



Eeschema

November 6, 2020

Contents

1	Eeschema 入門	1
1.1	概要	1
1.2	技術的な概要	1
2	Eeschema コマンド全般	3
2.1	マウスコマンド	4
2.1.1	基本コマンド	4
2.1.2	ブロックの操作	4
2.2	ホットキー	5
2.3	グリッド	6
2.4	ズームの選択	7
2.5	カーソルの座標表示	7
2.6	上部メニューバー	7
2.7	上部ツールバー	8
2.8	右ツールバーのアイコン	10
2.9	左ツールバーのアイコン	11
2.10	コンテキストメニューとクイックエディット	11
3	上部メニューバーの主なメニュー	13
3.1	ファイルメニュー	13
3.2	設定メニュー	14
3.2.1	シンボルライブラリーテーブルの管理	15
3.2.1.1	ライブラリーを追加	15
3.2.1.2	ライブラリーを削除	15
3.2.1.3	ライブラリーのプロパティ	16

3.2.2	全般オプション	17
3.2.2.1	表示	17
3.2.2.2	編集	18
3.2.2.3	コントロール	19
3.2.2.4	色	20
3.2.2.5	デフォルトのフィールド	20
3.3	ヘルプ・メニュー	21
4	上部ツールバーの主なツール	22
4.1	ページ設定	22
4.2	検索ツール	22
4.3	ネットリストツール	23
4.4	アノテーションツール	24
4.5	エレクトリカル・ルール・チェック (ERC) ツール	24
4.5.1	ERC ダイアログ: ERC	25
4.5.2	ERC ダイアログ: オプション	26
4.6	部品表 (BOM) ツール	26
4.7	フィールド編集ツール	28
4.7.1	簡単にフィールドを埋める裏技	29
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:	30
4.8.1	アクセス:	30
5	シンボル・ライブラリーの管理	31
5.1	シンボル・ライブラリー・テーブル	31
5.1.1	グローバル・シンボル・ライブラリー・テーブル	32
5.1.2	プロジェクト固有のシンボル・ライブラリー・テーブル	32
5.1.3	初期設定	32
5.1.4	テーブル要素の追加	33
5.1.5	環境変数の代替	33
5.1.6	使用パターン	34
5.1.7	以前のプロジェクトのリマッピング	34

6	回路図の作成と編集	36
6.1	はじめに	36
6.2	基本的な検討事項	36
6.3	開発フロー	37
6.4	シンボルの配置と編集	37
6.4.1	シンボルの検索と配置	37
6.4.2	電源ポート（コンポーネント）	38
6.4.3	（配置された）シンボルの編集と変更	38
6.4.3.1	シンボルの変更	39
6.4.3.2	テキストフィールドの編集	39
6.5	ワイヤー、バス、ラベル、電源ポートの接続	39
6.5.1	はじめに	39
6.5.2	接続（ワイヤーとラベル）	40
6.5.3	バスの接続	41
6.5.3.1	バスのメンバー	41
6.5.3.2	バスメンバー同士の接続	42
6.5.3.3	バスのシート間接続	42
6.5.4	電源ポートの接続	43
6.5.5	“空き端子” フラグ	44
6.6	回路図作成に関する補足	44
6.6.1	テキストコメント	44
6.6.2	シートの表題欄（タイトルブロック）	44
6.7	キャッシュされたシンボルのレスキュー	45
7	階層回路図	47
7.1	はじめに	47
7.2	階層内のナビゲーション	48
7.3	ローカルラベル、階層ラベル、グローバルラベル	48
7.3.1	プロパティ	48
7.4	階層作成の要約	48
7.5	シートシンボル	49
7.6	接続 - 階層ピン	49
7.7	接続 - 階層ラベル	50

7.7.1	(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	52
7.7.1.1	(単純な) ラベル	52
7.7.1.2	階層ラベル	52
7.7.1.3	非表示電源ピン	52
7.7.2	グローバルラベル	52
7.8	複合階層	53
7.9	平面階層	53
8	シンボルアノテーションツール	56
8.1	はじめに	56
8.2	例	57
8.2.1	アノテーションの順序	57
8.2.2	アノテーションの選択	58
9	ERC (エレクトリカル・ルール・チェック) による設計検証	61
9.1	はじめに	61
9.2	ERC の使用法	61
9.3	ERC の例	62
9.4	診断結果の表示	62
9.5	電源ピンと電源フラグ	63
9.6	ルールの設定	64
9.7	ERC レポートファイル	65
10	ネットリストの作成	66
10.1	概要	66
10.2	ネットリストフォーマット	67
10.3	ネットリストの例	67
10.4	ネットリストについての注釈	70
10.4.1	ネットリスト名の注意事項	70
10.4.2	PSPICE ネットリスト	70
10.5	他のフォーマット	72
10.5.1	ダイアログウィンドウの初期設定	72
10.5.2	コマンドライン・フォーマット	72
10.5.3	コンバーターとシートスタイル (プラグイン)	73
10.5.4	中間ネットリスト・ファイル・フォーマット	73

11	プロットと印刷	74
11.1	はじめに	74
11.2	プロットの共通コマンド	74
11.3	Postscript のプロット	74
11.4	PDF のプロット	75
11.5	SVG のプロット	75
11.6	DXF のプロット	76
11.7	HPGL のプロット	76
11.7.1	シートサイズ選択	77
11.7.2	オフセット調整	77
11.8	紙面印刷	77
12	シンボル・ライブラリー・エディター	78
12.1	シンボル・ライブラリーに関する一般情報	78
12.2	シンボル・ライブラリーの概要	78
12.3	シンボル・ライブラリー・エディターの概要	79
12.3.1	メイン・ツールバー	79
12.3.2	エレメント・ツールバー	81
12.3.3	オプション・ツールバー	81
12.4	ライブラリーの選択および保守	82
12.4.1	シンボルの選択と保存	82
12.4.1.1	シンボルの選択	82
12.4.1.2	シンボルの保存	83
12.4.1.3	別のライブラリーへのシンボルの移動	83
12.4.1.4	シンボル変更の取り消し	84
12.5	ライブラリー・シンボルの作成	84
12.5.1	新規シンボルの作成	84
12.5.2	他のシンボルからシンボルを作成	85
12.5.3	シンボル・プロパティ	85
12.5.4	代替シンボル表現をもつシンボル	87
12.6	グラフィック要素	87
12.6.1	グラフィック要素のメンバーシップ。	88
12.6.2	グラフィック・テキスト要素	89

12.7 複数ユニット・シンボルと代替ボディ・スタイル	89
12.7.1 異なったシンボル表現で複数ユニットを持つシンボルの例:	89
12.7.1.1 グラフィック・シンボル要素	90
12.8 ピンの作成および編集	91
12.8.1 ピンの概要	91
12.8.2 ピンのプロパティ	92
12.8.3 ピンの形状 (グラフィックスタイル)	92
12.8.4 ピンの電氣的タイプ	93
12.8.5 ピンのグローバル・プロパティ	93
12.8.6 複数ユニットおよび代替シンボル表現におけるピン定義	93
12.9 シンボルのフィールド	94
12.9.1 シンボル・フィールドの編集	94
12.10 電源シンボル	95
13 LibEdit - シンボル	97
13.1 概要	97
13.2 シンボル・アンカーの配置	98
13.3 シンボルのエイリアス	98
13.4 シンボルのフィールド	99
13.5 シンボルのドキュメント	100
13.5.1 シンボルのキーワード	100
13.5.2 シンボルのドキュメント・ファイル名 (Doc)	100
13.5.3 関連付けられたドキュメント・ファイル (DocFileName)	101
13.5.4 CvPcb のフットプリント・フィルター	101
13.6 シンボル・ライブラリー	102
13.6.1 シンボルの作成、エクスポート	102
13.6.2 シンボルのインポート	102
14 シンボル・ライブラリー・ブラウザー	103
14.1 はじめに	103
14.2 Viewlib - メイン・スクリーン	104
14.3 シンボル・ライブラリー・ブラウザーの上部ツールバー	104

15 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	106
15.1 中間ネットリスト	106
15.1.1 回路図のサンプル	107
15.1.2 中間ネットリストのサンプル	107
15.2 新しいネットリスト形式への変換	111
15.3 XSLT のアプローチ	111
15.3.1 PADS-PCB 形式ネットリストファイルの生成	111
15.3.2 CADSTAR 形式のネットリストファイルの生成	114
15.3.3 OrCAD PCB2 形式ネットリスト・ファイルの生成	117
15.3.4 Eeschema プラグイン・インタフェース	122
15.3.4.1 ダイアログウィンドウの初期化	123
15.3.4.2 プラグインの設定パラメータ	123
15.3.4.3 コマンドラインからのネットリストファイルの生成	123
15.3.4.4 コマンドラインフォーマット: xsltproc の例	124
15.3.5 BOM (部品表) の生成	124
15.4 コマンドラインフォーマット: python スクリプトの例	125
15.5 中間ネットリストファイルの構造	125
15.5.1 一般的なネットリストファイルの構造	127
15.5.2 ヘッダーセクション	127
15.5.3 コンポーネントセクション	127
15.5.3.1 コンポーネントのタイムスタンプに関する注意	128
15.5.4 ライブラリーパーツ・セクション	128
15.5.5 ライブラリー・セクション	129
15.5.6 ネット・セクション	129
15.6 xsltproc に関する追加情報	130
15.6.1 はじめに	130
15.6.2 概要	131
15.6.3 コマンドラインオプション	131
15.6.4 Xsltproc の戻り値	133
15.6.5 xsltproc に関する追加情報	133

16 シミュレーター	134
16.1 モデルの割り当て	134
16.1.1 パッシブ	135
16.1.2 モデル	136
16.1.3 ソース	138
16.2 Spice 指令	138
16.3 シミュレーション	139
16.3.1 メニュー	140
16.3.1.1 ファイル	140
16.3.1.2 シミュレーション	140
16.3.1.3 表示	140
16.3.2 ツールバー	140
16.3.3 プロット・パネル	141
16.3.4 出力コンソール	141
16.3.5 信号リスト	141
16.3.6 カーソル・リスト	141
16.3.7 調整パネル	141
16.3.8 調整ツール	142
16.3.9 プロブ・ツール	142
16.3.10 シミュレーションの設定	143

リファレンス・マニュアル

著作権

このドキュメントは以下の貢献者により著作権所有 © 2010-2018 されています。あなたは、GNU General Public License (<http://www.gnu.org/licenses/gpl.html>) のバージョン 3 以降、あるいはクリエイティブ・コモンズ・ライセンス (<http://creativecommons.org/licenses/by/3.0/>) のバージョン 3.0 以降のいずれかの条件の下で、配布または変更することができます。

このガイドの中のすべての商標は、正当な所有者に帰属します。

* 貢献者 *

Jean-Pierre Charras, Fabrizio Tappero.

翻訳

Asuki Kono <asukiaaa At gmail.com>, 2018. starfort <starfort AT nifty.com>, 2015-2019. 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://gitlab.com/kicad/services/kicad-doc/issues>
- KiCad ソフトウェアについて: <https://gitlab.com/kicad/code/kicad/issues>
- KiCad ソフトウェアの翻訳について: <https://gitlab.com/kicad/code/kicad-i18n/issues>

発行日とソフトウェアのバージョン

2015 年 5 月 30 日発行

Chapter 1

Eeschema 入門

1.1 概要

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ーソフトウェアであり、次のオペレーティングシステムで利用可能です。:

- Linux
- Apple macOS
- 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、ライブラリー管理および、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 の電氣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、シンボルの作成/編集やライブラリーの管理ができるシンボルライブラリーエディターを含んでいます。また、現代の回路図エディターソフトに必要な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電気リカル・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 か XSLT の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より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、いくつかの方法で複数のシートからなる回路図を扱うことができます:

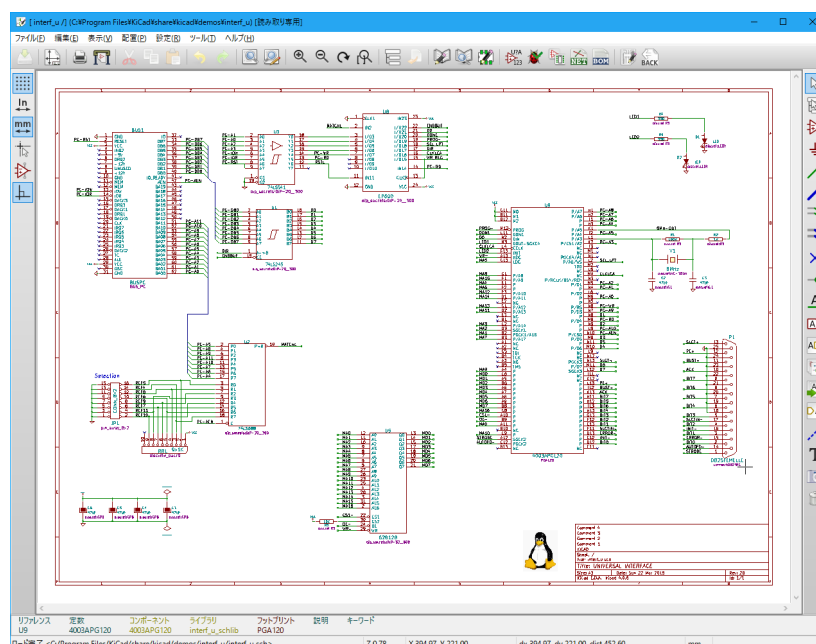
- 単一の階層 (各図が一度だけ使用される)。
- 複雑な階層 (いくつかの図は、一度以上の複数回使用される)。
- 平面的 (フラット) な階層 (マスター図面の中、いくつかの図面は明示的に接続されない)。

Chapter 2

Eeschema コマンド全般

下記のコマンドを実行できます:

- 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, Insert, スペース・キー）。特に“Esc”キーは、多くの実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製します。
- ホットキーを入力すると、カーソルがある場所に対して処理が行われます。ホットキーのリストは”ヘルプ” → ”ホットキーリスト” をメニューから選択するか、“Ctrl+F1”を入力することで表示できます。



2.1 マウスコマンド

2.1.1 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シンボル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: シンボル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。

