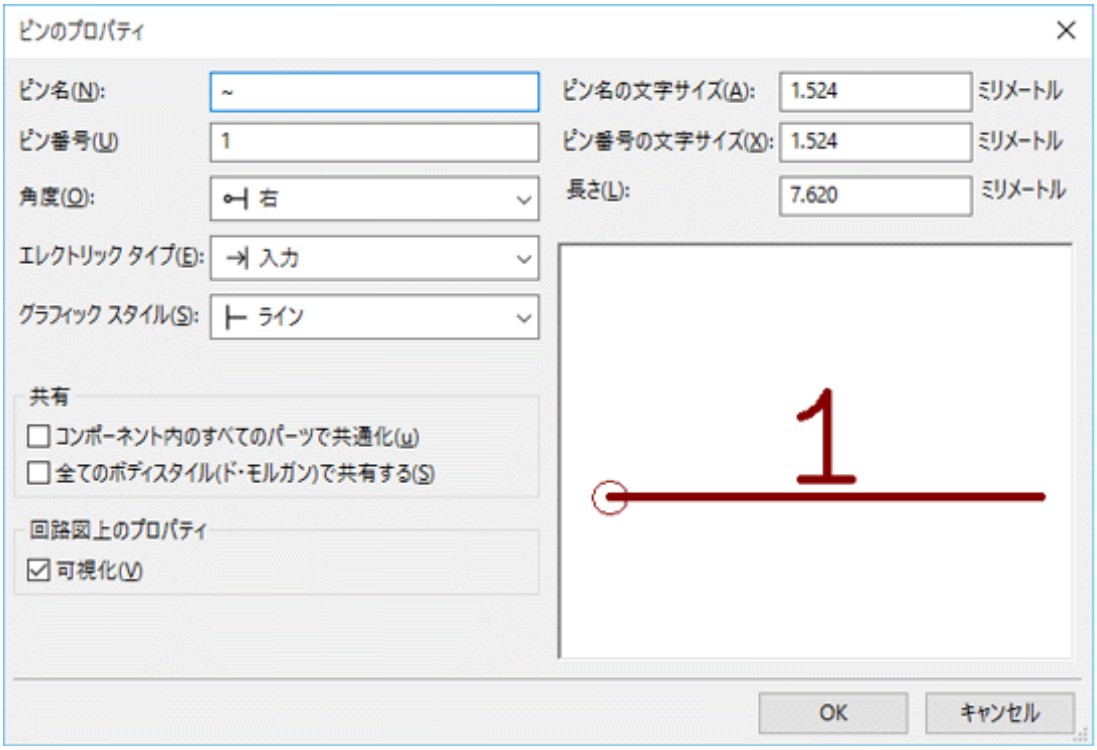
11.8.1 ピンの概要

ピンはその形状（長さ、グラフィックな外観)、名前および“番号“で定義されます。“番号“は４つまでの英文字または数字で定義されます。(番号は常に数字であるとは限りません。) ERC （エレクトリカル・ルール・チェック）ツールを機能させるためには、“電気的“タイプ（入力、出力、３ステート…）も定義されていなければなりません。このタイプが正しく定義されていない場合には、ERC チェックが非効率になります。(正しい結果を表示できません。)

重要な注意

- ピン名とピン番号に空白（半角スペース (space) ）を使用しないで下さい。
- 反転信号（上付き横線（overline）付信号）のピン名はチルダ記号 ~ で始めます。次の ~ 記号は上付き横線（overline）を無効にします。例えば、`\~FO~O` は FO O と表示されます。
- ピン名の文字数を減らしてこの信号のシンボルだけになった場合は、そのピンには名前がないと見なされます。
- # で始まるピン名は電源ポート用に予約されています。
- ピン“番号”は１～４文字の英数字から成ります。1、2、…、9999 は有効な数字ですが、A1、B3 …(標準的な PGA の表記法)、あるいは Anod、Gnd、Wine なども有効です。
- コンポーネントで重複したピン“番号“を使うことはできません。

11.8.2 ピンのプロパティ

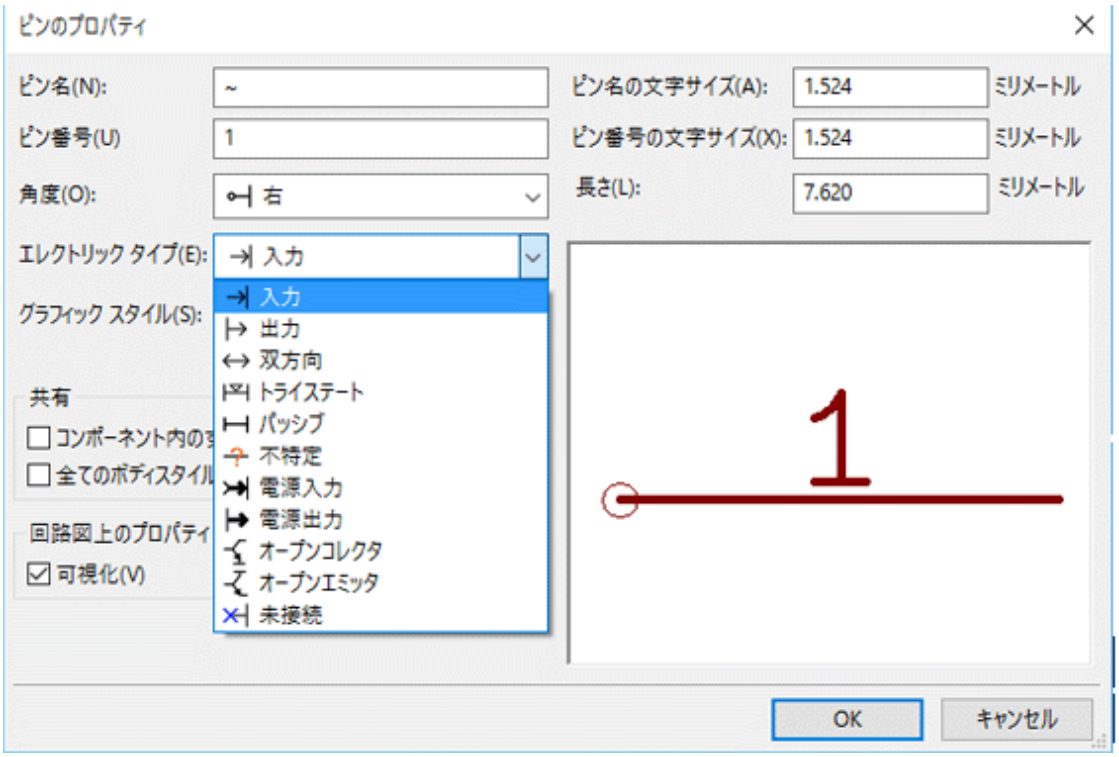


ピンプロパティダイアログでピンの全ての特性を編集することができます。このダイアログは、ピンを作成したり、あるいは既存のピンをダブルクリックすると、自動的にポップアップします。このメニューで以下の定義または変更が可能です。:

- “ピン名 (N)” と “ピン名の文字サイズ (A)”
- “ピン番号 (U)” と “ピン番号の文字サイズ (X)”。
- ピンの “角度 (O)” と “長さ (L)”。
- “エレクトリックタイプ (E)” および “グラフィックスタイル (S)”。
- “コンポーネントの全てのパーツで共通” および “全てのボディスタイル（ド・モルガン）で共有する (S)”
- “可視化 (V)”。（電源ピンで使用される。）

11.8.3 ピンの形状（グラフィックスタイル）

それぞれのピンの形状を下図に示します。形状の選択は単にグラフィック上の影響があるだけで、電気的タイプへは影響を与えません。(ERC チェックあるいはネットリスト機能には何の影響もありません。)



11.8.4 ピンの電気的タイプ

電気的タイプの選択を正しく行うことは ERC ツールとって重要です。それぞれの電気的タイプを下図に示します。:

- 双方向。入力および出力を切り替え可能な双方向ピン（例えばマイクロプロセッサのデータバス）を表します。
- トライステート。通常の 3 ステート (H, L, Hi-Z) 出力です。
- パッシブ。抵抗、コネクタなどの受動コンポーネントのピンに使用されます。
- 未指定。ERC チェックを必要としない場合に使用できます。

- 電源入力。コンポーネントの電源ピンに使用されます。自動的に同じタイプの同じ名前 (非表示電源ピン) の他のピンに接続されます。
- 電源出力。電圧レギュレータの出力に使用されます。
- オープンエミッタとオープンコレクタ。このように定義された論理回路の出力に使用できます。
- 未接続。コンポーネント内部で接続されていないピンに使用されます。

11.8.5 ピンのグローバルプロパティ

すべてのピンの長さ、あるいはテキストサイズ（名前、パーツ番号）を変更可能です。右クリックでポップアップされるピンのコンテキストメニューのグローバルコマンドを使用してこれら３つのパラメータの一つを設定します。変更したいパラメータをクリックし、新しい値を入力します。その値は現在の表現のすべてのコンポーネントピンに適用されます。

11.8.6 多パーツコンポーネントおよび代替シンボル表現におけるピンの定義

ピンの作成、編集を行う場合、多パーツコンポーネントおよび代替シンボル表現をもつコンポーネントは特に問題を含んでいます。ピンは個別のパーツ（ピン番号はパーツごとに指定される）と個別のシンボル表現（形状と位置はシンボル表現ごとに指定される）で指定されます。コンポーネントライブラリエディタは同時にこれらのピンの作成を行えます。デフォルトでは、ピンへの変更は、多パーツコンポーネントの全てのパーツと代替シンボル表現をもつコンポーネントの両方の表現に対して行われます。

これに対して、ピンの形状（グラフィックスタイル）と名前は例外です。この依存関係は多くの場合、ピンの作成、編集を容易にします。また、この依存関係はメインツールバーの  で無効にできます（トグル動作です）。これにより完全に独立して、各パーツと個別のシンボル表現に対してピンを作成することができます。

コンポーネントは２つのシンボル表現（ド・モルガンとして知られる表現）を持つことができ、ロジックゲートのように複数のパーツから構成することができます。あるコンポーネントでは、いくつかの異なったグラフィック要素やピンを必要とすることもあるでしょう。セクション 11.7.1 にあるリレーのサンプルのように、リレーは３つの別のパーツ（コイル、スイッチ接点１、スイッチ接点２）で表現することができます。

多パーツコンポーネントと代替シンボル表現を持つコンポーネントの管理はとても柔軟に行えます。ピンは、全パーツで共通または各パーツで固有に設定できます。ピンはまた、両方のシンボル表現で共通または各シンボル表現で固有に設定できます。

デフォルトでは、ピンは各パーツのそれぞれの表現に固有です。それは、ピンの数は各パーツで異なり、それら(パーツ）のデザインは各シンボル表現で異なるからです。全てのパーツでピンが共通の場合は、（電源ピンのように）単に一度作成するだけで済みます。

例えば、４つの２入力 NAND ゲート 7400 の出力ピンを考えてみましょう。４つの入力と２つのシンボル表現があるので、コンポーネントの定義では８個の出力ピンを別々に定義します。新しい 7400 コンポーネントが作られると、通常シンボル表現のパーツＡがライブラリエディタに表示されるでしょう。別のシンボル表現のピン形状（スタイル）を編集するには、まずツールバーのボタン  をクリックして有効にしなければなりません。各パーツのピン番号を編集するには、ドロップダウン 

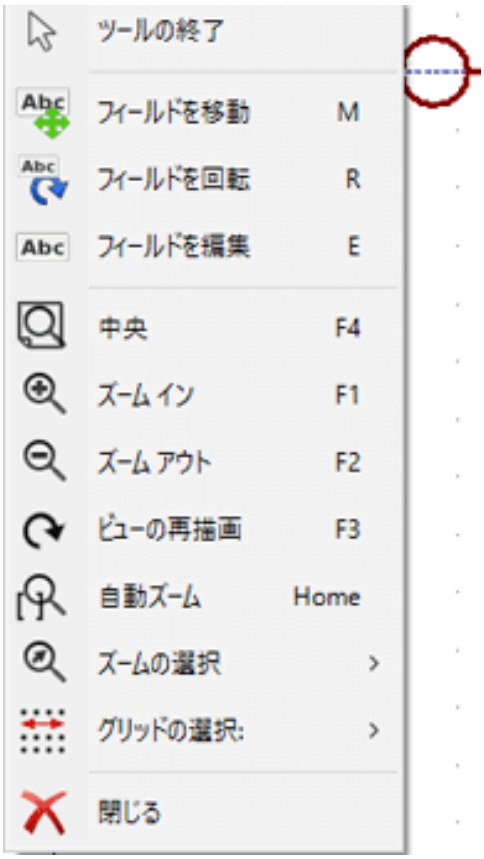
を使ってパーツを選択します。

11.9 コンポーネントのフィールド

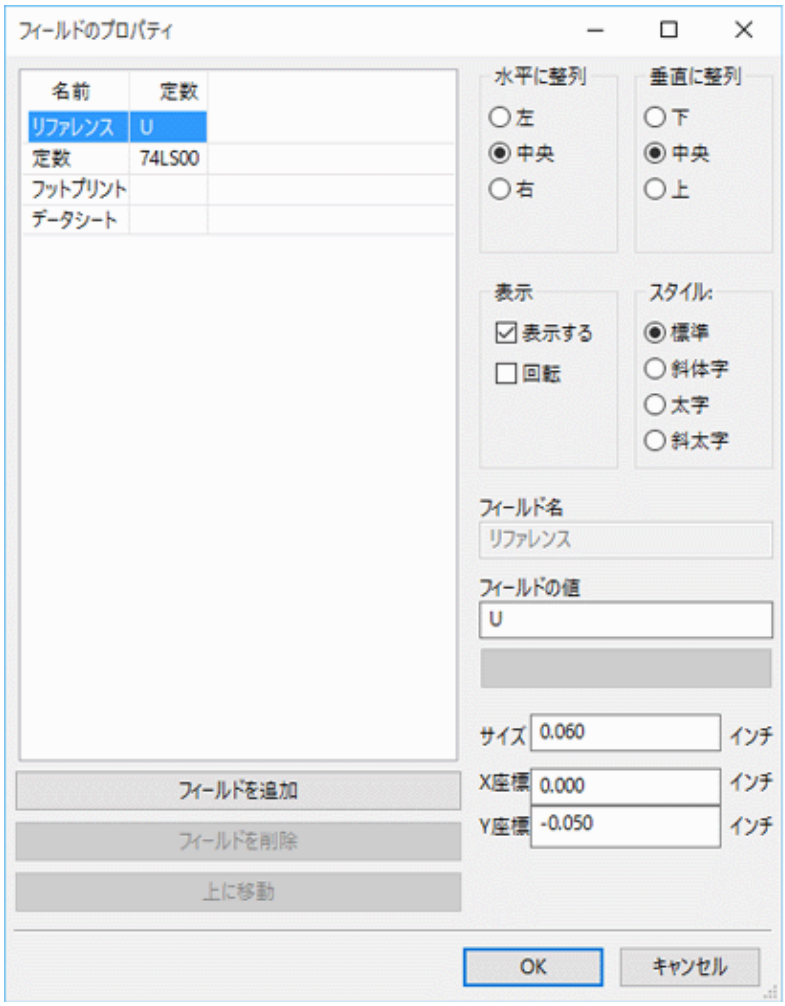
全てのライブラリコンポーネントには４つのデフォルトフィールドが定義されています。リファレンス、定数、フットプリント、データシートのフィールドは、コンポーネントが作られるか、コピーされると必ず生成されます。リファレンス、定数のフィールドだけは、必須項目です。既存のフィールドに対しては、ピンを右クリックすることでコンテキストメニューのコマンドを使うことができます。ライブラリにあるコンポーネントは、通常、この４つのデフォルトフィールドで定義されています。製造業者名、品番、価格などのような追加フィールドをライブラリコンポーネントへ追加できますが、一般的にこれは、追加フィールドを回路図にある全てのコンポーネントに適用できるよう回路図エディタで行われます。

11.9.1 コンポーネントフィールドの編集

既存のフィールドを編集するには、フィールドテキストを右クリックして次のフィールド・コンテキストメニューを表示します。



未定義フィールドの編集、新しいフィールドの追加、(追加された) オプションフィールドの削除を行うには、メインツールバーの T で、次のコンポーネントプロパティ・ダイアログを開きます。



フィールドはコンポーネントに関連したテキストの区画 (section) です。コンポーネントのグラフィック表示に属するテキストと混同すべきではありません。

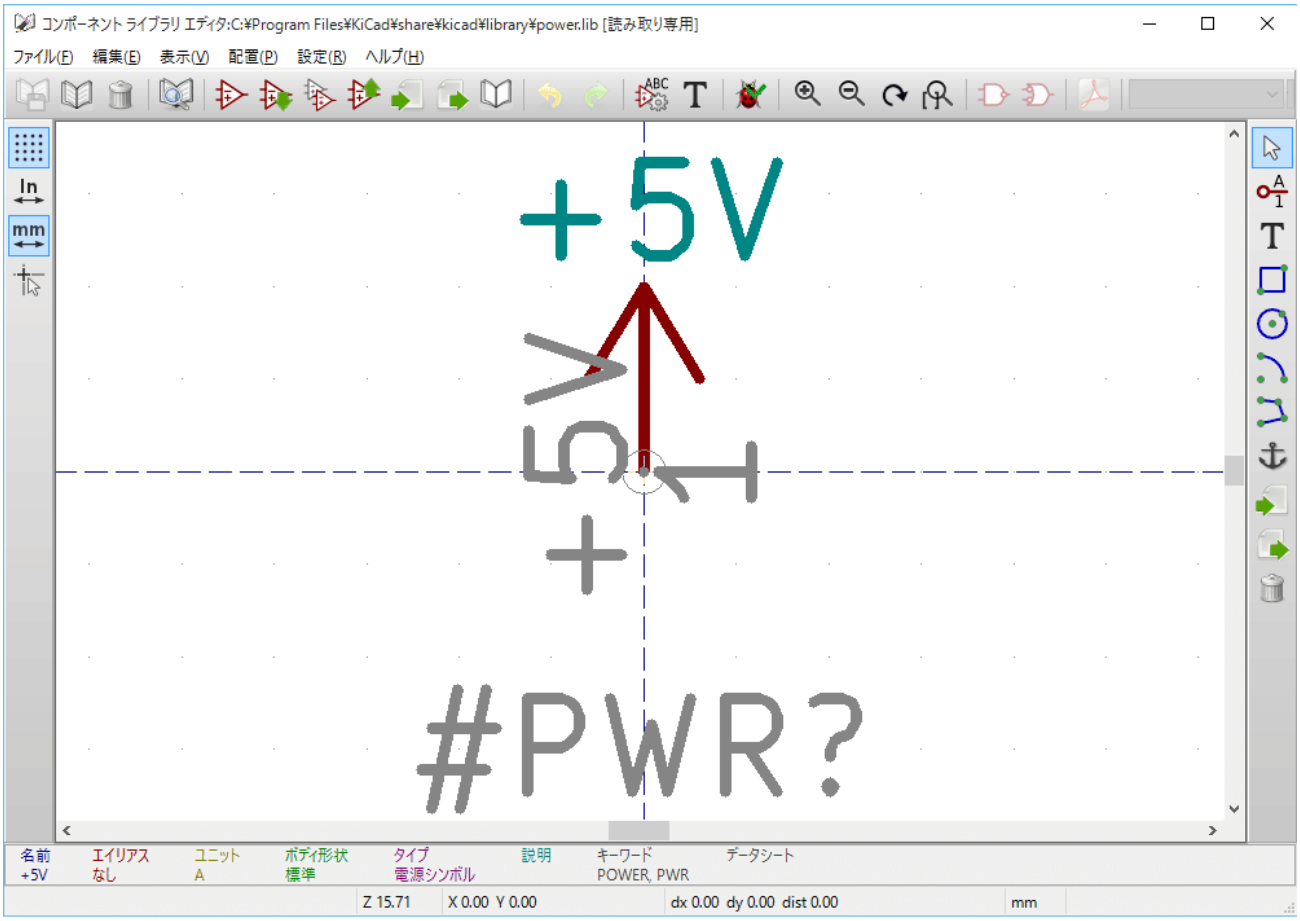
重要な注意

- 値フィールドのテキストを変更することは、型として使用する既存のコンポーネントを元にして新規コンポーネントを作成することに相当します。ライブラリに保存する場合、この新規コンポーネントは値フィールドに入っている名前を持ちます。
- 空のフィールドや非表示属性を持つフィールドを編集するには、上のコンポーネントプロパティ・ダイアログを使用する必要があります。
- フットプリントは、LIBNAME:FPNAME フォーマットを用いた完全なフットプリントとして定義されます。ここでの、LIBNAME はフットプリント・ライブラリ・テーブル中で定義されたフットプリントライブラリの名前であり、FPNAME は LIBNAME ライブラリ中でのフットプリントの名前を表します。(Pcbnew “リファレンス・マニュアル” の “フットプリント・ライブラリ・テーブル “を参照)

11.10 電源ポートシンボル

電源ポートのシンボルは通常のコンポーネントと同様に作成します。Power.lib のような専用のライブラリにそれらを集めると便利かもしれません。それらはグラフィカルなシンボル（希望する形状）で構成され、“非表示電源

“タイプのピンです。電源ポートのシンボルはこのため回路図入力ソフトウェアで他のすべてのコンポーネントと同様に扱われます。いくつかの基本的な注意事項があります。以下は +5V の電源シンボルの例です。