2.1.2 ブロックの操作

Eschema 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すると、範囲の選択ができます。

“Shift”、“Ctrl”、“Shift + Ctrl” を押しながら範囲を選択すると、選択した範囲をコピー、ドラッグ、削除することができます:

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するか Esc キーを押して取消す。

ブロック移動コマンドが実行されていても、マウスの右ボタンからポップアップメニューを開くことで、他のブロックコマンドを選択できます。



2.2 ホットキー

- “Ctrl+F1” で現在のホットキーのリストを表示します。
- ホットキーの割り当てを変更したい場合は、回路図エディターのオプションダイアログ（設定 → 一般オプション）のコントロールタブで設定できます。

以下はデフォルトのホットキーのリストです:

ヘルプ (このウィンドウ)	Ctrl+F1
ズーム・イン	F1
ズーム・アウト	F2
再描画をズーム	F3
中心へズーム	F4
スクリーンにフィット	Home
選択範囲をズーム	@
ローカル座標をリセット	Space
アイテムを編集	E
アイテムを削除	Del
アイテムを回転	R
アイテムをドラッグ	G
元に戻す	Ctrl+Z
やり直し	Ctrl+Y
マウスの左クリック	Return
マウスの左ダブル・クリック	End
回路図を保存	Ctrl+S
回路図をロード	Ctrl+O
アイテムを検索	Ctrl+F
次のアイテムを検索	F5
次の DRC マーカーを検索	Shift+F5
検索と置換	Ctrl+Alt+F
最後のアイテムを繰り返す	Ins
ブロックを移動 → ブロックをドラッグ	Tab
ブロックをコピー	Ctrl+C
ブロックをペースト	Ctrl+V
ブロックを切り取り	Ctrl+X
回路図のアイテムを移動	M
シンボルまたはラベルを複製	C
シンボルを追加	A
電源を追加	P
X 軸でミラー	X
Y 軸でミラー	Y
標準シンボルの方向	N
シンボルの定数を編集	V

シンボルのリファレンスを編集	U
シンボルのフットプリントを編集	F
シンボル・エディターで編集	Ctrl+E
配線の開始	W
バスの開始	B
ワイヤー／バスの線を終端	K
ラベルを追加	L
階層ラベルを追加	H
グローバル・ラベルを追加	Ctrl+L
ジャンクションを追加	J
空き端子フラグを追加	Q
シートを追加	S
ワイヤー・エントリーを追加	Z
バス・エントリーを追加	/
図形ポリラインを追加	I
図形テキストを追加	T
回路図から基板を更新	F8
フィールドを自動配置	O
シートから抜ける	Alt+BkSp
ノードを削除	BkSp
接続をハイライト表示	Ctrl+X

全てのホットキーは、ホットキーエディター（設定 → 一般オプション → [コントロール](#)）で割り当てを変更できます。

ホットキーの設定は、設定 → インポートとエクスポート → ホットキーのインポート・エクスポートでインポートやエクスポートを行なえます。

2.3 グリッド

Eeschema では、カーソルは常にグリッド上にあります。グリッドはカスタマイズ出来ます。:

- ポップアップメニューあるいは「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。
- 回路図エディターオプションダイアログ（設定 → 一般オプション）の” カラー” タブで、色を変えられます。
- 表示はツールバーの左側のボタンで切り替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、シンボル・エディター上で回路記号のピンを配置するのに適したグリッド・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、シンボル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。

2.4 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: ズームイン
 - F2: ズームアウト
 - F4 または (マウスを止めたまま) 中央ボタンをクリック: 画面の中央をカーソル位置近辺に設定
- ウィンドウのズーム:
 - マウスホイール: ズームイン/ズームアウト
 - Shift+ マウスホイール: 上/下パン
 - Ctrl+ マウスホイール: 左/右パン

2.5 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) にリセットされます。これは2点間の距離を測る時に役立ちます。

X 160.00 Y 38.10	dx 160.00 dy 38.10 dist 164.47	mm
------------------	--------------------------------	----

2.6 上部メニューバー

上部メニューバーでは、回路図を保存したり、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。

ファイル(F) 編集(E) 表示(V) 配置(P) 設定(S) ツール(T) ヘルプ(H)

2.7 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。:



KiCad がプロジェクトモードで動いているときは、個別のファイルを扱うための最初の 2 つのアイコンは表示されません。

	新規回路図の作成 (単独実行時のみ)
	既存回路図を開く (単独実行時のみ)
	回路図プロジェクトの保存
	ページの設定 (用紙サイズと表題欄の編集)
	プリントダイアログを開く
	現在のシートで最後にコピーか切り取りをした要素, あるいはブロックを貼り付け
	最後の変更を元に戻す
	最後の変更をやり直す
	シンボルとテキストの検索 (ダイアログの呼び出し)
	ダイアログを表示して、テキストの検索と置換
	回路図ビューの再描画
	回路図表示を画面サイズに合わせる
	ズームイン/ズームアウト (画面中央近辺)
	図面図の階層ナビゲータを表示 (ツリー構造 (サブシートがある場合) と階層を表示)
	現在のシートそのまま上層へ (現在のシートをそのままにして、階層を一つ上げる)
	シンボル・ライブラリー・エディターの呼び出し - ライブラリーとシンボルの表示と変更
	シンボルライブラリーブラウザーを表示
	シンボルをアノテーション
	ERC (エレクトリカル・ルール・チェック): 自動で電氣的接続をチェック
	CvPcb を呼び出し、シンボルとフットプリントの関連付ける
	ネットリストをエクスポート (Pcbnew, Spice フォーマット及びその他フォーマット)





	シンボルのフィールドを編集
	部品表 (BOM : Bill of Materials) を生成
	Pcbnew (プリント基板のレイアウト) の実行
	CvPcb か Pcbnew で割り当てられた、フットプリントフィールドの情報をバックインポート

2.8 右ツールバーのアイコン

このツールバーは次のツールを含んでいます:







- シンボル、ワイヤー、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

	有効になっているコマンドやツールをキャンセル。
	配線やネットラベルを違う色でハイライト表示。KiCad がプロジェクトモードで動作している場合は、Pcbnew の対応する導体もハイライト表示されます。
	配置する新しいシンボルを選択するためにシンボル選択ダイアログを表示。
	シンボルセレクトダイアログで電源ポートを選んで配置。
	配線を描画。
	バスを描画。
	ワイヤー - バスエントリーを配置。これらの要素は装飾的な役割だけなので、配線間の接続に使用すべきではありません。
	バス - バスエントリーを配置。これは 2 本のバス同士をつなぐだけです。
	空き端子フラグを配置。接続されないシンボルのピン上に配置します。これは ERC 機能において、ピンが意図的に接続されていないのか、誤って未接続なのかを確認するために使用されます。
	ジャンクション（接続点）を配置。接続状況がはっきりしない 2 本の交差する配線やピンを接続します。
	ローカルラベルを配置。同じシート内でアイテムを接続します。異なるシート間で接続したい場合は、グローバルラベルか階層ラベルを使用してください。
	グローバルラベルを配置。異なるシート間であっても、同じ名前の全てのグローバルラベルは接続されます。
	階層のラベルを配置。階層ラベルを持つサブシートと親シート間を接続します。
	階層シートを作成。サブシートのデータを保存するためには、ファイル名を指定する必要があります。
	サブシートから階層ラベルをインポート。このコマンドは階層サブシートでのみ実行できます。対象となるサブシートの階層ラベルに対応する階層ピンを作成します。
	階層ピンをサブシートに配置。このコマンドは階層サブシートでのみ実行できます。対象となるサブシートがなくても、階層ピンを作成できます。

	図形ラインかポリゴンを配置。装飾用です。ワイヤーと異なり接続されません。
	テキスト・コメントを配置。
	ビットマップ・イメージを追加。
	選択されたアイテムの削除。

2.9 左ツールバーのアイコン

このツールバーは表示オプションを管理します:

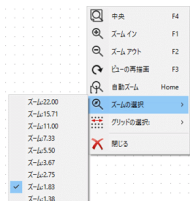
	グリッド表示を切替
	単位をインチに切り替え
	単位を mm へ切り替え
	カーソルの形を選択 (全画面か、小さくか)
	非表示ピンの表示・非表示を切り替え
	配線、バスの自由角度での描画と 90 での描画を切り替え

2.10 コンテキストメニューとクイックエディット

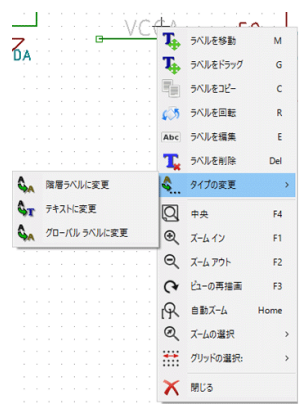
右クリックで選択要素に応じたコンテキストメニューを開き、下記の機能にアクセスします:

- ・ズーム倍率。
- ・グリッド調整。
- ・パラメータ編集。

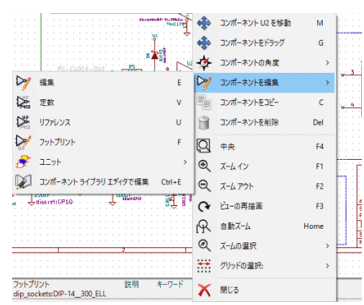
要素を選択しないでコンテキストメニューを呼び出す。



ラベルの編集。



シンボルの編集。



Chapter 3

上部メニューバーの主なメニュー

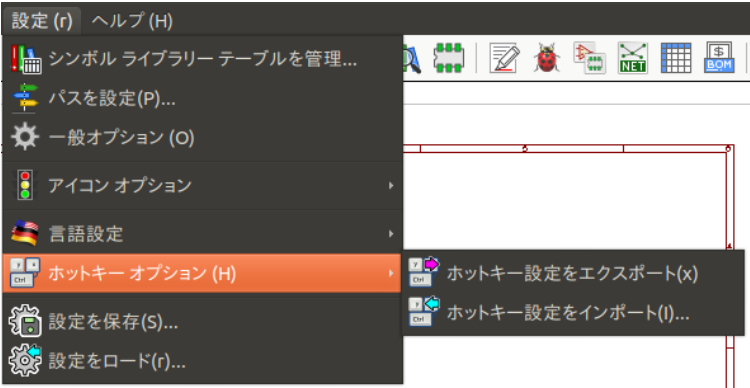
3.1 ファイルメニュー



新規	今の回路図を閉じて、新規回路図の作成（単独実行時のみ）。
開く	回路図プロジェクトを開く（単独実行時のみ）。
最近開いたファイル	最近開いた回路図ファイルのリストから開く。（単独実行時のみ）
回路図シートから追加	現在の回路図へ別のシートの内容を追加する。
KiCad 以外の回路図ファイルを インポート	異なるフォーマットで保存された回路図プロジェクトをインポートしま す。
保存	現在のシートと全ての階層を保存する。
現在のシートを保存	プロジェクト内の現在のシートだけを保存する。
名前を付けて現在のシートを保 存	名前を付けて現在のシートを保存。
ページ設定	ページの寸法とタイトルブロックを設定する。
印刷	回路図を印刷する（ プロットと印刷 の章を参照）。

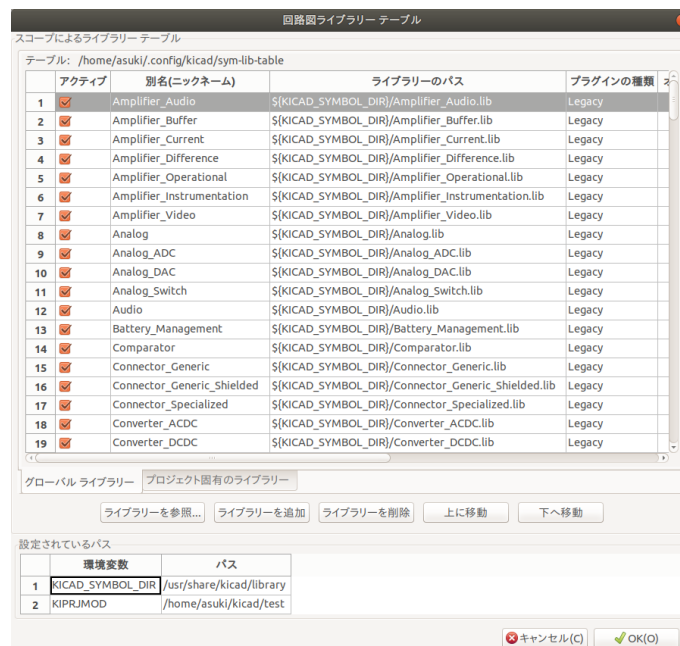
プロット	Postscript HPGL あるいは SVF フォーマットでエクスポートする (プロットと印刷 の章を参照)。
閉じる	アプリケーションを終了する。

3.2 設定メニュー



シンボルライブラリーテーブルを管理	シンボルライブラリーの追加や削除を行います。
パスを設定	検索に利用するパスを設定します。
全般オプション	詳細設定です (単位、グリッドサイズ、フィールド名、など)。
言語設定	インターフェースの言語を選択します。
アイコンオプション	アイコンの見た目を設定します。
インポートとエクスポート	設定項目のファイルへの抽出や、ファイルからの読み取りを行います。

3.2.1 シンボルライブラリーテーブルの管理



Eeschema はスコープの異なる 2 種類のシンボルライブラリーを利用します。

- グローバルライブラリー

グローバルライブラリーテーブルで管理されるシンボルは、全てのプロジェクトで利用できます。このリストは、ホームディレクトリの `sym-lib-table` に保存されます (OS によってパスは異なります)。

- プロジェクト固有のライブラリー

プロジェクト固有のライブラリーテーブルで管理されるシンボルは、そのテーブルに紐づくプロジェクトで利用できます。テーブルは `sym-lib-table` ファイルとして、プロジェクトのディレクトリに保存されます。

下に示す回路図ライブラリーテーブルのタブを切り替えることで、”グローバルライブラリー” か ”プロジェクト固有のライブラリー” を閲覧できます。

3.2.1.1 ライブラリーを追加

ライブラリーを参照.. ボタンをクリックしてファイルを選択するか、* ライブラリーを追加 * をクリックしてライブラリーファイルへのパスを入力することで、ライブラリーを追加できます。選択したライブラリーは、現在のプロジェクトの (グローバルもしくはプロジェクト固有の) ライブラリーテーブルに追加されます。