以下のステップに従ってシンボルを作成します。:

- +5V (+5V のネットに接続を行う (establish) ので重要) という名前で“電源入力”タイプのピンを追加します。ピンの番号は 1（番号は重要ではない）で、長さは 0、形状（グラフィックスタイル）は“ライン”とします。
- 小さな円とピンから円までの線分を配置します。
- ピン上にシンボルのアンカーを置きます。
- コンポーネントの定数を +5V とします。
- コンポーネントのリファレンスを \#+5V とします。リファレンスのテキストは、最初の文字を必ず # とします。それ以外の文字は重要ではありません。従来どおり、リファレンスがこのシンボルで始まるすべてのコンポーネントはコンポーネントリストにもネットリストのどちらにも現れません。さらに、シンボルのオプションで、リファレンスは非表示として宣言されます。

新規の電源ポートシンボルは、他のシンボルを型として使用すると容易にそして速く作成できます。:

- 型となるシンボルを読み込む。
- 新規電源ポートの名前となるピン名を編集する。

- 値フィールドを編集する (電源ポートの値を表示したい場合、ピン名と同じ名前にする)。
- その新しいコンポーネントを保存する。

Chapter 12

コンポーネント・ライブラリ・エディタ–補足

12.1 概要

コンポーネントは、次の要素で構成されています。

- グラフィカル表現（幾何学的な図形、テキスト)。
- ピン。
- ポストプロセッサで使われる関連テキスト、フィールド：ネットリスト、部品リスト。

次の２つのフィールドは初期化されます: リファレンスと定数。コンポーネントに関連したデザイン名、関連するフットプリント名、フリーフィールドであるその他フィールドは、一般に空欄のままにすることができ、回路図の編集中に追記できます。

しかしながら、コンポーネントに関連付けられたドキュメントを一緒に管理することで、コンポーネントを調べたり、ライブラリの使用やメンテナンスが容易になります。関連付けられたドキュメントは以下で構成されています：

- 説明（コメント行)。
- TTL CMOS NAND2 のように空白文字で区切られたキーワード行。
- ドキュメントファイル名（例：アプリケーションノートや pdf ファイル)。
  ドキュメントファイルのデフォルトディレクトリは以下の通りです:
  kicad/share/library/doc（Windows の場合）
  見つからない場合:
  kicad/library/doc（Windows の場合）
  linux の場合:
  /usr/local/kicad/share/library/doc
  /usr/share/kicad/library/doc
  /usr/local/share/kicad/library/doc

キーワードにより、さまざまな選択基準に従ってコンポーネントを検索することができます。キーワードと説明は、様々なメニュー、特にライブラリからコンポーネントを選択する場合に表示されます。

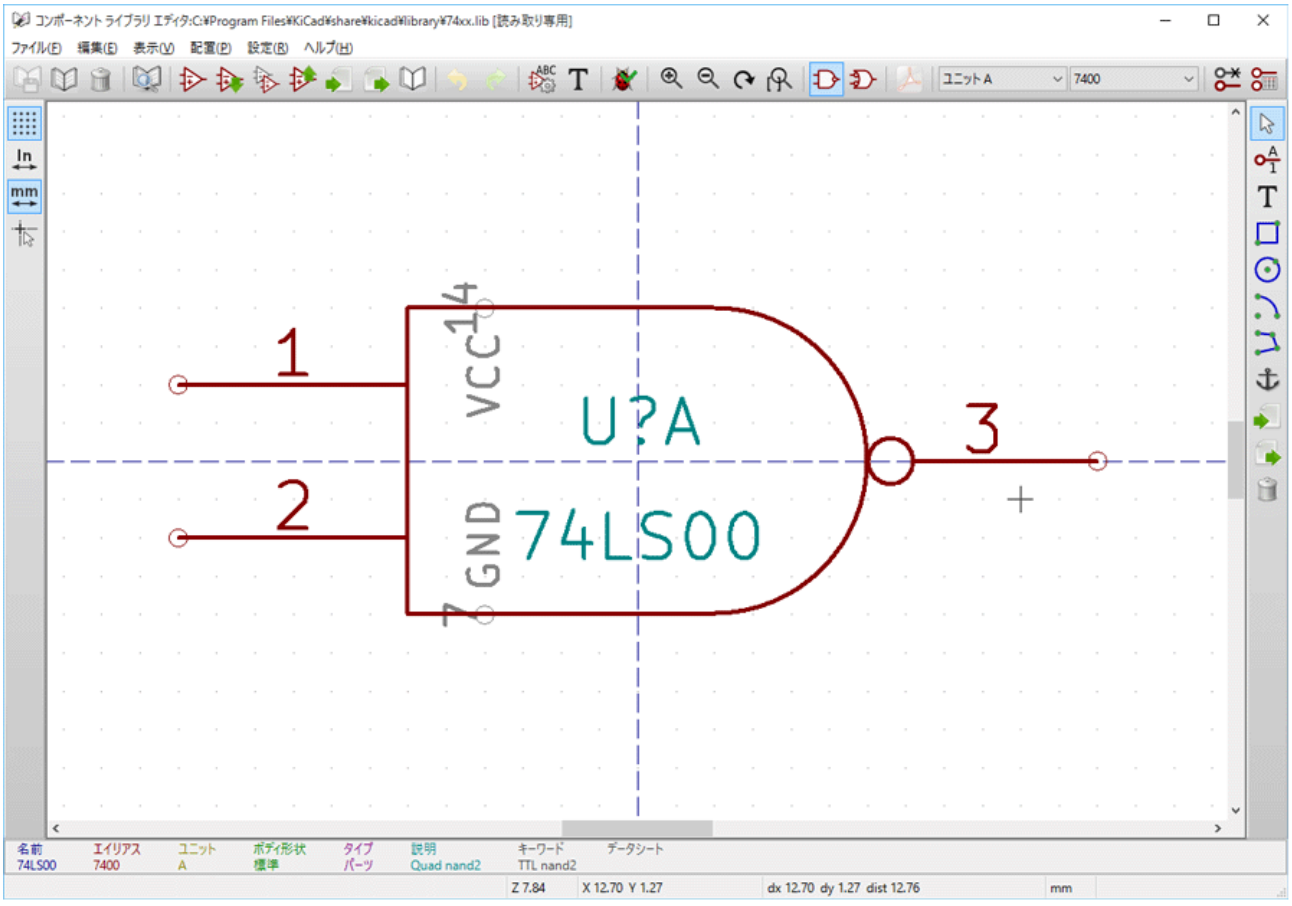
コンポーネントはまた、アンカーポイントを持っています。回転やミラーはこのアンカーポイントに対して相対的に行われ、配置時にはこの点が基準位置として使用されます。このように、このアンカーを正確に配置することは大切です。

コンポーネントは、エイリアス、つまり等価名（equivalent names）を持つことができます。これにより、作成する必要があるコンポーネントの数をかなり減らすことができます（たとえば、74LS00 は、74000、74HC00、74HCT00 …といったエイリアスを持つことができます）。

管理を容易にするため、最終的にコンポーネントはライブラリ（メーカやテーマ別に分類された）として配布されます。

12.2 コンポーネントアンカーの配置

アンカーは、座標（0,0）にあり、画面上に青い軸で示されています。



アンカーは、アイコン  を選択し、新しくアンカーを配置したい位置を左クリックすることで再配置できます。図は自動的に新しいアンカーポイントで再センタリングされます。

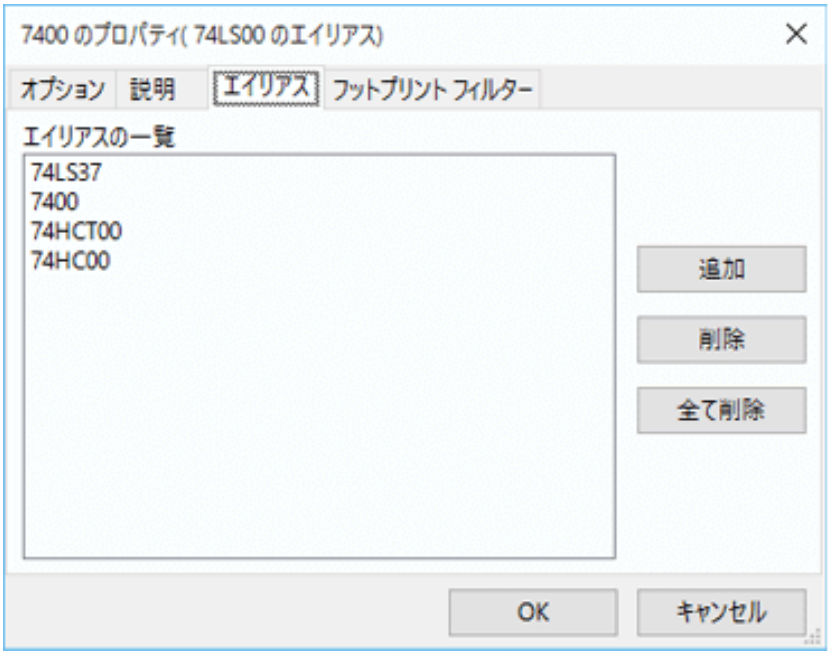
12.3 コンポーネントのエイリアス

エイリアスとは、ライブラリ内での同じコンポーネントに対応した別名のことです。類似したピン配列、表現を持つコンポーネントは、いくつかのエイリアスを持った１つのコンポーネントで表すことができます。(例えば 7400 は 74LS00、74HC00、74LS37 というエイリアスを持ちます)

エイリアスの使用により素早く完全なライブラリを構築することができます。エイリアスを使うことで、よりライブラリをコンパクトにでき、KiCad での読み込みを容易にします。

エイリアスのリストを変更するためには、メインツールバーのアイコン 

を左クリックして“コンポーネントのプロパティ”を開き、エイリアスのタブを選択する必要があります。



このように希望のエイリアスを追加したり、削除することができます。ただし、現在のエイリアスは編集されているので、明示的に削除することはできません。

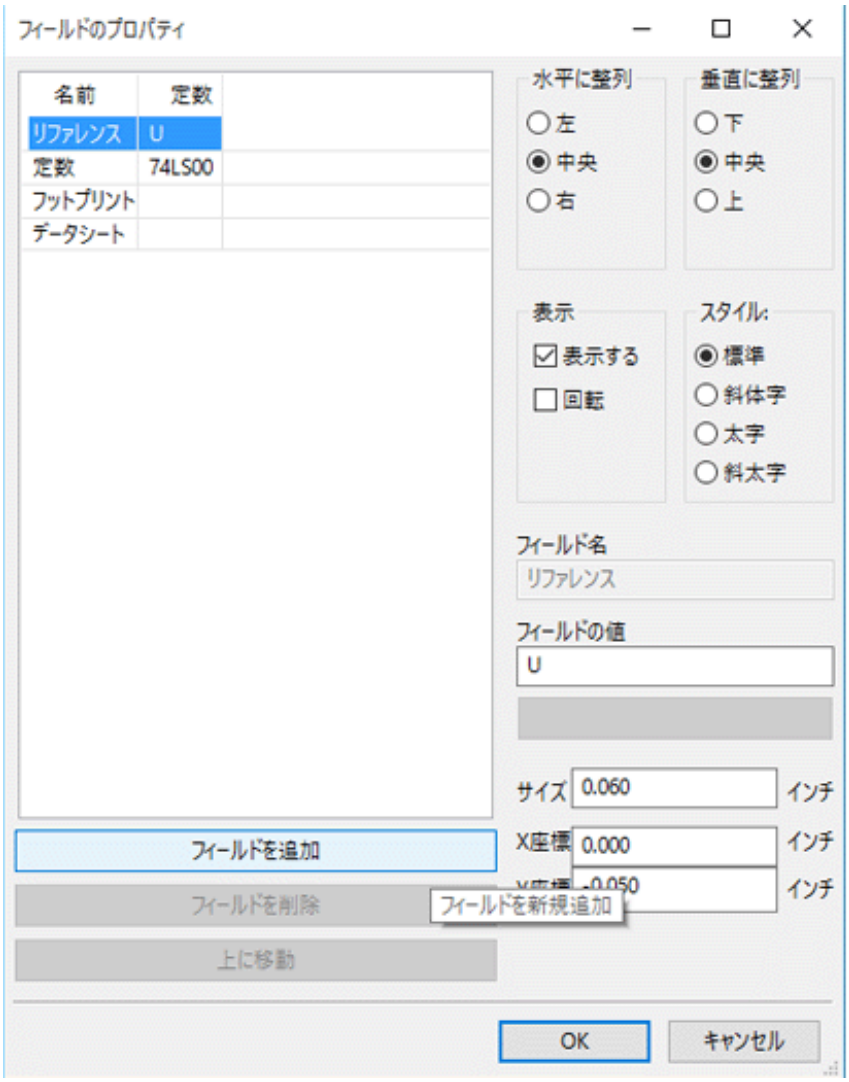
全てのエイリアスを削除するには、まずルートコンポーネントを選択することが必要です。エイリアスの最初のコンポーネントは、メインツールバーにあるエイリアス選択のドロップダウンリスト最上部に表示されます。

12.4 コンポーネントのフィールド

フィールドは、 

ツールで編集することができます。

４つの特殊フィールド（コンポーネントに付属するテキスト）と、設定可能なユーザ・フィールドがあります。



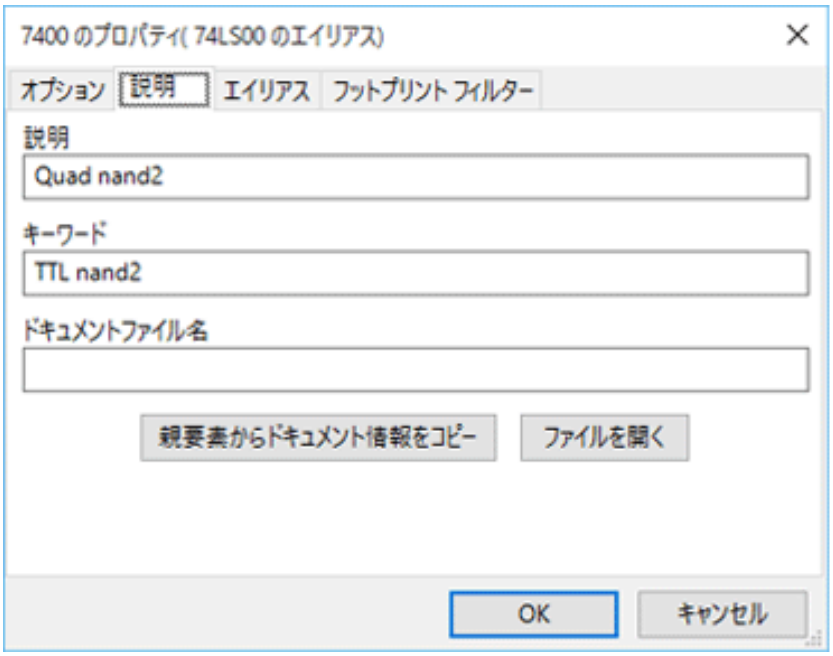
特殊フィールド

- リファレンス (Reference)：部品の参照番号（の接頭辞）。
- 定数 (Value) ：ライブラリ内のコンポーネント名であり、回路図のデフォルト値フィールド。
- フットプリント (Footprint) ：基板で使用されるフットプリント名。CvPcb を使ってフットプリントのリストを設定する場合にはそれほど有用ではありませんが、CvPcb を使用しない場合には必須です。
- データシート (Sheet) ：現時点では使用されていない予約済フィールドです。

12.5 コンポーネントのプロパティ

コンポーネントのプロパティを編集するためには、メインツールバーのアイコン 

を左クリックしてコンポーネントプロパティを開く必要があります。“説明”、“キーワード”、“ドキュメントファイル名” を編集するには、“説明” タブを開きます。



“説明” はエイリアス間の違いを表す唯一のものとなるため、正しいエイリアスあるいはルートコンポーネントを確実に選択してください。“親要素からドキュメント情報をコピー” ボタンをクリックすると、現在編集しているエイリアスに対し、ルートコンポーネントからドキュメントの情報をコピーすることができます。

12.5.1 コンポーネントのキーワード

“キーワード” により、特定の選択基準（機能、技術的ファミリなど）に従って、コンポーネントを選択的な方法で検索することができます。

Eeschema の検索ツールは、大文字と小文字を区別しません。ライブラリで使われる現状もっとも多いキーワードは次のものです。

- CMOS TTL ：ロジックファミリ
- AND2 NOR3 XOR2 INV⋯：論理ゲート (AND2 = 2 入力 AND ゲート, NOR3 = 3 入力 NOR ゲート)
- JKFF DFF⋯：JK あるいは D フリップフロップ
- ADC, DAC, MUX⋯
- OpenCol はオープンコレクタ出力を持つ論理ゲートです。例えば、回路図エディタ内で NAND2 OpenCol というキーワードでコンポーネントを検索すると、Eeschema はこれら２つのキーワードを持つコンポーネントのリストを表示します。

12.5.2 コンポーネントのドキュメントファイル名 (Doc)

特に ViewLib メニューおよびライブラリで表示されたコンポーネントの一覧からコンポーネントを選択する場合、様々なメニューに “説明”、“キーワード”、“データシート”（ドキュメントファイル）が表示されます。

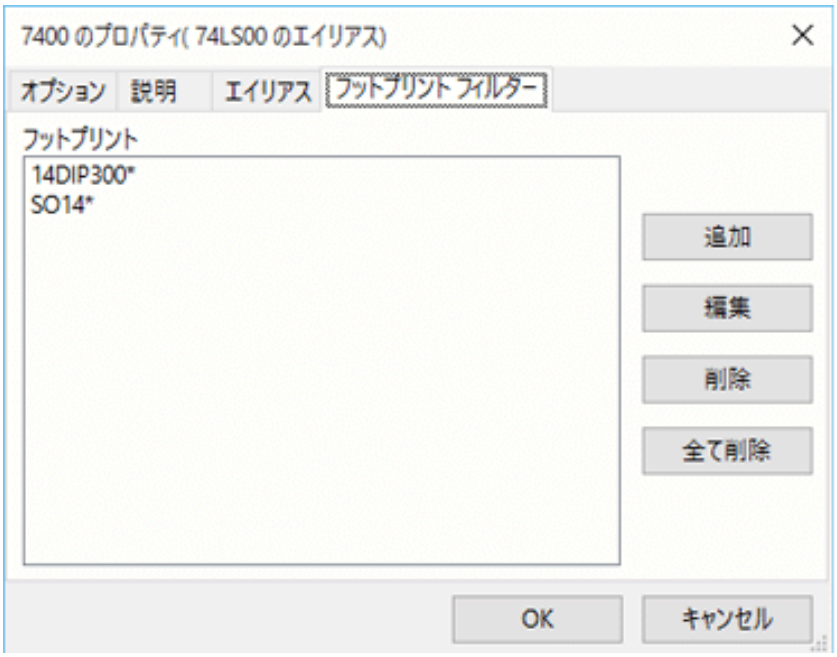
コンポーネントプロパティの “ドキュメントファイル名” が存在する場合、回路図エディタソフト内でドキュメントファイルにアクセスが可能であり、コンポーネントを右クリックして表示されるコンテキストメニューから利用できます。

12.5.3 ドキュメントファイル (DocFileName) に関して

利用可能なドキュメントファイルは、技術文書、アプリケーション回路例です。有効なファイルは、pdf ファイル、回路図などで、使用中のコンピュータから利用できるものです。

12.5.4 CvPcb のフットプリントフィルタ

コンポーネントで使用できるフットプリントのリストを入力できます。編集するには“フットプリントフィルタ”タブを開きます。このリストは Cvpcb で（使用可能なフットプリントのみを表示するための）フィルタとして働きます。空のリストは何もフィルタしません。