3.2.1.2 ライブラリーを削除

一つか複数のライブラリーをクリックし ライブラリーを削除ボタンをクリックすることで、ライブラリーを削除できます。

3.2.1.3 ライブラリーのプロパティ

テーブルの各行は、いくつかのフィールドでライブラリーの情報を表示しています:

アクティブ	ライブラリーの有効/無効を。この機能は、一時的にロードするライブラリーを減らしたいときに有効です。
別名（ニックネーム）	短く、ユニークな別名を、シンボルに割り当てられます。シンボルには <ライブラリーのニックネーム>:<シンボル名> という文字列が割り当てられます。
ライブラリーのパス	ライブラリーの場所を示します。
プラグインの種類	ライブラリーのファイルフォーマットを決めます。
オプション	プラグインで利用する場合、ライブラリー固有のオプションを保持します。
説明	ライブラリーについて完結に説明します。

3.2.2 全般オプション

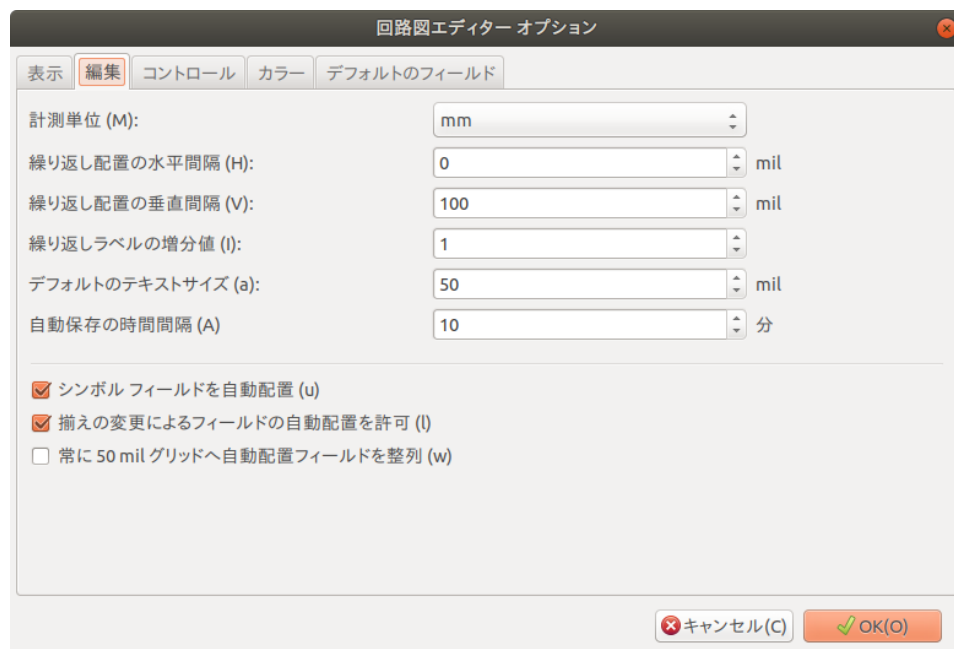
3.2.2.1 表示



グリッドサイズ	グリッドサイズを選べます。 通常グリッド (0.050 インチあるいは 1.27mm) での作業を 推奨します。 より細かいグリッドはコンポーネント作成で使用します。
バス線幅	バスを描画するペンのサイズです。
線幅	ペンのサイズが指定されていないオブジェクトを描画するとき に利用されます。
パーツ ID の表記方法	コンポーネント名を示すために使われる接尾辞の形式です。 (U1A, U1.A, U1-1, etc.)
アイコンスケール	ツールバーのアイコンのサイズを調整できます。
グリッドを表示	グリッドの表示を切り替えれます。

バス、配線を 90 度入力に制限	チェックがある場合、バスや配線は垂直または水平になります。そうでなければ、バス、配線はどのような角度にも配置されます。
非表示ピンの表示	非表示 (または <i>hidden</i>) ピンを表示。チェックがある場合、電源ピンも表示されます。
ページの境界を表示	チェックがある場合、画面上にページの境界が表示される。
シンボルの選択中にフットプリントをプレビュー	シンボルを選択すると、フットプリントのプレビューとフットプリントの選択肢が表示されます。 メモ: 問題の発生や処理が遅くなる可能性があるので、利用は自己責任でお願いします。

3.2.2.2 編集

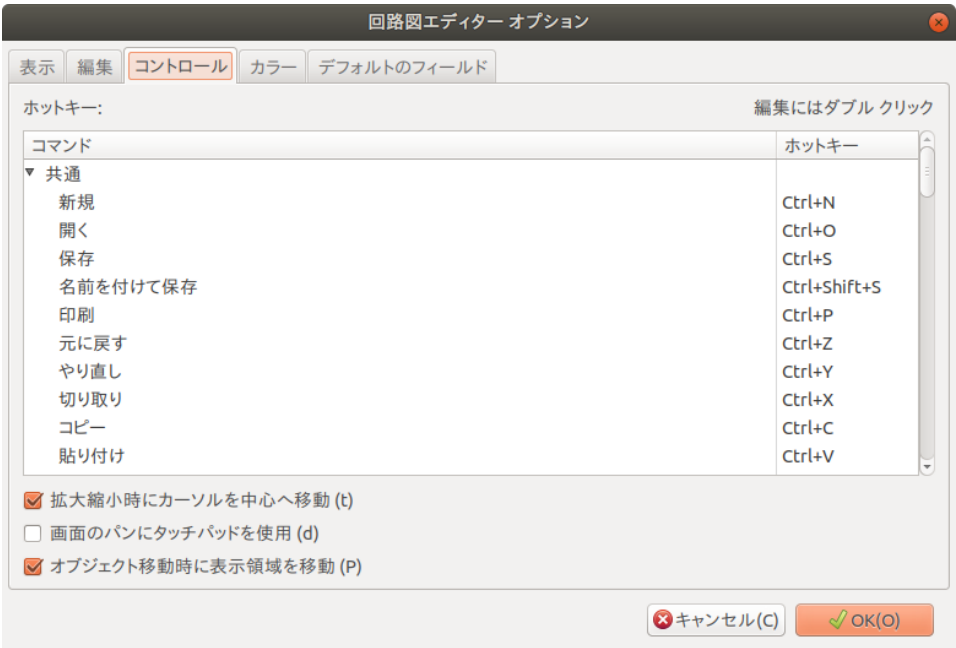


計測単位	表示とカーソル座標の単位（インチまたは mm ）を選べます。
繰り返し配置の水平間隔	アイテムをコピーする際の X 軸の増加値（通常は 0）（シンボル、ラベル、配線などの要素を設置した後に Insert キーを押すと、直前に扱っていた要素をコピーできます。）
繰り返し配置の垂直間隔	アイテムをコピーする際の Y 軸の増加値（通常は 0.100 inch か 2,54 mm）。
繰り返しラベルの増分値	バスの配線のように、数字で終わるラベルの増加量（通常は 1 か -1 ）。
デフォルトのテキストサイズ:	新しいテキストやラベルが新規作成される時に使用されます。
自動保存の間隔	バックアップを保存する間隔（分）。
シンボルフィールドを自動配置	チェックされた場合、シンボルフィールド（値やリファレンス）は他のアイテムと衝突しないように移動して配置されます。

揃えの変更によるフィールドの自動配置を許可	'シンボルフィールドを自動配置'の追加オプションです。新しいアイテムが配置された際に、シンボルフィールドのテキストを揃えます。
常に 50 mil グリッドへ自動配置フィールドを整列	'シンボルフィールドを自動配置'の追加オプションです。チェックされた場合、フィールドは 50 mil のグリッドで配置されます。そうでなければ、自由な場所に配置されます。

3.2.2.3 コントロール

ホットキーの再定義とインターフェースの動作設定を行なえます。



アクションをダブルクリックか右クリックすると、ポップアップメニューが表示されます:

編集	アクションに対する新しいホットキーを定義します。(ダブルクリックするのと同じです。)
変更を元に戻す	ホットキーに関する直近の変更を元に戻します。
デフォルト値を復元	アクションのホットキーをデフォルト値にします。
全ての変更を元に戻す	ホットキーに関する直近全ての変更をもとに戻します。
デフォルト値を全て復元	全てのアクションのホットキーをデフォルト値にします。

オプションの説明:

カーソルを中心にしてズーム	チェックした場合、カーソルの中心が拡大や縮小の中心になります。
画面のパンにタッチパッドを使用	有効にした場合、マウスホイール（またはタッチパッドのジェスチャー）を動かすとビューはパンされ、Ctrl を押しながら動かすとズームします。そうでなければ、基本ズーム、Ctrl か Shift を押しながらだとパンします。

オブジェクト移動時にパン	チェックした場合、配線作業や要素移動中にウィンドウからカーソルが出ようとすると、自動的にウィンドウが追従します。
--------------	--

3.2.2.4 色

様々な要素の描画色です。色見本をクリックすると、その要素の新しい色を選択できます。



3.2.2.5 デフォルトのフィールド

シンボルに配置される追加のカスタムフィールドや関連する値を定義します。



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。

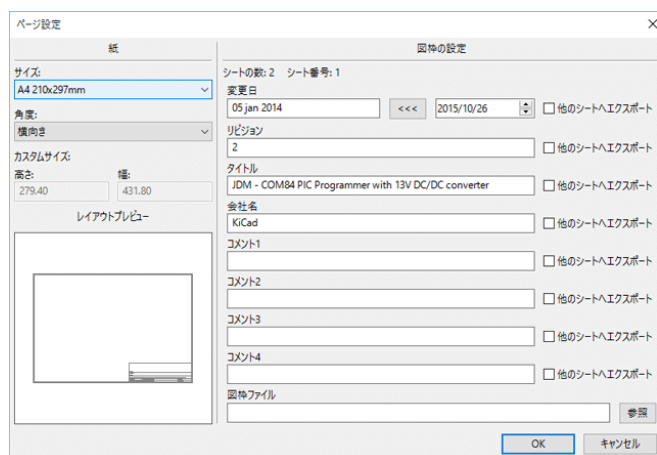
バグレポートを送る時は、ビルドや環境の情報を明らかにするために、“バージョン情報をコピー” を使って下さい。

Chapter 4

上部ツールバーの主なツール


4.1 ページ設定

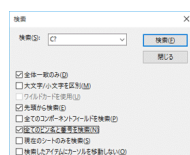
シート設定アイコン  でシートのサイズと右下隅にある表題欄のテキストセクションを定義できます。



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、自動的に更新されません。


4.2 検索ツール

検索アイコン  で検索ツールを開けます。



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。見つかったら、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。

4.3 ネットリストツール

ネットリストアイコン  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。

このツールは、階層全体（通常のオプション）の全ての接続を記述するファイルを作成します。

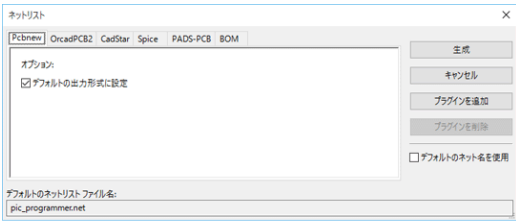
マルチシートの階層において、ローカルラベルは自身が属するシート内だけで通用します。例えば、シート 3 のラベル 1 は、（意図的にそれらを接続していない場合）シート 5 のラベル 1 とは接続されません。これは、シート名がローカルラベルと関連付けられているからです。

注意

Eeschema のラベルのテキストの長さに制限はありませんが、ネットリストが含む値を他のプログラムが利用することを考慮してください。

注意

作成したファイルで、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	チェックすると、デフォルトのフォーマットで Pcbnew を開きます。
------------	-------------------------------------

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）

外部プラグインを追加してネットリストフォーマットリストを拡張できます（下記に示す PadsPcb プラグインなど）。

[ネットリストの作成](#) にネットリスト作成に関する詳しい情報があります。

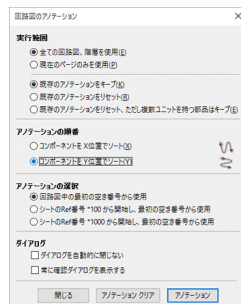
4.4 アノテーションツール



でアノテーションツールを起動できます。このツールはコンポーネントに対してリファレンス（参照番号）の割り付けを行います。

複数ユニットコンポーネント（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を、もしくはまだアノテートされていない新規のコンポーネントだけを、アノテートできます。



実行範囲

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）
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）。
既存のアノテーションを保持	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）。
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）。
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、全ての複数ユニット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（水平方向か垂直方向か）を選択します。

アノテーションの選択

割り当てられたリファレンスのフォーマットを選択。

4.5 エレクトリカル・ルール・チェック (ERC) ツール

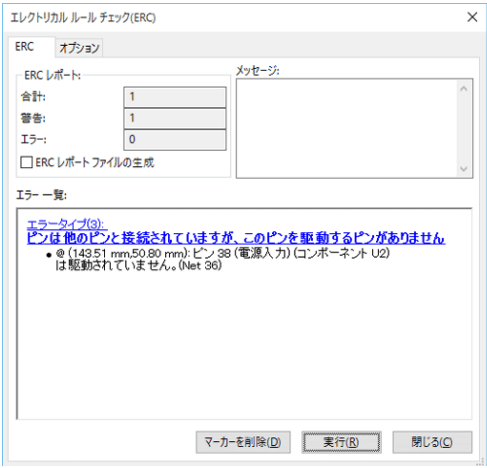


アイコンで、エレクトリカル・ルール・チェック (ERC) ツールを実行できます。

このツールは設計の検証を行い、接続忘れや矛盾を検出できます。

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。

4.5.1 ERC ダイアログ : ERC



エラーは電気的・ルール・チェック (ERC) ダイアログに表示されます：

- 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：

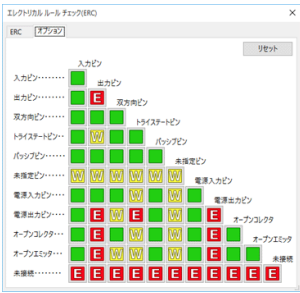
ERC レポートファイルの生成	ERC レポートファイルを生成するには、このオプションをチェックします。
-----------------	--------------------------------------