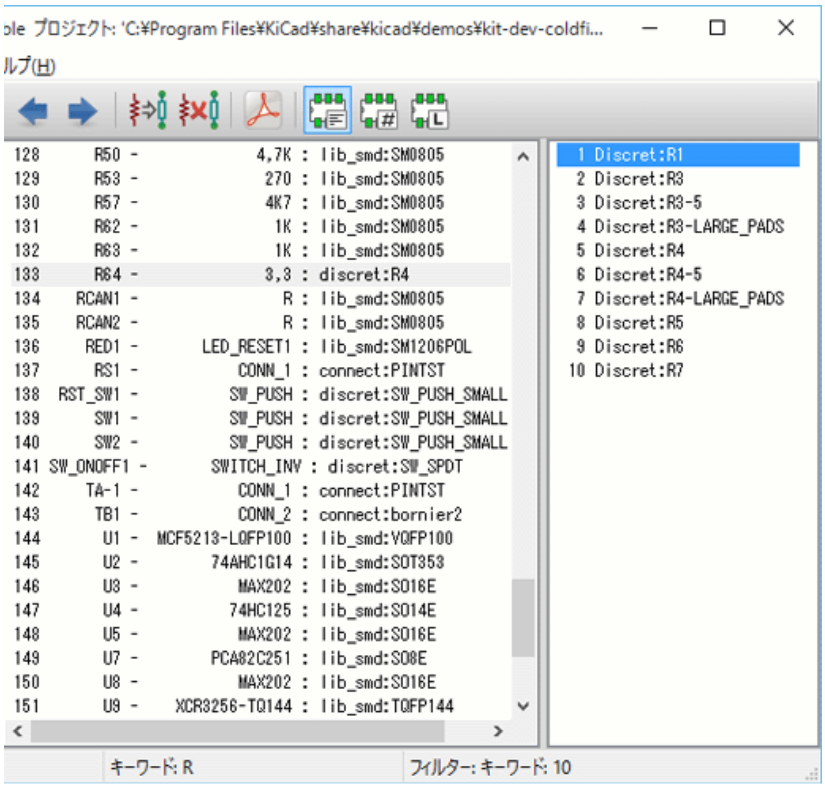
ワイルドカード文字が使用できます。

S014* により、名称が SO14 で始まる全てのフットプリントを CvPcb で表示できます。

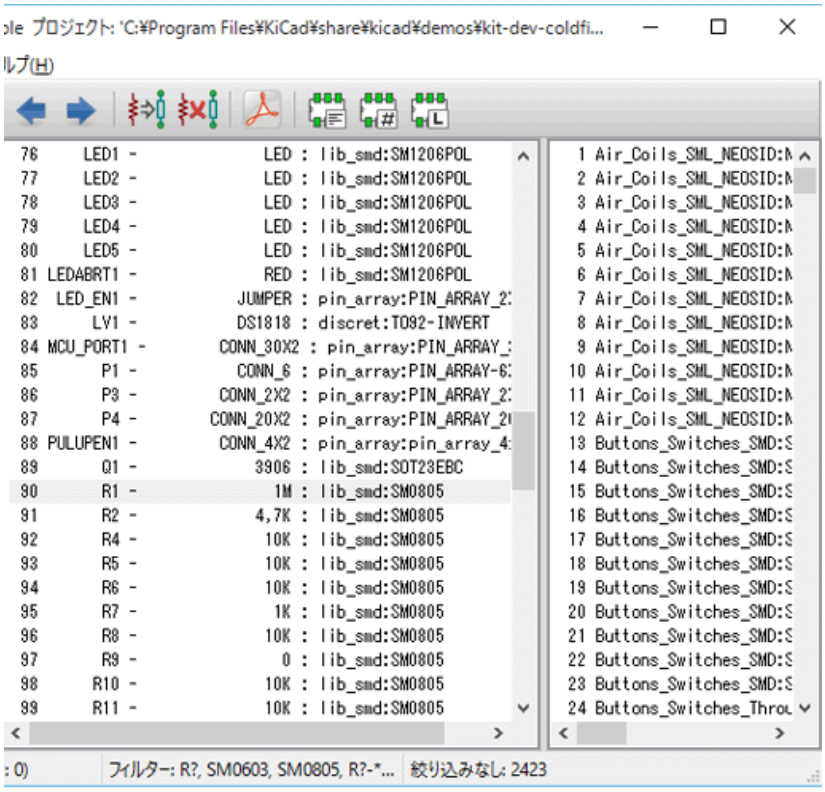
抵抗については、R? により R で始まる２文字の全てのフットプリントを示すことができます。

フィルタ有り、無しのサンプルを示します。

フィルタ有り


ole プロジェクト: 'C:¥Program Files¥KiCad¥share¥kicad¥demos¥kit-dev-coldfi...
ルプ(H)
128 R50 - 4,7K : lib_smd:SM0805
129 R53 - 270 : lib_smd:SM0805
130 R57 - 4K7 : lib_smd:SM0805
131 R62 - 1K : lib_smd:SM0805
132 R63 - 1K : lib_smd:SM0805
133 R64 - 3,3 : discret:R4
134 RCAN1 - R : lib_smd:SM0805
135 RCAN2 - R : lib_smd:SM0805
136 RED1 - LED_RESET1 : lib_smd:SM1206POL
137 RS1 - CONN_1 : connect:PINTST
138 RST_SW1 - SW_PUSH : discret:SW_PUSH_SMALL
139 SW1 - SW_PUSH : discret:SW_PUSH_SMALL
140 SW2 - SW_PUSH : discret:SW_PUSH_SMALL
141 SW_ONOFF1 - SWITCH_INV : discret:SW_SPDT
142 TA-1 - CONN_1 : connect:PINTST
143 TB1 - CONN_2 : connect:bornier2
144 U1 - MCF5213-LQFP100 : lib_smd:VQFP100
145 U2 - 74AHC1G14 : lib_smd:SOT353
146 U3 - MAX202 : lib_smd:SO16E
147 U4 - 74HC125 : lib_smd:SO14E
148 U5 - MAX202 : lib_smd:SO16E
149 U7 - PCA82C251 : lib_smd:SO8E
150 U8 - MAX202 : lib_smd:SO16E
151 U9 - XCR3256-TQ144 : lib_smd:TQFP144
1 Discret:R1
2 Discret:R3
3 Discret:R3-5
4 Discret:R3-LARGE_PADS
5 Discret:R4
6 Discret:R4-5
7 Discret:R4-LARGE_PADS
8 Discret:R5
9 Discret:R6
10 Discret:R7
キーワード: R
フィルター: キーワード: 10

フィルタなし


ole プロジェクト: 'C:¥Program Files¥KiCad¥share¥kicad¥demos¥kit-dev-coldfi...
ルプ(H)
76 LED1 - LED : lib_smd:SM1206POL
77 LED2 - LED : lib_smd:SM1206POL
78 LED3 - LED : lib_smd:SM1206POL
79 LED4 - LED : lib_smd:SM1206POL
80 LED5 - LED : lib_smd:SM1206POL
81 LEDABRT1 - RED : lib_smd:SM1206POL
82 LED_EN1 - JUMPER : pin_array:PIN_ARRAY_2
83 LV1 - DS1818 : discret:TO92-INVERT
84 MCU_PORT1 - CONN_30X2 : pin_array:PIN_ARRAY_
85 P1 - CONN_6 : pin_array:PIN_ARRAY-6
86 P3 - CONN_2X2 : pin_array:PIN_ARRAY_2
87 P4 - CONN_20X2 : pin_array:PIN_ARRAY_2
88 PULUPEN1 - CONN_4X2 : pin_array:pin_array_4
89 Q1 - 3906 : lib_smd:SOT23EBC
90 R1 - 1M : lib_smd:SM0805
91 R2 - 4,7K : lib_smd:SM0805
92 R4 - 10K : lib_smd:SM0805
93 R5 - 10K : lib_smd:SM0805
94 R6 - 10K : lib_smd:SM0805
95 R7 - 1K : lib_smd:SM0805
96 R8 - 10K : lib_smd:SM0805
97 R9 - 0 : lib_smd:SM0805
98 R10 - 10K : lib_smd:SM0805
99 R11 - 10K : lib_smd:SM0805
1 Air_Coils_SML_NEOSID:M
2 Air_Coils_SML_NEOSID:M
3 Air_Coils_SML_NEOSID:M
4 Air_Coils_SML_NEOSID:M
5 Air_Coils_SML_NEOSID:M
6 Air_Coils_SML_NEOSID:M
7 Air_Coils_SML_NEOSID:M
8 Air_Coils_SML_NEOSID:M
9 Air_Coils_SML_NEOSID:M
10 Air_Coils_SML_NEOSID:M
11 Air_Coils_SML_NEOSID:M
12 Air_Coils_SML_NEOSID:M
13 Buttons_Switches_SMD:S
14 Buttons_Switches_SMD:S
15 Buttons_Switches_SMD:S
16 Buttons_Switches_SMD:S
17 Buttons_Switches_SMD:S
18 Buttons_Switches_SMD:S
19 Buttons_Switches_SMD:S
20 Buttons_Switches_SMD:S
21 Buttons_Switches_SMD:S
22 Buttons_Switches_SMD:S
23 Buttons_Switches_SMD:S
24 Buttons_Switches_Throu
: 0)
フィルター: R?, SM0603, SM0805, R?-*...
絞り込みなし: 2423

12.6 シンボルライブラリ

頻繁に使用されるシンボルを含んだグラフィックシンボルのライブラリファイルを簡単にコンパイルすることができます。これはコンポーネント（三角形、AND、OR、ExOR ゲートなどの形状）の作成や、節約と再利用に使うことができます。

これらのファイルは、.sym という拡張子を持ち、デフォルトでライブラリディレクトリに保存されます。シンボルは一般的にそう多くないので、コンポーネントのようにライブラリ内には含まれません。

12.6.1 シンボルの作成、エクスポート

コンポーネントは  ボタンで、シンボルとしてエクスポートすることができます。一般的に 1 つのグラフィックを作成でき、ピンがある場合には、全てのピンを削除することをお勧めします。

12.6.2 シンボルのインポート

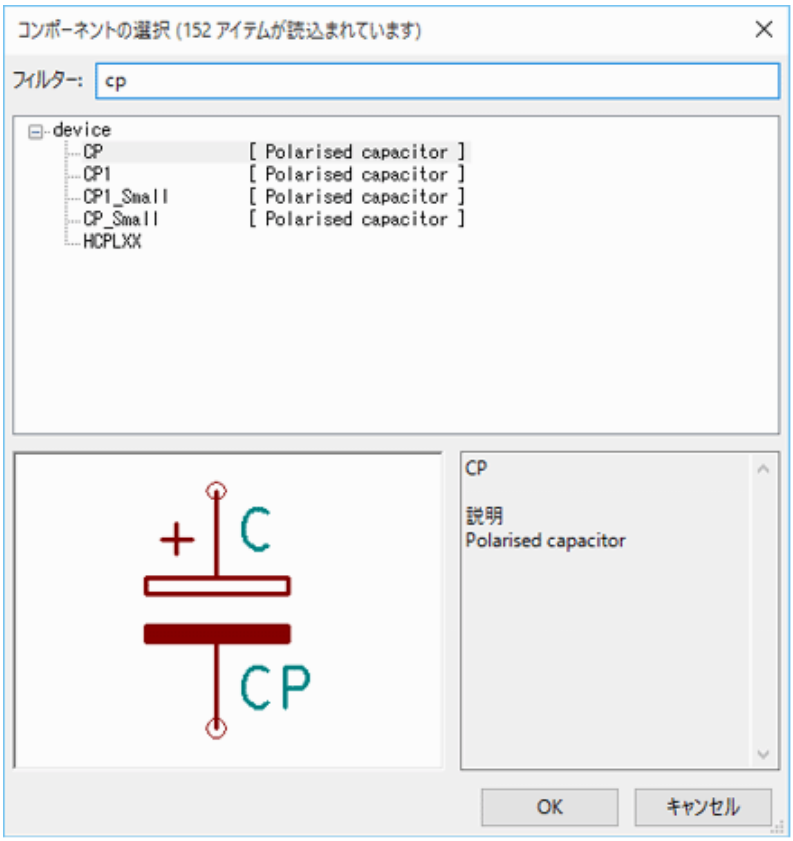
インポートすることで、編集しているコンポーネントにグラフィックを追加することができます。シンボルは、  ボタンでインポートされます。インポートされたグラフィックは、既存のグラフィックス内で作成されたものとして追加されます。