コマンド：

マーカーを削除	全ての ERC エラー/警告マーカーを削除します。
実行	ERC（電気的ルールチェック）を実行します。
閉じる	ダイアログを閉じます。

- 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。

4.5.2 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定義できます。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。

オプション :

同様のラベルを確認	大文字小文字だけが異なるラベル (例: label/Label/LaBeL) を報告します。ネット名は大文字と小文字を区別するため、このようなラベルは別のネットと認識されます。
固有のグローバルラベルを確認	一度しか使われていないグローバルラベルを報告します。通常、最低 2 つは接続のために使われている必要があります。

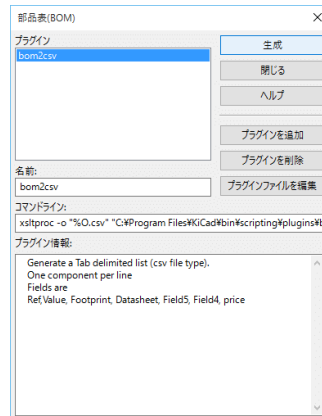
コマンド :

デフォルトへ初期化	もともとの設定に戻します。
-----------	---------------

4.6 部品表 (BOM) ツール



アイコンで、部品表 (BOM) ツールを起動します。このツールで、コンポーネントと階層接続グローバルラベルの一覧表を生成できます。

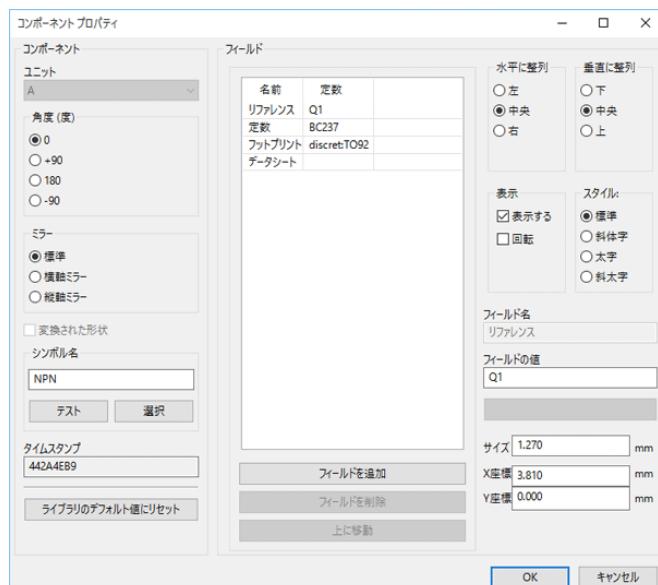


Eeschema の部品表 (BOM) 作成ツールは XSLT または Python の外部プラグインを利用します。KiCad プログラムのあるディレクトリに、いくつかの例がインストールされています。

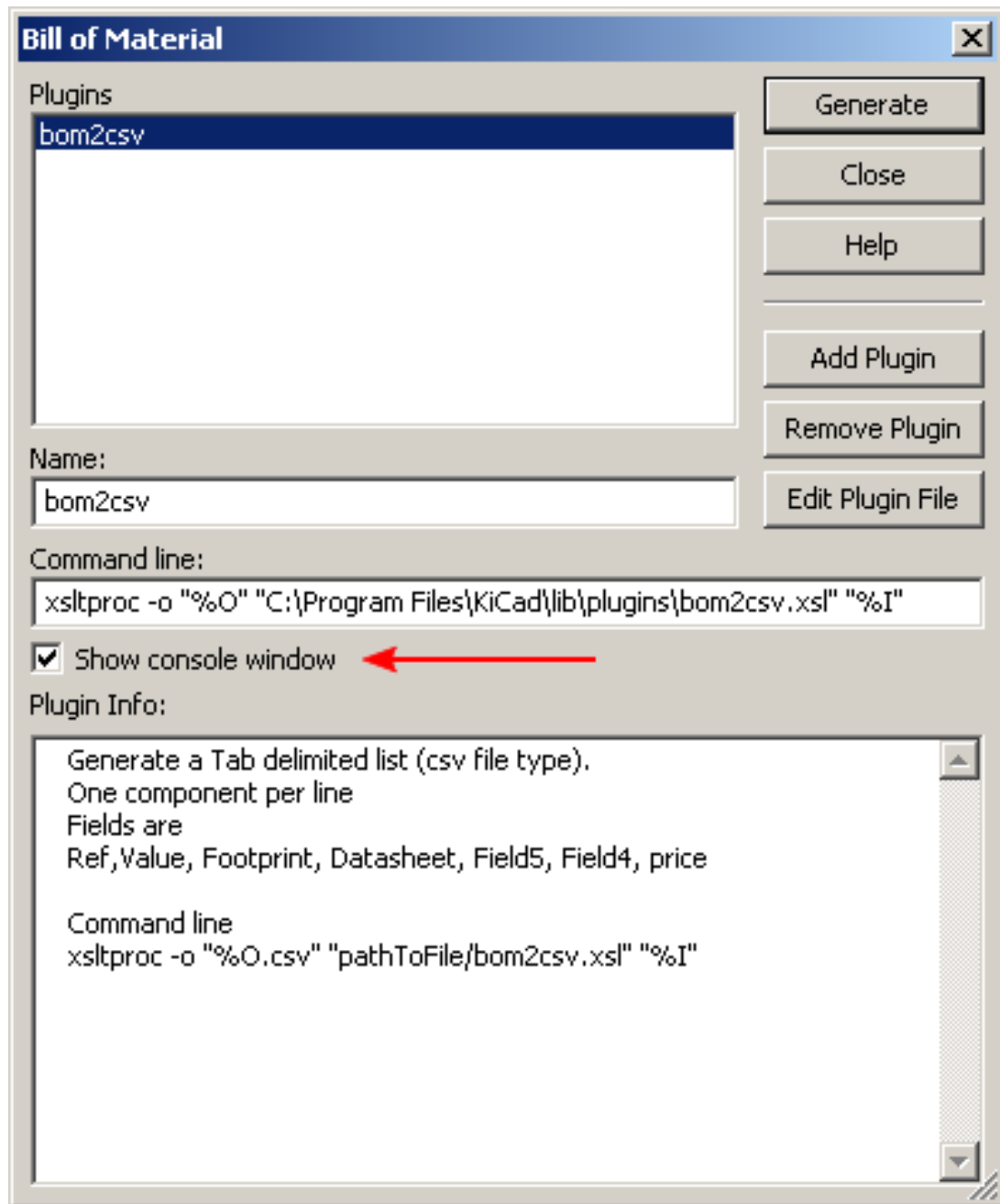
部品表 (BOM) での使用に役立つコンポーネントのプロパティには、以下のものが挙げられます。:

- 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:

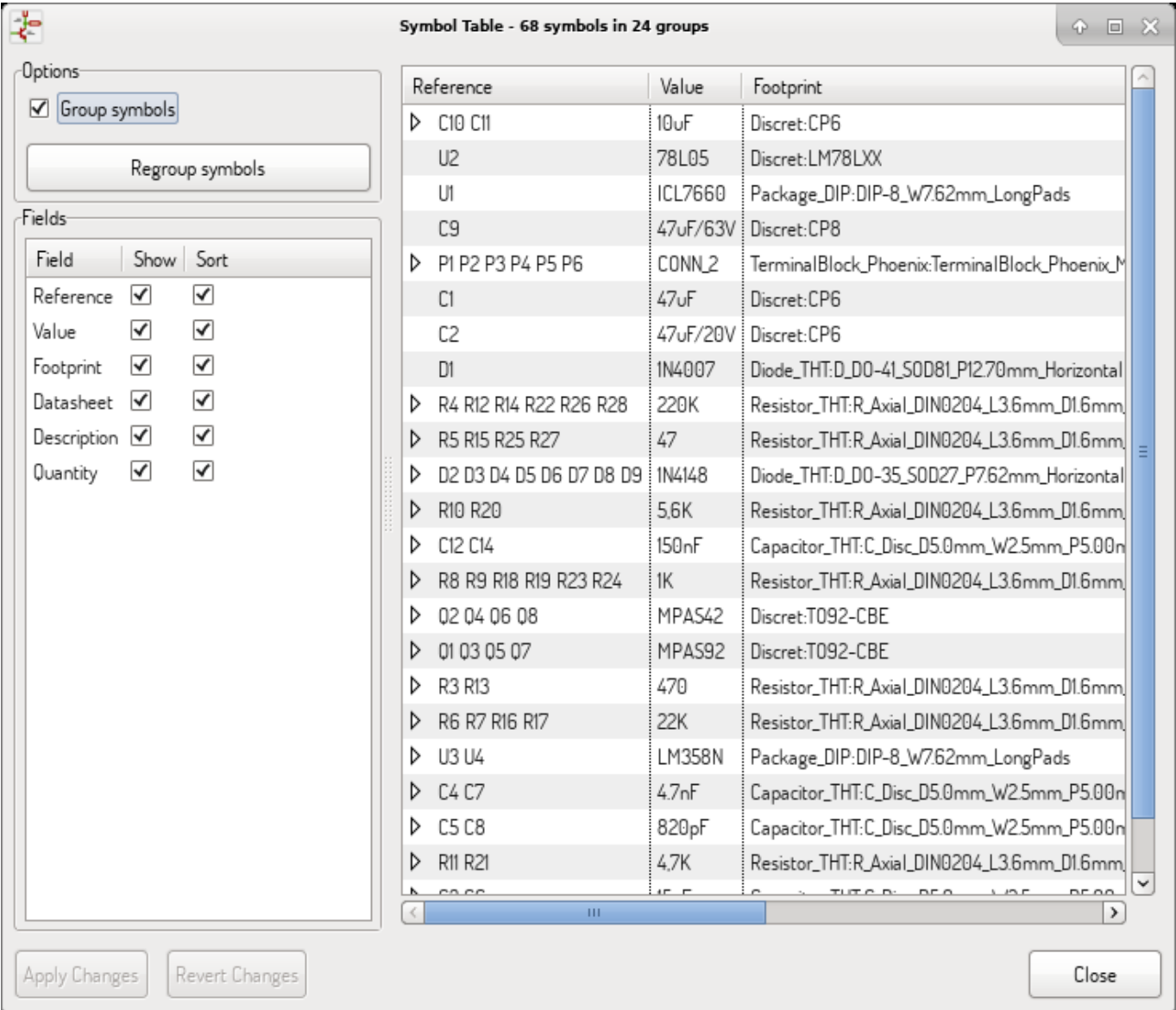


Windows 環境の BOM 作成ダイアログには、（赤い矢印で示している）外部プラグインで表示ウィンドウをコントロールするための特別なオプションがあります。デフォルトの設定では、BOM 作成コマンドは隠れたウィンドウで実行され、*Plugin info* フィールドに結果が表示されます。このオプションを有効にすると、実行中のウィンドウを表示します。GUI でプラグインの動作確認を行う際に、この機能は必要になるでしょう。



4.7 フィールド編集ツール

The icon  アイコンで全てのシンボルを編集や閲覧するためのスプレッドシートを開けます。



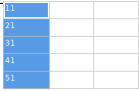
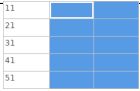
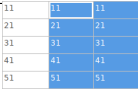
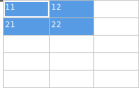
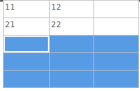
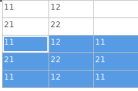
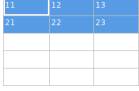
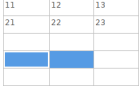
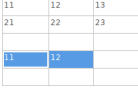
フィールドの値を変更したら、'適用' ボタンをクリックして変更を受理するか、'元に戻す' ボタンをクリックして変更を破棄する必要があります。

4.7.1 簡単にフィールドを埋める裏技

スプレッドシート中で実行する、いくつかの特別なコピー/ペーストの方法があります。これらは少数のコンポーネントで繰り返してフィールドの値を入力する時に役に立ちます。

These methods are illustrated below.


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

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

注意
これらのテクニックは、グリッドコントロールを持っている他のダイアログでも有効です。

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:

 **BACK** アイコンで、バックアノテートツールを起動します。

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

シンボル・ライブラリーの管理

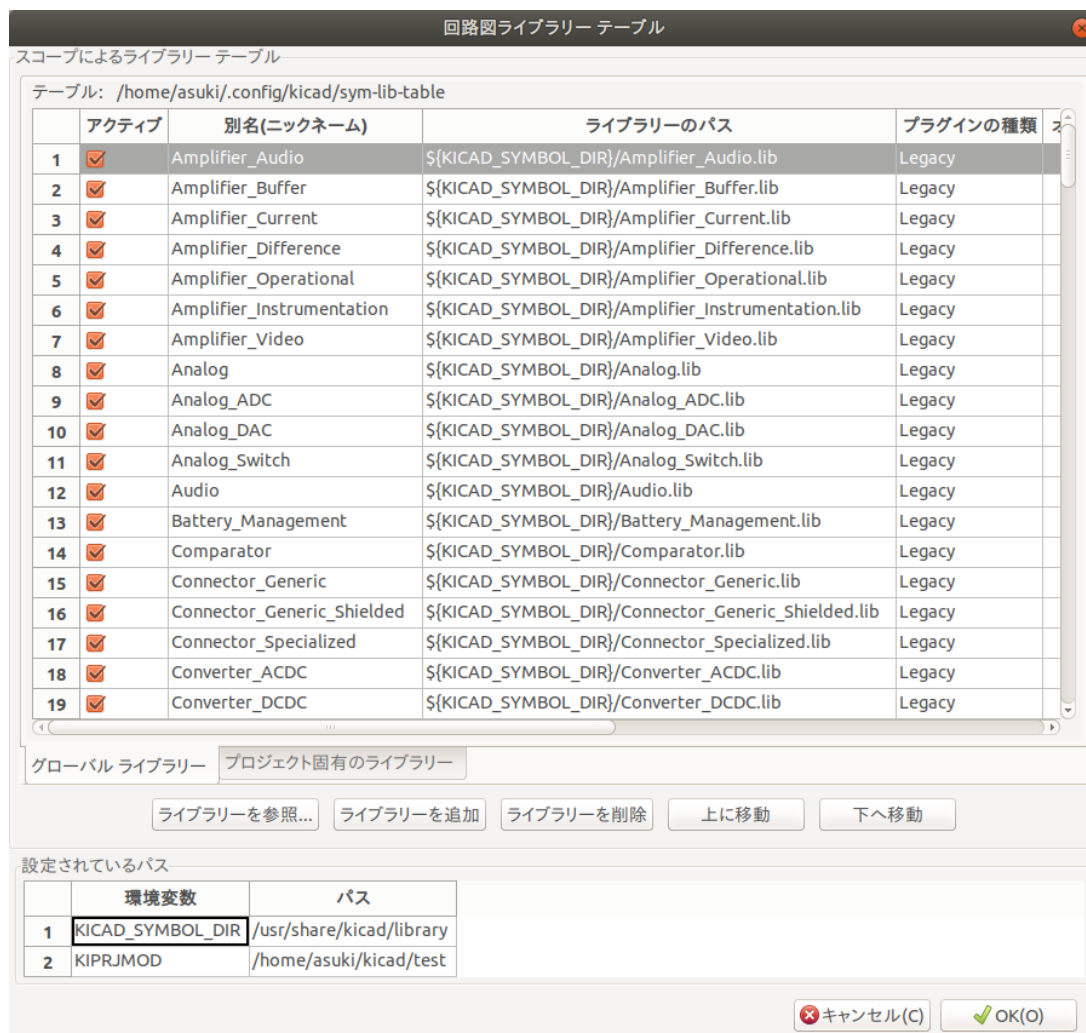
シンボル・ライブラリーは、回路図を作成するときに使われるシンボルのコレクションを保持しています。回路図中の各シンボルは、ライブラリーのニックネームとシンボル名から作られるフルネームで一意的に識別されます。Audio:AD1853 は、その例です。

5.1 シンボル・ライブラリー・テーブル

シンボル・ライブラリー・テーブルは、KiCad が認識している全てのライブラリー・ファイルのリストを保持しています。シンボル・ライブラリー・テーブルは、グローバル・シンボル・ライブラリー・テーブル・ファイルとプロジェクト固有のシンボル・ライブラリー・テーブル・ファイルから構成されます。

シンボルがロードされる時、シンボル・ライブラリー・テーブルでライブラリーの位置を調べるため、Eeschema は、サンプルにある Audio のような、ライブラリーのニックネームを使用します。

下図に、シンボル・ライブラリー・テーブル編集ダイアログを示します。



5.1.1 グローバル・シンボル・ライブラリー・テーブル

グローバル・シンボル・ライブラリー・テーブルは、現在のプロジェクトに関係なく、常に有効なライブラリーのリストを保持しています。このテーブルは、ユーザーのホーム・フォルダーにある `sym-lib-table` ファイルに保存されます。このフォルダーの位置はオペレーティング・システムに依存します。

5.1.2 プロジェクト固有のシンボル・ライブラリー・テーブル

プロジェクト固有のシンボル・ライブラリー・テーブルは、現在のプロジェクト・ファイルで明示的に有効となるライブラリーのリストを保持しています。プロジェクト固有のシンボル・ライブラリー・テーブルは、プロジェクト・ファイルと一緒にロードされている時のみ、編集可能です。プロジェクト・ファイルが読み込まれていないか、プロジェクトのパスにシンボル・ライブラリー・テーブルがない場合、編集可能な空のテーブルが作られ、後でプロジェクト・ファイルと一緒に保存されます。

5.1.3 初期設定

Eeschema の初めての実行時には、グローバル・シンボル・テーブル・ファイル `sym-lib-table` はユーザーのホーム・フォルダーに見つかりません。Eeschema は、システムの KiCad テンプレート・フォルダーにあるデフォル

トのシンボル・テーブル・ファイル `sym-lib-table` をユーザーのホーム・フォルダーへ `sym-lib-table` ファイルとしてコピーしようとします。もし、デフォルトのテンプレート `sym-lib-table` ファイルが見つからなかった場合、`sym-lib-table` ファイルの位置指定を変更するためのダイアログが表示されます。`sym-lib-table` が見つからないか、ダイアログが使われなかった場合、ユーザーのホーム・フォルダーには空のシンボル・ライブラリー・テーブルが作られるでしょう。こうなった場合、ユーザーは自分で `sym-lib-table` をコピーし、手作業でテーブルを設定できます。

注意

デフォルトのシンボル・ライブラリー・テーブルは、KiCad の一部としてインストールされる標準のシンボル・ライブラリーを全て含んでいます。システムの実行速度と使用状況に依存することは、望ましいこととは限りません。シンボル・ライブラリーの読み込みに必要な時間は、シンボル・ライブラリー・テーブルにあるライブラリーの数に比例します。もしシンボル・ライブラリーの読み込み時間が長すぎるなら、グローバル・ライブラリー・テーブルから全く使わない、あるいは滅多に使わないライブラリーを削除し、必要に応じてプロジェクト・ライブラリー・テーブルに削除したライブラリーを追加して下さい。

5.1.4 テーブル要素の追加

シンボル・ライブラリーを使うには、まず最初にグローバル・テーブルかプロジェクト固有のテーブルを追加しなければなりません。プロジェクト固有のテーブルは、プロジェクト・ファイルが開かれた時のみ有効です。

各ライブラリーのエントリーは固有のニックネームを持つ必要があります。

ニックネームは実際のライブラリー・ファイル名やファイル・パスとは全く関係ありません。コロン : と / 文字はニックネーム内のいかなる場所でも使用できません。各ライブラリーのエントリーは、そのライブラリーの種類で有効なファイル・パス、ファイル名を持つ必要があります。パスは、絶対、相対、または環境変数で指定できます。(下記セクション参照)

プラグインの種類は、ライブラリーが正しく読み込まれるよう、適切に選択しなければなりません。今のところ、KiCad は以前のシンボル・ライブラリー・ファイルのプラグインのみ、サポートしています。

説明フィールドは、ライブラリー・エントリーの説明を追加するためのものです。オプション・フィールドは現在のところ使われておらず、追加されたオプションはライブラリーがロードされても何も影響を与えません。

- 同じテーブルには重複したライブラリーのニックネームを持つことが出来ないことに注意して下さい。しかしながら、グローバルとプロジェクト固有のライブラリー・テーブルの両方で重複したライブラリーのニックネームを持つことは可能です。
- もし名前の衝突が起こった場合、プロジェクト固有のテーブル・エントリーがグローバル・テーブル・エントリーに優先します。
- エントリーがプロジェクト固有のテーブルに定義されている場合、エントリーを含んだ `sym-lib-table` は現在開かれているプロジェクト・ファイルのあるフォルダーに書き込まれます。

5.1.5 環境変数の代替

シンボル・ライブラリー・テーブルの最も強力な機能の一つは、環境変数の代替です。環境変数に保存されたシンボル・ライブラリーへのパスを独自に定義することができます。環境変数の代替は、ライブラリー・パスで `${ENV_VAR_NAME}` 構文を使うことにより、サポートされます。

デフォルトでは、KiCad は実行時に 2つの環境変数を定義します:

- `KIPRJMOD`: 常に現在開いているプロジェクトのディレクトリーを指す環境変数。`KIPRJMOD` は変更できません。
- `KICAD_SYMBOL_DIR`: KiCad と一緒にインストールされたデフォルトのシンボル・ライブラリーのパスを指す環境変数。

設定/ パス設定を定義することで `KICAD_SYMBOL_DIR` を上書きできます。これにより、デフォルトの KiCad シンボル・ライブラリーの代わりに自分自身のライブラリーを使うことができます。

`KIPRJMOD`: この環境変数により、プロジェクト固有のシンボル・ライブラリー・テーブルにあるライブラリーへの絶対パス (いつも既知とは限らない) を定義せずにプロジェクトのパスにライブラリーを保存することができます。

5.1.6 使用パターン

シンボル・ライブラリーは、読み込まれているプロジェクトに対して、グローバル、固有どちらとしてでも定義できます。ユーザーのグローバル・テーブルで定義されたシンボル・ライブラリーは常に有効で、ユーザーのホーム・フォルダーにある `sym-lib-table` ファイル内に保存されます。プロジェクト固有のシンボル・ライブラリー・テーブルは、現在開かれているプロジェクト・ファイルに対してのみ有効です。

各方法には長所と短所があります。全てのライブラリーをグローバル・テーブルに定義すると、必要な時にいつでも使うことができます。この短所は、読み込み時間が長くなることです。

全てのシンボル・ライブラリーをプロジェクト固有のテーブルへ定義すると、プロジェクトが必要とするライブラリーだけとなるので、シンボル・ライブラリーの読み込み時間が短くて済みます。この短所は、プロジェクトごとに必要とされるシンボル・ライブラリーをそれぞれ忘れずに追加しなければならないことです。

使用パターンの一つは、よく使うライブラリーをグローバル、そのプロジェクトでのみ必要とされるライブラリーはプロジェクト固有のライブラリー・テーブルに定義することでしょう。ライブラリーを定義するにあたっての制約は特にありません。

5.1.7 以前のプロジェクトのリマッピング

シンボル・ライブラリー・テーブルの実装以前に作られた回路図を読み込んだとき、Eeschema は回路図にあるシンボルのライブラリー・リンクを適切なライブラリー・テーブルのシンボルにリマップしようと試みます。このプロセスの成功は、いくつかの要素に左右されます。:

- 回路図で使われているオリジナルのライブラリーがまだ有効であり、シンボルが回路図に追加された時から変更されていない。
- レスキュー・ライブラリーを作成するか、既存のレスキュー・ライブラリーを最新に保っていることを確認したときに、全てのレスキュー動作が実行されている。
- プロジェクトのシンボル・キャッシュ・ライブラリーが完全な状態のままで、破損していない。



警告

リマッピングでは、プロジェクト・フォルダーにあるレスキュー・バックアップ・フォルダーの中にリマッピング中に変更された全てのファイルをバックアップします。何か問題が起きた時に備えて、リマッピングの前には常にプロジェクトをバックアップして下さい。

**警告**

適切なシンボルをリマッピングで得ることを保証するために無効化されていたとしても、レスキュー動作は実行されます。この動作をキャンセルしてはいけません。さもないと、リマッピング動作は、回路図のシンボルを正しくリマップすることに失敗します。

注意

オリジナル・ライブラリーが削除され、レスキューが実行されていない場合、最終手段としてキャッシュ・ライブラリーがリカバリー・ライブラリーに使われます。新しいファイル名でキャッシュ・ライブラリーをコピーし、シンボル・ライブラリー・テーブルが実装される以前のバージョンの Eeschema を使ってライブラリー・リストの先頭に新しいライブラリー・ファイルを追加します。

Chapter 6

回路図の作成と編集

6.1 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 Eschema 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。

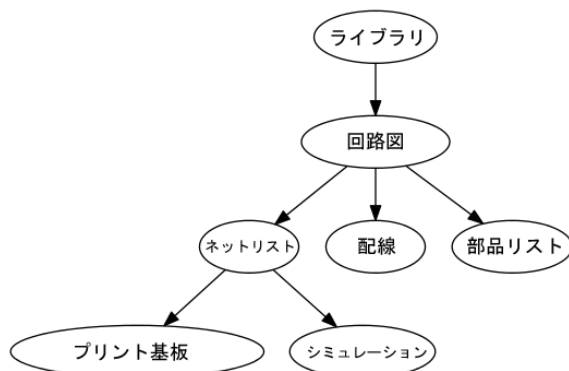
6.2 基本的な検討事項

Eschema 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](#)

回路図は主にシンボル、ワイヤー、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、バスエントリー、コメント、破線などのグラフィック要素も配置できます。

6.3 開発フロー



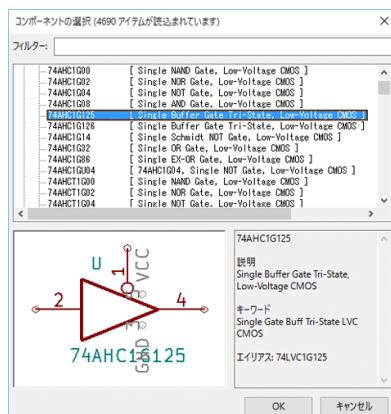
シンボルはシンボルライブラリーから回路図へと配置されます。回路図が完成した後で、PcbNew が回路の接続情報とフットプリントをインポートできるように、ネットリストが作られます。

6.4 シンボルの配置と編集

6.4.1 シンボルの検索と配置



アイコンを使って、回路図にシンボルをロードします。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。



“シンボルの選択” 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シンボルをフィルターリングできます。以下の入力により、アドバンスドフィルターを使用することが可能です：

- **ワイルドカード:** 文字 `?` と `*` を使用します。これらは”任意の一文字”と”長さ 0 以上の任意の文字列”を意味します。
- **関係型:** ライブラリーの部品の説明またはキーワードがフォーマット”Key:123” のタグを含んでいる場合、”Key>123” (以上)、”Key<123” (以下) 等のように入力することで、それに応じて一致させることができます。数字は以下の大文字・小文字を区別しない接尾辞の何れかを含めることができます：