Chapter 13

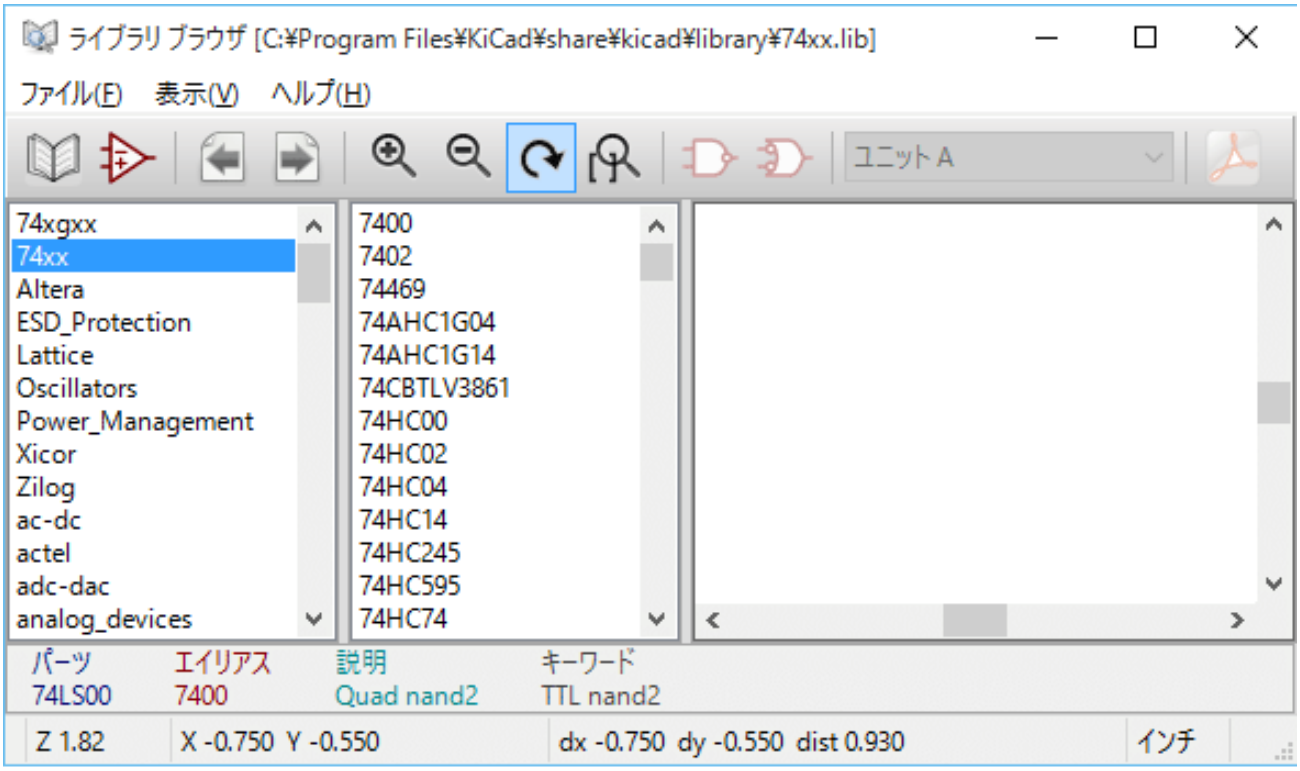
ライブラリブラウザ (Viewlib)

13.1 はじめに

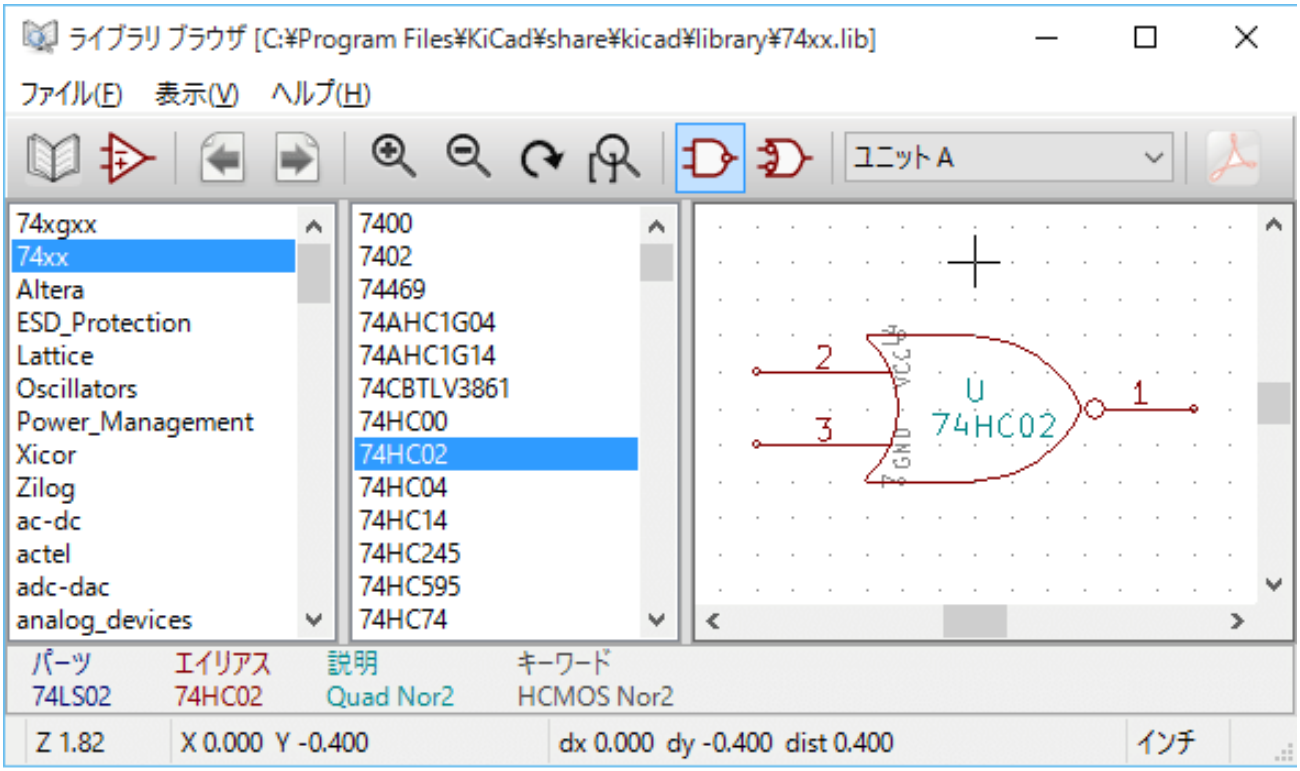
ライブラリブラウザを利用すると、ライブラリの内容をすばやく確認することができます。ライブラリブラウザは、次の２通りの方法で呼び出せます。１）メインツールバーのアイコン  を左クリックする。２）右ツールバーの“コンポーネントの配置”を左クリックして有効化する → 図面上で左クリックして“コンポーネントの選択”を呼び出す。→ “コンポーネントの選択”の画面左下のシンボル表示部を左クリックする。



13.2 ライブラリブラウザ - メインウィンドウ



ライブラリの内容を確認するには、１番左端にあるリストからライブラリを選択する必要があります。利用可能なコンポーネントが左から２番目のリストに表示されます。



13.3 ライブラリブラウザ・上部ツールバー

ライブラリブラウザの上部に表示されているツールバーを以下に示します。

利用可能なコマンドは以下のとおりです。

	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 previous component.
	Display next component.
	Zoom tools.
	Selection of the representation (normal or converted) if exist.
	Selection of the part, only for multi-part components.
	If it exist, display the associated documents. Exists only when called by the place component dialog frame from Eeschema.
	Close Viewlib and place the selected component in Eeschema. This icon is only displayed when Viewlib has been called from Eeschema (click on a symbol in the component chooser).

Chapter 14

カスタマイズされたネットリストと BOM (部品表) ファイルの生成

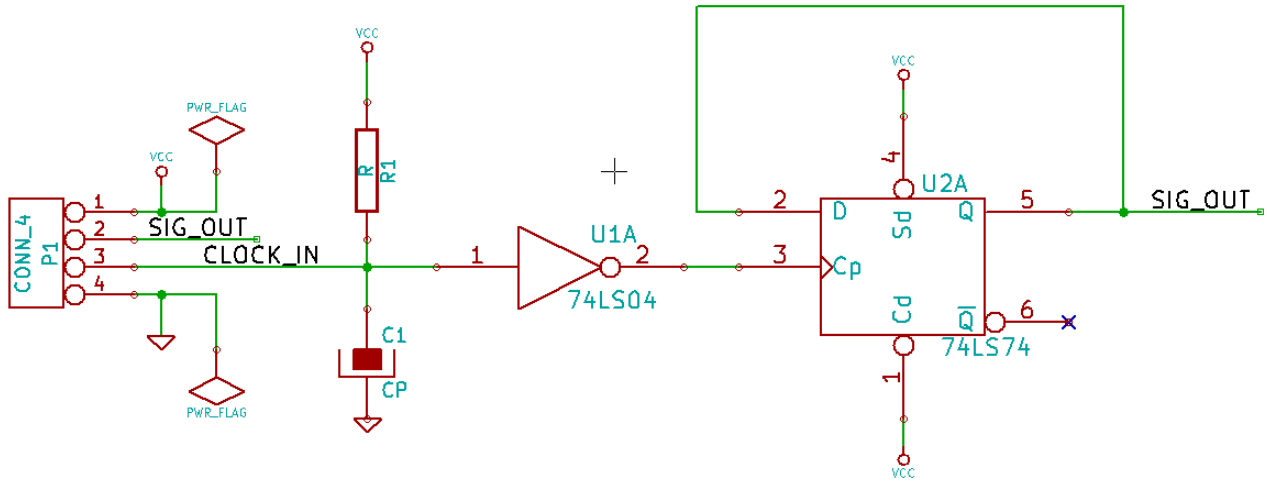
14.1 中間ネットリスト

部品表 (BOM) ファイルとネットリストファイルは、Eeschema が生成する中間ネットリストから変換されます。

このファイルは XML フォーマットで書かれており、中間ネットリストと呼ばれています。この中間ネットリストには設計中の基板に関する大量のデータが含まれており、後処理で部品表 (BOM) やさまざまなレポートを生成することができます。