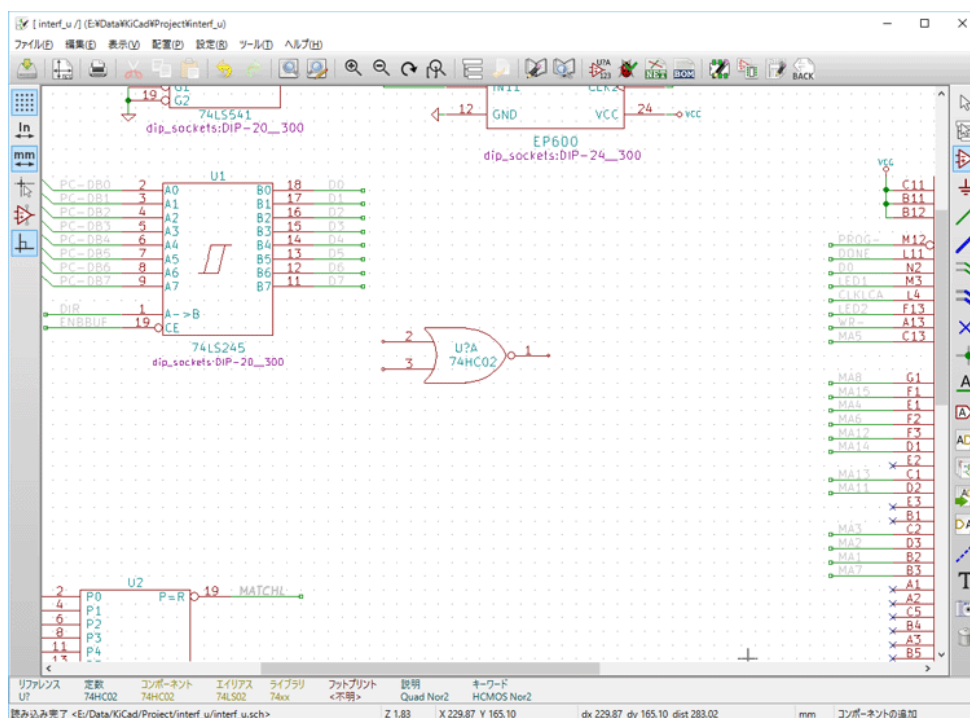
=== | p | n | u | m | k | meg | g | t | 10⁻¹² | 10⁻⁹ | 10⁻⁶ | 10⁻³ | 10³ | 10⁶ | 10⁹ | 10¹² ===

=== | ki | mi | gi | ti | 2¹⁰ | 2²⁰ | 2³⁰ | 2⁴⁰ ===


- 正規表現: もし正規表現に精通しているのであれば、これらを使うこともできます。使用されている正規表現のタイプは **wxWidgets Advanced Regular Expression style** で、Perl の正規表現と似たような形式です。

シンボル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、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行うことができます。

配置中のシンボルは次のようになります:



6.4.2 電源ポート (コンポーネント)

電源ポートもシンボルのひとつです (“power” ライブラリーに分類されています)。よって、これまで説明したシンボル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” ライブラリーを探して直接参照します。

6.4.3 (配置された) シンボルの編集と変更

シンボルの編集には、以下の 2 種類があります:

- シンボル自身の変更: 部品の位置、向き、複数ユニットを持つシンボルのユニット選択。
- シンボルフィールドの変更: リファレンス、定数、フットプリント、等。

シンボルを配置したら、それらの部品定数を変更できます (特に抵抗やコンデンサなど)。しかし、(シンボルをユニットで固定したい場合は、手でリファレンスを割り当てる必要がありますが、) 配置したシンボルへのリファレンスの割り当てやユニットの選択は、必ずしも必要なことではありません。なぜなら、これらシンボル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るからです。

6.4.3.1 シンボルの変更

シンボルの設定を編集するには、マウスカーソルをシンボル上へ移動させ、以下のいずれかの操作をします。:

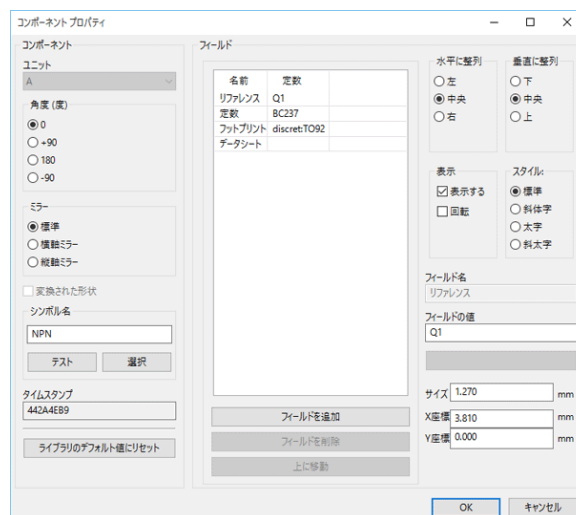
- シンボルをダブルクリックして、全ての設定を編集できるダイアログを開く。
- 右クリックしてコンテキストメニューを開き、表示された編集コマンドを選択する: 移動、回転、編集、削除など。

6.4.3.2 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、シンボルをダブルクリックし、シンボルシンボル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）シンボルに対するもので、シンボルのアンカー位置への相対座標で指定します。“ライブラリーのデフォルト値にリセット” ボタンは、シンボル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。

6.5 ワイヤー、バス、ラベル、電源ポートの接続

6.5.1 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:

この点は、ワイヤーに接しているか、ピンの接続位置に重なっていません。

注 2:

接続を確立するためには、ワイヤー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ーがピンを乗り越えた場合）、これらは接続されません。

注 3:

交差したワイヤー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、ワイヤー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。

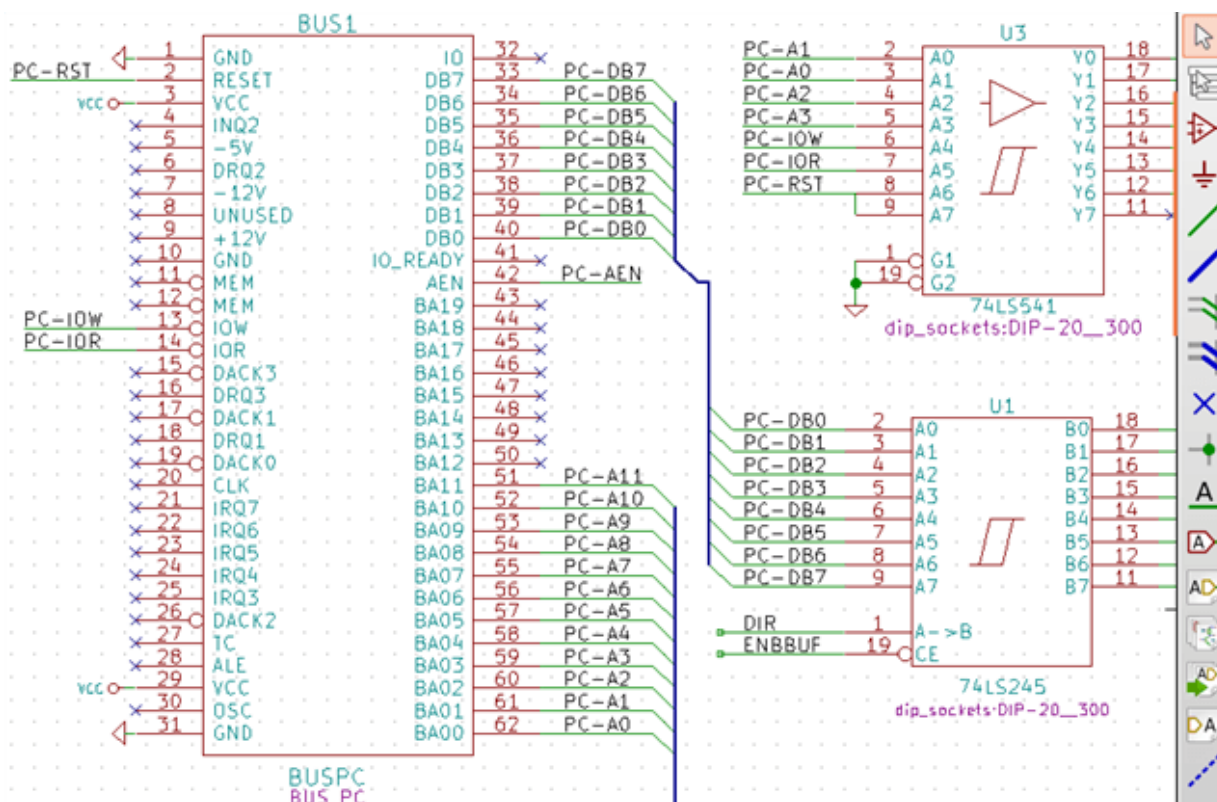
以前に示した図（ワイヤーが DB25FEMALE 22, 21, 20, 19 ピンに接続されているもの）では、このジャンクション（接続点）シンボルを使った場合を示しています。

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。

6.5.3 バスの接続

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。



6.5.3.1 バスのメンバー

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。

6.5.3.2 バスメンバー同士の接続

バスの同一メンバー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ーへ配置されたラベルによってピン間が接続されます。バスワイヤーへのバスエントリー部 (45 度曲がっているワイヤー部分) 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ー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ーを配置します。同様に繰り返しコマンドを利用し、ラベルの下へワイヤーを配置していきます。
- 必要に応じて、同じ方法 (最初のエントリーを配置し、繰り返しコマンドを使用する) を用い、バスエントリーを配置して下さい。

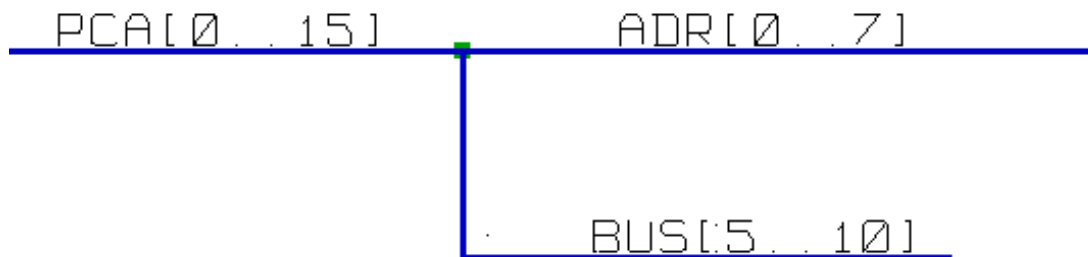
注意

メインメニュー “設定” → “回路図エディターオプション” 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))
-

6.5.3.3 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)

6.5.4 電源ポートの接続


シンボル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シンボル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、下記の理由で少し厄介です:

- 電源ピンが非表示であるため、ワイヤーを接続できない。
- 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。

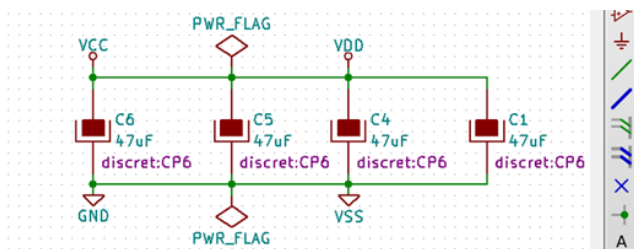
注意

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” → “回路図エディターオプション” 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。

Eschema 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、“power” シンボルライブラリーに収められています。



6.5.5 “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。

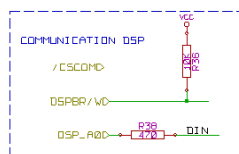
ピンを無接続の空き端子としたい場合、空き端子フラグ (ツール ) を配置することが必要です。このシンボルは生成するネットリストに影響を与えません。

6.6 回路図作成に関する補足


6.6.1 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ーとは異なり、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。



6.6.2 シートの表題欄 (タイトルブロック)

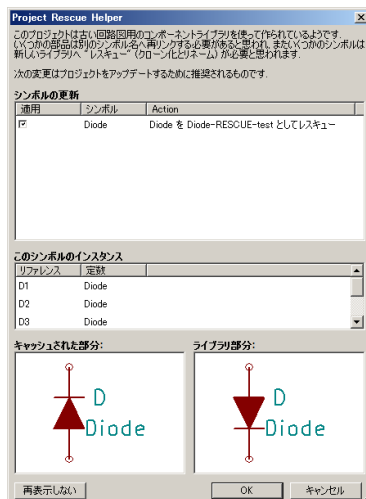
表題欄は、 “ページの設定” ツールで編集することができます。

シート数Y、シート番号X (Sheet X/Y) は自動更新されます。

6.7 キャッシュされたシンボルのレスキュー

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ーからシンボルを読み込みます。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。: ライブラリーにあるシンボル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、プロジェクトのシンボル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。

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ーも一緒に保存されます。これにより、全てのライブラリーがなくてもプロジェクトを配布できます。キャッシュとシステムライブラリーの両方に存在するシンボル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ーを調べます。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、現在のライブラリー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、古いシンボルは特別な“レスキュー”ライブラリーへと保存され、全てのシンボル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シンボル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“キャッシュされたシンボルのレスキュー”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シンボルを読み込むようになります。このオプションは、ライブラリーの設定で元に戻せます。

Chapter 7


階層回路図

7.1 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り解決策となります。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、回路図で使われるシンボルの実体を表すもので、実際にはシンボル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ー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。
-


Eschema はこれらの階層を全て扱うことが可能です。

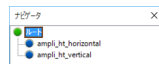
階層回路図の作成は簡単です。階層全体はルート回路図から始まるように管理され、ただ一つの回路図しかないので見えます。

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。
- サブシート間の電氣的な接続方法。

7.2 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を行えます。






シート名をクリックすると、そのシートに移動できます。シート名を右クリックして”シートに入る”を選択するか、境界線をダブルクリックしても、シートに移動できます。

現在のシートから抜けて親のシートに移動するには、回路図の何もオブジェクトが無いところで右クリックして”シートから抜ける”を選択するか、Alt Backspace を入力します。

7.3 ローカルラベル、階層ラベル、グローバルラベル

7.3.1 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。

7.4 階層作成の要約


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つの回路図間に電氣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。

7.5 シートシンボル

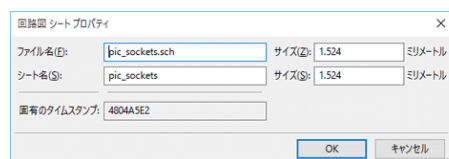
対角上の2点を指定して作成した矩形でサブシートを表します。

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。

これらのラベルは、通常のシンボルのピンに似ています。  “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必要されます。



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）。

7.6 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シンボルのピンに似ていますが、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：

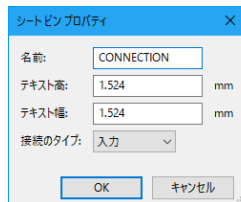
- 必要なピンをサブシート作成前に配置（手動による配置）。
- 必要なピンおよびグローバルラベルをサブシート作成後に配置（半自動配置）。

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”という名前の階層ピンを作成する例です：



階層ピンの作成時あるいは作成後に右クリックしてコンテキストメニューの“シートピンを編集”を選択するとシートピンプロパティが開き、ピンの名前と接続のタイプを定義することができます。


シート内では、階層ラベルは階層ピンと同じ名前にプリセットされなければなりません。手動配置では正しくこれらの名前を合わせるために十分な配慮が必要となりますが、これが下記の第2の方法がより好ましい理由となっています。

自動配置：

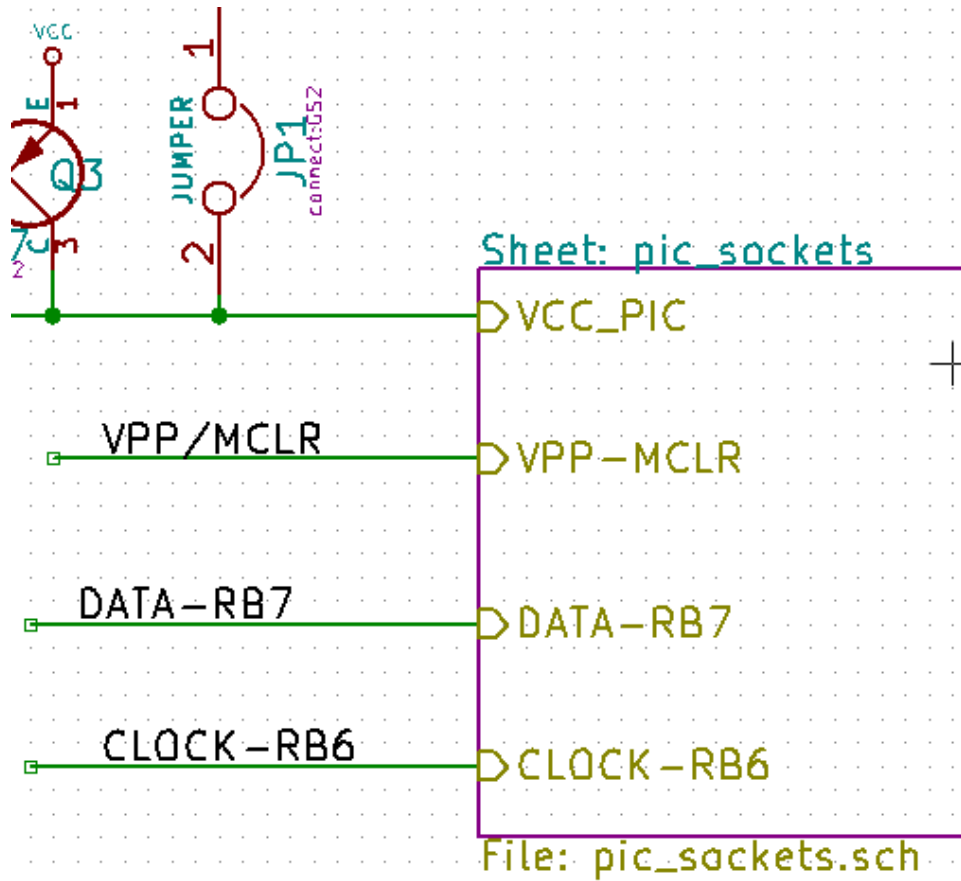
-  “シート内の対応する階層ラベルからインポートされた階層ピンを入力” ツールを選択します。
- 階層シンボルをクリックして、そこからグローバルラベルに対応するピンをインポートして対応する回路図に配置します。新しいグローバルラベルが存在する場合、つまり配置済みのピンに対応したものでないなら、階層ピンが現れます。
- このピンを配置したい場所でクリックします。

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。

7.7 接続 - 階層ラベル

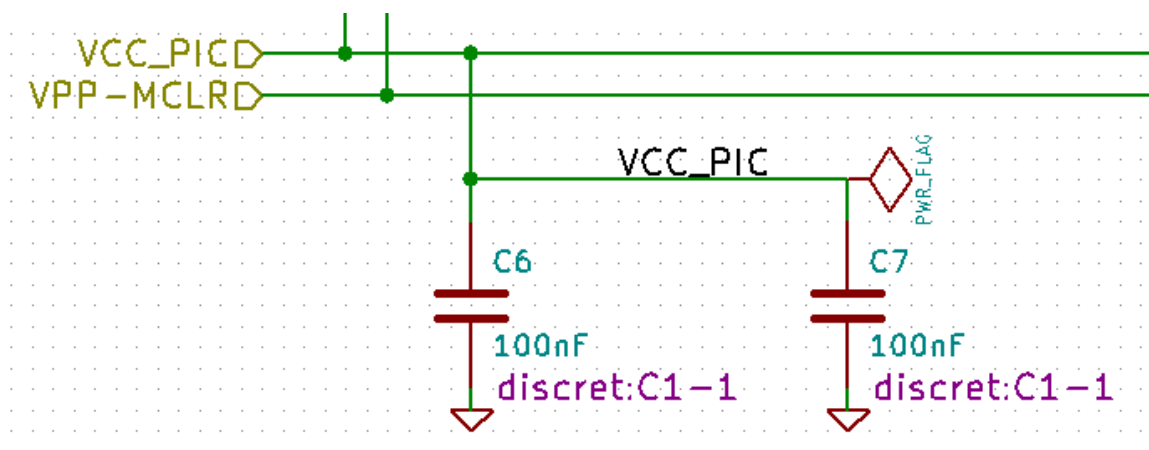
作成したシートシンボルの各ピンはサブシート内の階層ラベルと一致していなければなりません。階層ラベルはラベルと似ていますが、サブシートおよびルートシート間の接続を行います。2つの相補的なラベル（ピンと HLabel）のグラフィカルな表示は似ています。階層ラベルの作成は、 “階層のラベル入力” ツールで行います。

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。

7.7.1 (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

ワイヤーによる接続以外に、接続を行う様々な方法について説明します。

7.7.1.1 (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。

7.7.1.2 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、同一シート内で階層ラベルの “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、別のシートの階層ラベルあるいは “TOTO” というラベルには接続されません。

階層ラベル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のシンボルのピンに接続されていると見なされます。

7.7.1.3 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なされます。このため、“非表示の電源ピン” として宣言されている VCC という名前の電源ピンは、それが置かれているシートにある VCC という名前の全てのピンに接続されます。

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 ラベルを回路図全体の VCC 接続したいなら、VCC 電源シンボル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。

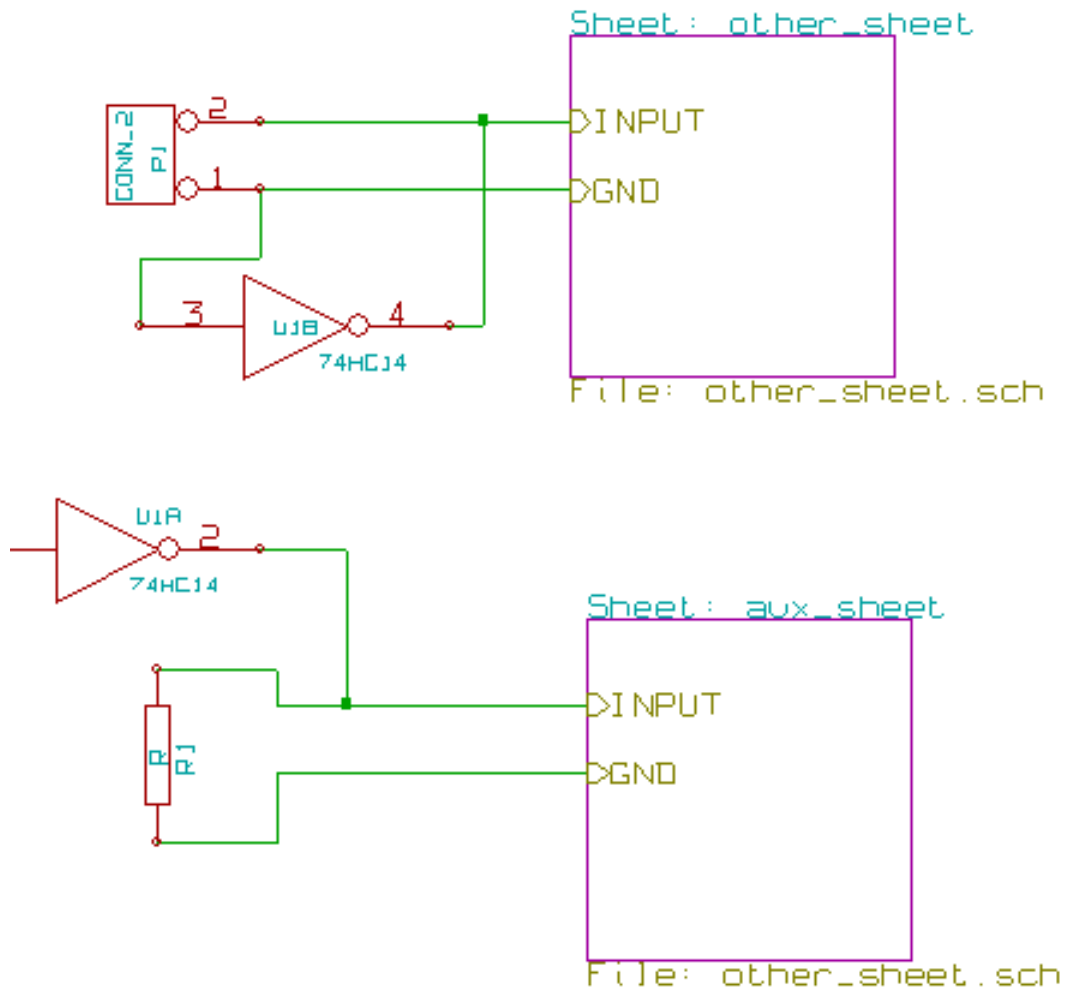
7.7.2 グローバルラベル

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)

7.8 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。シート名は固有でなければなりません。

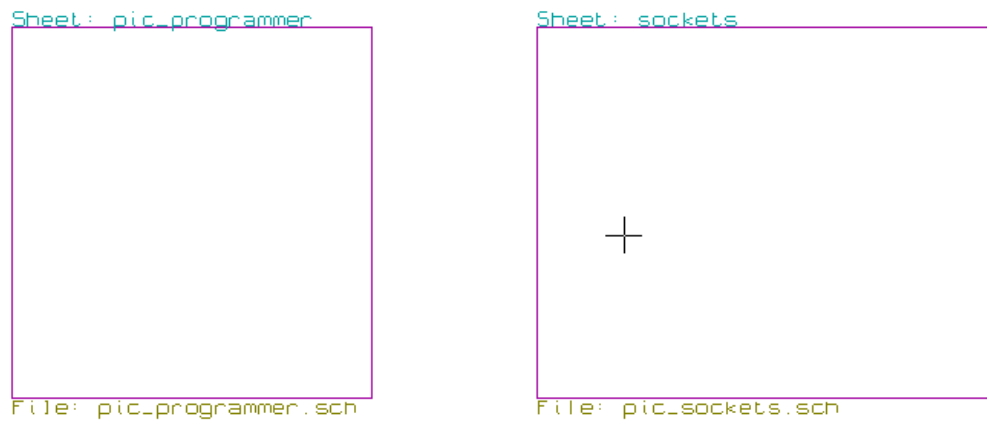


7.9 平面階層

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。それには下記のルールを順守して下さい:

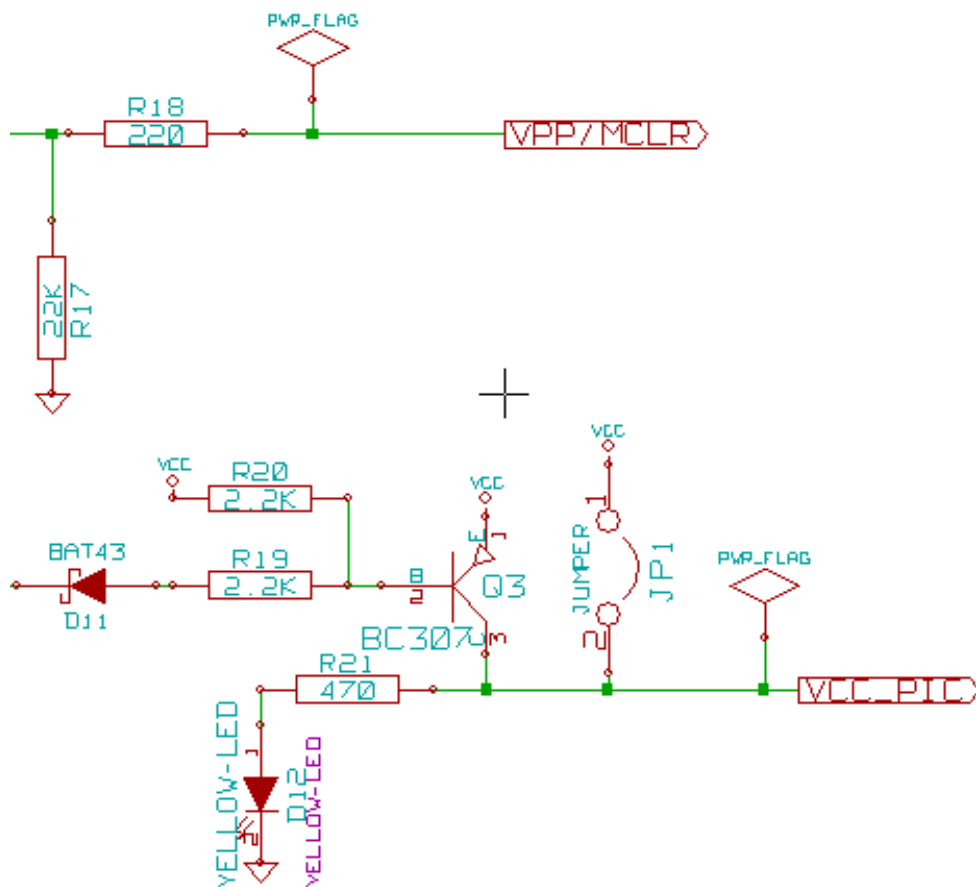
- ルートシートを作成し、他のすべてのシートをそれに含めます。ルートシートはシート間のリンクとして機能します。
- 明示的な接続はまったく必要ありません。
- 全てのシート間の接続には、階層ラベルではなくグローバルラベルを使用します。