出力するファイル (部品表 (BOM) かネットリスト) 次第で、中間ネットリストの利用される部分が変わってきます。

14.1.1 回路図サンプル



14.1.2 中間ネットリストのサンプル

上記回路図に対応する中間ネットリスト (XML 文法を利用しています) を以下に示します。

```
<?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"/>
      <pin num="14" name="VCC" type="power_in"/>
    </pins>
  </libpart>
  <libpart lib="74xx" part="74LS74">
    <description>Dual D FlipFlop, Set &amp;amp; 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 新しいネットリスト形式への変換

中間ネットリストファイルに後処理のフィルタリングをすることで、部品表 (BOM) ファイルのような他形式のファイルを生成できます。この変換はテキストからテキストへの変換なので、この後処理フィルタは、Python や XSLT など、入力として XML を扱える処理系で記述できます。

XSLT はそれ自身が XML 言語で書かれる、XML ファイルの処理に最適な言語です。*xsltproc* と呼ばれるフリーソフトがあり、ダウンロードしてインストールできます。xsltproc は、中間ネットリスト入力ファイルを読み込むことができ、入力を変換するためにスタイルシートを適用して、結果をファイルへ保存します。xsltproc を使用するためには、XSLT による変換処理のためのスタイルシートが必要となります。一度 xsltproc を設定して実行すると、これら全ての変換プロセスは、Eeschema によって制御されます。

14.3 XSLT のアプローチ

XSL 変換 (XSLT) に関するドキュメントは、下記より参照することが出来ます:

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

14.3.1 PADS-PCB 形式ネットリストファイルの生成

PADS-PCB 形式のネットリストは、次の２つのセクションより構成されます。

- フットプリントの一覧
- ネットリスト：ネット情報によりグループ化されたパッド情報

以下に、中間ネットリストから pads-pcb 形式へ変換するためのスタイルシートを掲載します。:

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

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

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

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

<xsl:template match="/export">
    <xsl:text>*PADS-PCB*&nl;*PART*&nl;</xsl:text>
    <xsl:apply-templates select="components/comp"/>
    <xsl:text>&nl;*NET*&nl;</xsl:text>
    <xsl:apply-templates select="nets/net"/>
    <xsl:text>*END*&nl;</xsl:text>
</xsl:template>

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

<!-- for each net -->
<xsl:template match="net">
    <!-- nets are output only if there is more than one pin in net -->
    <xsl:if test="count(node)>1">
        <xsl:text>*SIGNAL* </xsl:text>
        <xsl:choose>
            <xsl:when test = "@name != '' ">
```

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

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

</xsl:stylesheet>
```

xsltproc を実行し得られた、PADS-PCB 用のネットリストファイルを以下に示します。:

```
*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*
```

この変換は、次のコマンドラインにより実行することができます：

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

14.3.2 CADSTAR 形式のネットリストファイルの生成

CADSTAR 形式のネットリストは、下記の２セクションで構成されています。

- フットプリントの一覧
- ネットリスト：ネット情報によりグループ化されたパッド情報

以下に変換するためのスタイルシートを掲載します。:

```
<?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>
```

CADSTAR 形式の出力ファイルです。

```
.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 OrCAD PCB2 形式ネットリストファイルの生成

このフォーマットは、フットプリントの一覧のみで構成されています。それぞれのフットプリントは接続されるネットの情報を含みます。

変換を行うためのスタイルシートファイルを、以下に示します。

```
<?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 &lt;2">
                <xsl:text>?</xsl:text>
            </xsl:if>
```

```
        </xsl:if>
    </xsl:if>

</xsl:template>

</xsl:stylesheet>
```

OrCAD PCB2 形式の出力ファイルです。

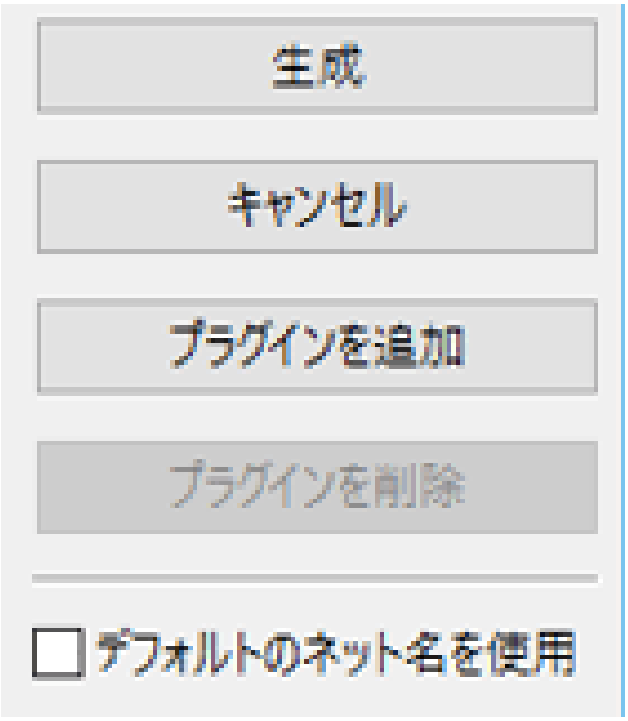
```
( { Eeschema Netlist Version 1.1  29/08/2010 21:07:51
eeschema (2010-08-28 BZR 2458)-unstable}
 ( 4C6E2141 $noname P1 CONN_4
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 /CLOCK_IN )
  (  4 GND )
 )
 ( 4C6E20BA $noname U2 74LS74
  (  1 VCC )
  (  2 /SIG_OUT )
  (  3 N-04 )
  (  4 VCC )
  (  5 /SIG_OUT )
  (  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 Eeschema プラグイン・インタフェイス

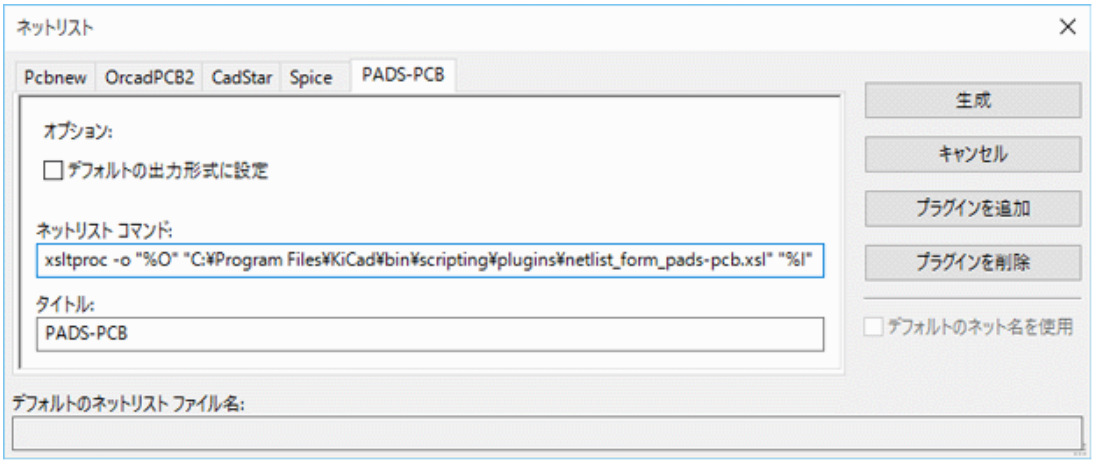
中間ネットリストの変換は Eeschema の中で自動的に実行させることができます。

14.3.4.1 ダイアログウィンドウの初期化

新しいネットリストプラグインをユーザインタフェースへ追加するには、“プラグインの追加”ボタンをクリックします。



PADS-PCB タブの設定は、下記のようになります:



14.3.4.2 プラグインの設定パラメータ

Eeschema のプラグイン設定ダイアログでは、下記の情報が必要になります:

- タイトル: ネットリストフォーマットの名前など
- 変換を行うためのコマンドライン

ネットリストボタンをクリックすると、次のように実行されます。

1. Eeschema は test.xml のように、中間ネットリスト *.xml を生成します。
2. Eeschema は test.xml を読み込むことでプラグインを実行し、test.net を生成します。

14.3.4.3 コマンドラインからのネットリストファイルの生成

xsltproc.exe を利用して中間ネットリストへスタイルシートを適用する場合、下記のコマンドにより *xsltproc.exe* が実行されます。

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

Windows 環境で KiCad を利用している場合のコマンドラインは以下のようになります。

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

Linux 環境の場合のコマンドを以下に示します。

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

netlist_form_pads-pcb.xsl には、適用するスタイルシートのファイル名が入ります。このスタイルシートのファイル名をダブルクォーテーション (") で囲むのを忘れないようにしてください。これにより、Eeschema による変換後のスペースの使用を許容します。

コマンドラインフォーマットはファイル名のパラメータを受け付けます:

サポートしているフォーマットパラメータを下記に示します:

- %B　選択された出力ファイルのパスとファイル名から、パスと拡張子を除いたもの。
- %I　現在の入力ファイル（中間ネットリストファイル）の完全なパスとファイル名。
- %O　ユーザが選んだ出力ファイルの完全なパスとファイル名。

%I は実際の中間ネットリストファイル名へ置換されます。

%O は実際の出力ファイル名へ置換され、最終的なネットリストファイルとなります。

14.3.4.4 コマンドラインフォーマット: xsltproc の例

xsltproc のコマンドラインフォーマットは、下記のようになります：

<path of xsltproc> xsltproc <xsltproc parameters>

Windows 環境の場合:

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

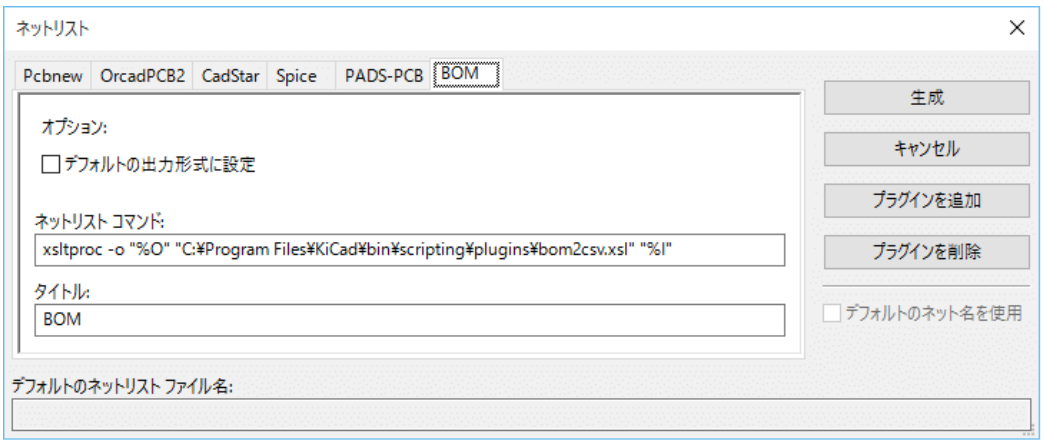
Linux 環境の場合:

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

上記は、xsltproc が Windows 環境下で kicad/bin 以下にインストールされていると仮定したものです。

14.3.5 BOM （部品表）の生成

中間ネットリストファイルは、使用されているコンポーネント全ての情報を含んでいるため、ここから BOM（部品表）を生成することができます。以下に BOM（部品表）を生成させるための、Windows (Linux) 環境下でのプラグイン設定ウィンドウを示します:



bom2csv.xsl のパスは、システムによって異なります。現状で最適な BOM （部品表）を生成する XSLT スタイルシートは、ここでは *bom2csv.xsl* とします。

14.4 コマンドラインフォーマット: python スクリプトの例

python を使用した場合のコマンドラインフォーマットは、下記のようになります:

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

Windows 環境の場合:

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

Linux 環境の場合:

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

(あなたの PC に python がインストールされている必要があります。)

14.5 中間ネットリストファイルの構造

ネットリストファイルの例を次に示します。

```
<?xml version="1.0" encoding="utf-8"?>
<export version="D">
  <design>
    <source>F:\kicad_aux\netlist_test\netlist_test.sch</source>
    <date>29/08/2010 21:07:51</date>
```

```
    <tool>eeschema (2010-08-28 BZR 2458)-unstable</tool>
  </design>
  <components>
    <comp ref="P1">
      <value>CONN_4</value>
      <libsource lib="conn" part="CONN_4"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2141</tstamp>
    </comp>
    <comp ref="U2">
      <value>74LS74</value>
      <libsource lib="74xx" part="74LS74"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20BA</tstamp>
    </comp>
    <comp ref="U1">
      <value>74LS04</value>
      <libsource lib="74xx" part="74LS04"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E20A6</tstamp>
    </comp>
    <comp ref="C1">
      <value>CP</value>
      <libsource lib="device" part="CP"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E2094</tstamp>
    <comp ref="R1">
      <value>R</value>
      <libsource lib="device" part="R"/>
      <sheetpath names="/" tstamps="/"/>
      <tstamp>4C6E208A</tstamp>
    </comp>
  </components>
  <libparts/>
  <libraries/>
  <nets>
    <net code="1" name="GND">
      <node ref="U1" pin="7"/>
      <node ref="C1" pin="2"/>
      <node ref="U2" pin="7"/>
      <node ref="P1" pin="4"/>
    </net>
    <net code="2" name="VCC">
      <node ref="R1" pin="1"/>
      <node ref="U1" pin="14"/>
      <node ref="U2" pin="4"/>
      <node ref="U2" pin="1"/>
      <node ref="U2" pin="14"/>
```

```
      <node ref="P1" pin="1"/>
    </net>
    <net code="3" name="">
      <node ref="U2" pin="6"/>
    </net>
    <net code="4" name="">
      <node ref="U1" pin="2"/>
      <node ref="U2" pin="3"/>
    </net>
    <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>
```

14.5.1 一般的なネットリストファイルの構造

中間ネットリストファイルは、次の５つのセクションで構成されています。

- ヘッダーセクション
- コンポーネントセクション
- ライブラリパーツセクション
- ライブラリセクション
- ネットセクション

このファイルは <export> タグで囲まれたものとなります。

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

14.5.2 ヘッダーセクション

このヘッダは <design> タグで囲まれます。

```
<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>
```

このセクションはコメントセクションとして捉えることができます。

14.5.3 コンポーネントセクション

このコンポーネントセクションは <components> タグで囲まれたものとなります。

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

このセクションには、回路図中で使用されているコンポーネントの一覧が含まれます。それぞれのコンポーネントは、次のように記載されます。:

```
<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.

14.5.3.1 コンポーネントのタイムスタンプに関する注意

ネットリスト（つまり基板上）のコンポーネントを識別する際、タイムスタンプ情報はコンポーネントに固有の情報となります。一方で、KiCad はコンポーネントを基板上の対応するフットプリントから識別する方法を補助的に用意しています。これは、回路図プロジェクト中のコンポーネントを再アノテーションできるようにし、コンポーネントとフットプリント間の結びつき情報を消失しないようにするためのものです。

タイムスタンプは、それぞれのコンポーネントや回路図プロジェクト内のシートにおいて固有の識別子です。しか

しながら、複雑な階層構造内で同じシートが複数回参照される場合、同じタイムスタンプを持つコンポーネントが存在することになってしまいます。

このような複雑な階層構造を持つシートでは、シートのパス情報を利用して固有の識別子を持ちます。コンポーネントは、“そのシートのパス＋タイムスタンプ”を固有の識別子として持ちます。

14.5.4 ライブラリパーツ・セクション

このライブラリパーツセクションは、<libparts> タグで囲まれたものとなり、このセクションは回路図ライブラリの情報を定義するものとなります。このセクションは、次のものを含みます：

- <fp> で定義されるフットプリント名（名前にはワイルドカードも使われます）
- <fields> で定義されるフィールド
- <pins> で定義されるピンリスト

```
<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>
```

<pin num=“1” type=“passive” /> のような行は、ピンの電気的な種類を定義するものです。有効なピンの種類は、次のものがあります。

Input	通常の入力
Output	通常の出力
Bidirectional	入力または出力
Tri-state	バスの入出力
Passive	受動部品のピン
Unspecified	不明な種類
Power input	コンポーネントの電源入力
Power output	レギュレータ IC のような部品の電源出力
Open collector	アナログコンパレータでよくみられるオープンコレクタ出力
Open emitter	ロジック IC でみられるオープンエミッタ出力
Not connected	回路図上でオープン（未接続）とすべきピン

14.5.5 ライブラリ・セクション

ライブラリセクションは <libraries> タグで囲まれたものとなります。このセクションはプロジェクトから利用されているライブラリ情報を含みます。

```
<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>
```

14.5.6 ネット・セクション

ネットセクションは <nets> タグで囲まれたものとなります。このセクションは、回路図上の接続情報を定義するものです。

```
<nets>
  <net code="1" name="GND">
    <node ref="U1" pin="7"/>
    <node ref="C1" pin="2"/>
    <node ref="U2" pin="7"/>
    <node ref="P1" pin="4"/>
  </net>
  <net code="2" name="VCC">
    <node ref="R1" pin="1"/>
    <node ref="U1" pin="14"/>
    <node ref="U2" pin="4"/>
    <node ref="U2" pin="1"/>
    <node ref="U2" pin="14"/>
    <node ref="P1" pin="1"/>
  </net>
</nets>
```

このセクションでは、回路図上の全てのネットを羅列します。

ネット情報の例を次に示します。

```
<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	ネットの内部的な識別番号
name	ネット名
node	ネットに接続されるピン

14.6 xsltproc に関する追加情報

次の Web ページを参照してください: *http://xmlsoft.org/XSLT/xsltproc.html*

14.6.1 はじめに

xsltproc は XSLT スタイルシートを XML 文書に適用するためのコマンドラインツールです。これは GNOME プロジェクトの一環として開発され、GNOME デスクトップ環境が無くても利用することが可能です。

xsltproc はスタイルシート名と適用するファイル名をオプションとし、コマンドラインより起動されます。標準入力を利用する場合、ファイル名には-記号を利用します。

スタイルシートが XML 文書内に含まれている場合、コマンドラインでスタイルシート名を指示する必要はありません。xsltproc は自動的にスタイルシートを検出し利用します。標準では、出力が 標準出力 *(stdout)* となっています。ファイルとして結果を出力したい場合には、-o オプションを利用します。

14.6.2 概要

```
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* ]
[ *....* ]
```

14.6.3 コマンドラインオプション

-V 又は *--version*

利用している libxml と libxslt のバージョン情報を表示します。

-v 又は *--verbose*

xsltproc がスタイルシートとドキュメントを処理する各段階でメッセージを出力します。

-o 又は *--output file*

< ファイル名 > で指定されたファイルへ結果を出力します。「チャンク」などとして知られているように、複数出力したい場合は -o ディレクトリ名/ として指定したディレクトリへファイルを出力します。この場合、ディレクトリは予め作成しておく必要があります。

--timing

スタイルシートの構文解析、ドキュメントの構文解析、スタイルシートの適用、結果の保存にかかった時間を表示します。ミリ秒の単位で表示されます。

--repeat

タイミングテストの為に、変換を 20 回繰り返し実行します。

--debug

デバッグの為に、変換されたドキュメントの XML ツリーを出力します。

--novalid

ドキュメントの DTD の読み込みをスキップします。

--noout

結果を出力しません。

--maxdepth <値>

libxslt の無限ループを防ぐため、テンプレートの最大スタック深度を調整します。デフォルトは 500 です。

--html

HTML ファイルを入力ファイルとします。

--param <パラメータ名><値>

スタイルシート中の、パラメータで指定された < パラメータ名 > *name* および < 値 > *value* の処理を行いません。パラメータ名と値のペアは、最大 32 個まで指定することができます。値をノードの識別ではなく、文字列として処理したい場合は、--stringparam オプションを利用してください。

--stringparam < パラメータ名 > < 値 >

< パラメータ名 > *name* と < 値 > *value* で指定された値について、ノードの識別ではなく文字列として扱うようにします。(注：これら文字列は utf-8 エンコードされている必要があります。)

--nonet

DTD のエンティティやドキュメントをインターネットから取得しません。

--path < パス >

DTD やエンティティ、ドキュメントの読み込みに、< パス > で指定されたファイルのリスト（半角スペースやカンマで区切られる）を使用します。

--load-trace

処理中に読み込まれた全てのドキュメントを、標準エラー出力 (stderr) へ出力します。

--catalogs

SGML_CATALOG_FILES 内で指定された SGML カタログを外部エンティティの解決に利用します。標準では、xsltproc は XML_CATALOG_FILES で指定された場所を探します。XML_CATALOG_FILES が定義されていない場合、/etc/xml /catalog を利用します。

--xinclude

Xinclude の仕様に基づき、入力ドキュメントの処理を行います。Xinclude の詳細は、次を参照してください：
http://www.w3.org/TR/xinclude/

--profile --norman

スタイルシートのそれぞれのパーツの処理時に、プロファイル情報の詳細を出力します。これはスタイルシートのパフォーマンスを最適化するために利用できます。

--dumpextensions

登録済みの拡張子のリストを標準出力 (stdout) へ出力します。

--nowrite

ファイルやリソースへの書き込みを行いません。

--nomkdir

ディレクトリを作成しません。

--writesubtree < パス >

< パス > で指定されたパス内のファイルのみ書込します。

--nodtdattr

ドキュメント内 DTD の標準アトリビュートを適用しません。

14.6.4 Xsltproc の戻り値

xsltproc はスクリプトからの呼び出し時などに利用しやすいよう、戻り値でステータスを返します。

0: 通常

1: 引数なし

2: パラメータが多すぎる

3: 不明なオプション

4: スタイルシートの構文解析に失敗 (parse error)

5: スタイルシート内にエラー

6: ドキュメントのひとつにエラー

7: 未サポートの xsl : 出力メソッド

8: 文字列パラメータがクオートとダブルクオートの両方を含んでいる

9: 内部処理エラー

10: 中断シグナル (CTRL+C など) により処理を終了

11: 出力ファイルに書き込めない

14.6.5 xsltproc に関する追加情報

libxml web ページ: http://www.xmlsoft.org/

W3C XSLT ページ: http://www.w3.org/TR/xslt