ルートシートの例を以下に示します。

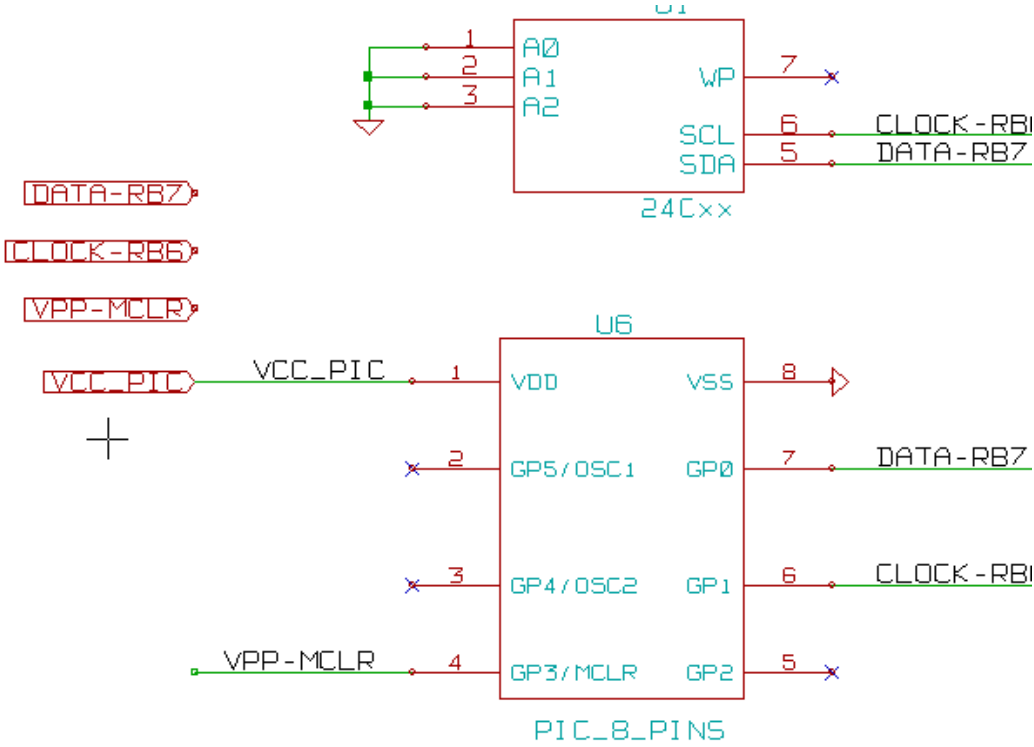


2 ページあり、それらはグローバルラベルで接続されています。

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。

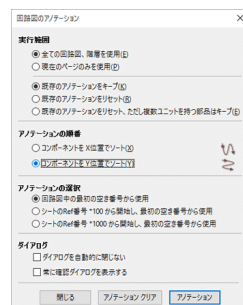
DATA-RB7
CLOCK-RB6
VPP-MCLR

Chapter 8

シンボルアノテーションツール

8.1 はじめに

アノテーションツールを使うことで、回路図中のシンボル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シンボルについては、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“回路図をアノテーション”のメインウィンドウを示します。



アノテーション方法一覧:

- 全てのシンボルをアノテート（「既存のアノテーションをリセット®」を選択）。
- 既存の複数ユニットをシンボルの順番を入れ替えないで、全てのシンボルをアノテート（「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）。
- アノテートされていないシンボルだけアノテート。“?”で終わるリファレンス（参照番号）を持つシンボルが対象です。
- 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シンボル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。

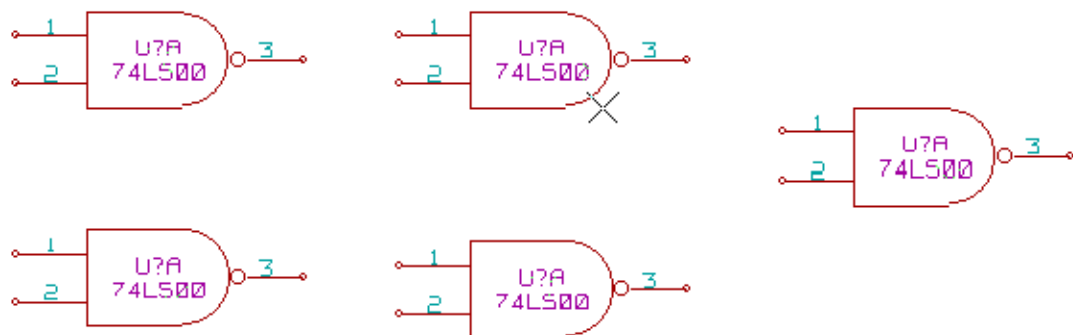
「アノテーションの選択」オプションでは、リファレンス（参照番号）の計算方法を指定します。：

- 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（各リファレンス接頭辞につき）1からリファレンス番号が振られます。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 からリファレンス番号が振られます。それぞれのリファレンス接頭辞（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番号が振られます。
- 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。

8.2 例

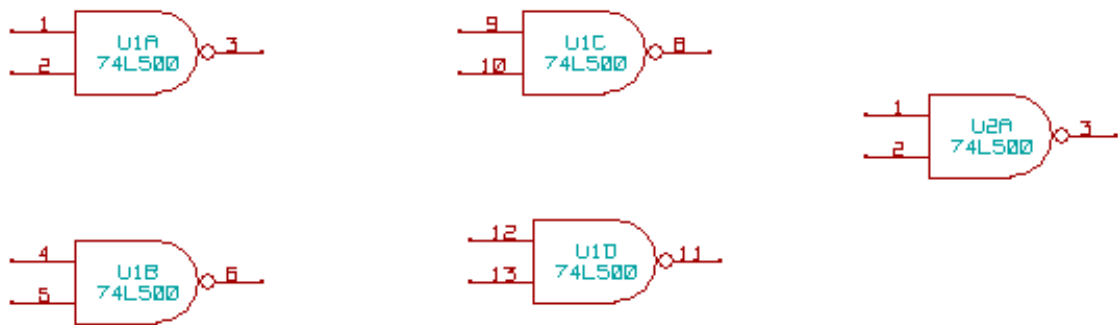
8.2.1 アノテーションの順序

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。

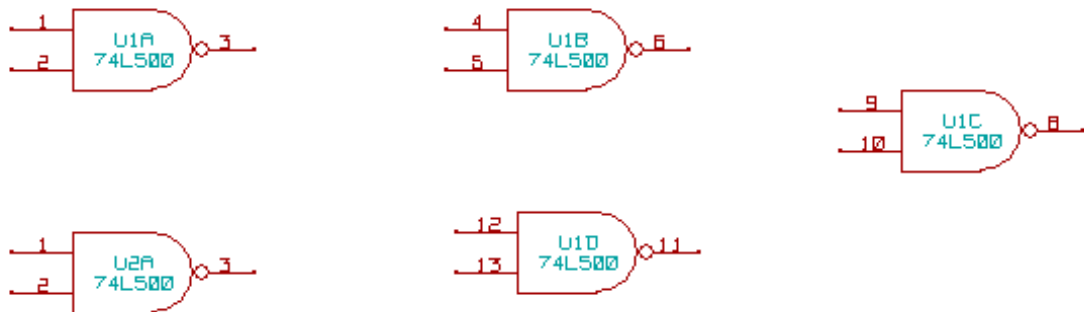


アノテーションを実行すると、以下のような結果が得られます。

コンポーネントをX位置でソートした場合：



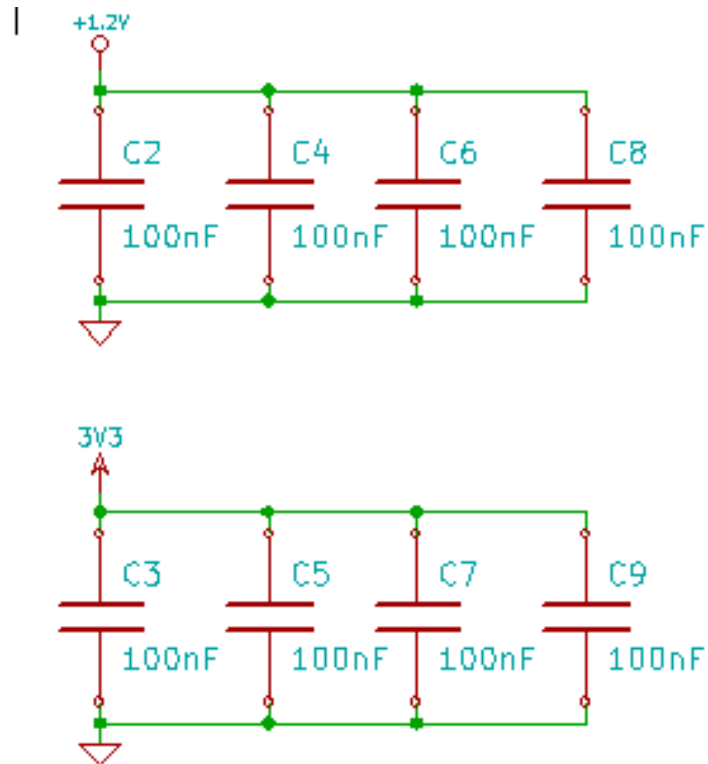
コンポーネントを Y 位置でソートした場合。



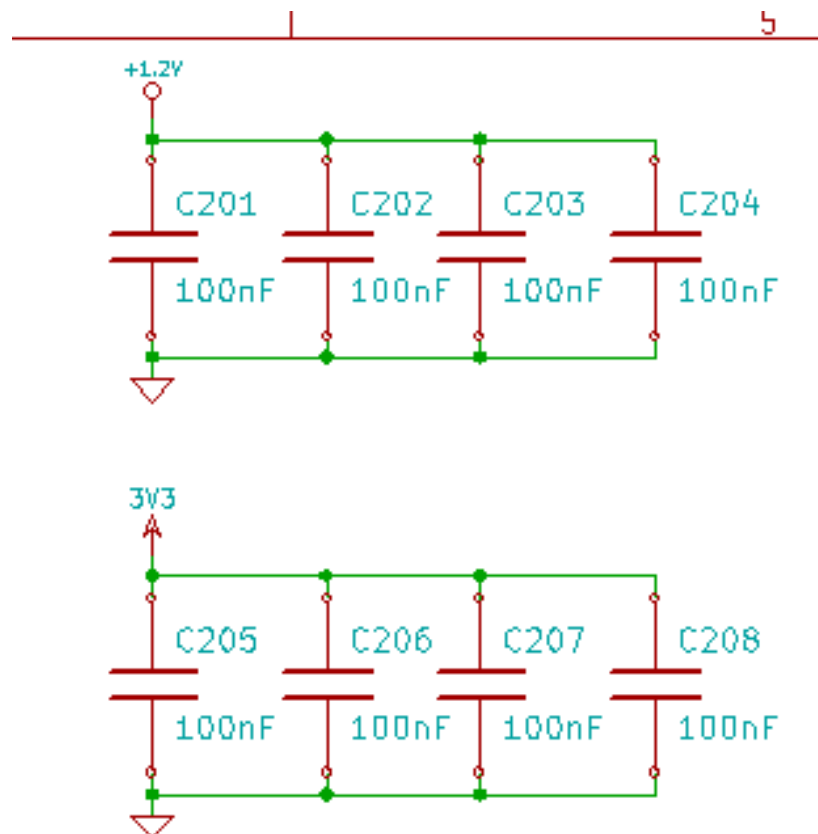
アノテーションにより、74LS00 の 4 つのゲートが U1 パッケージにまとめられ、5 番目のゲートは次の U2 へと分類されました。

8.2.2 アノテーションの選択

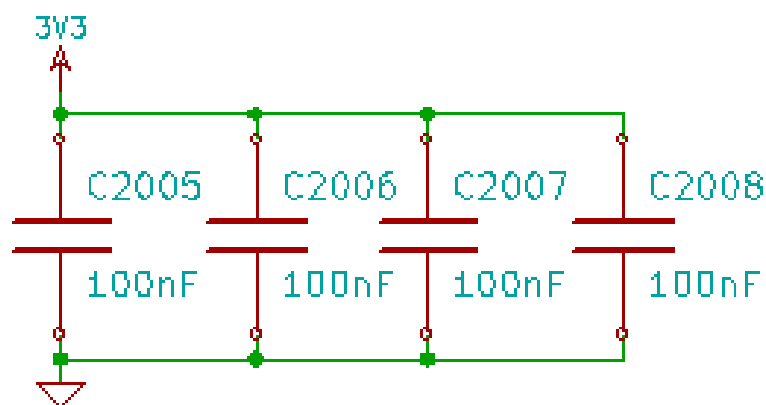
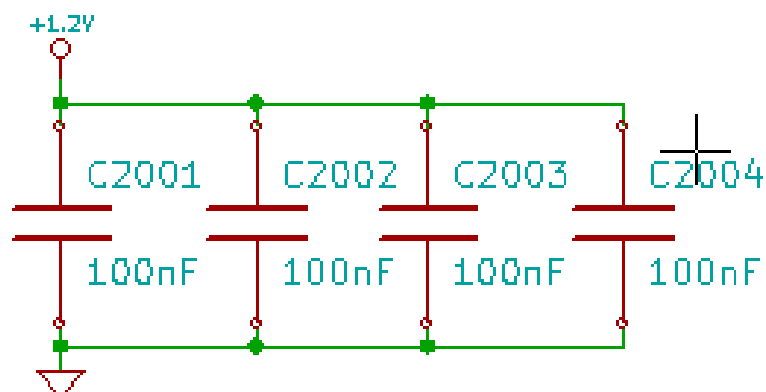
下記の図は、部品をシート 2 に配置し、「回路図中の最初の空き番号から使用する」オプションを利用してアノテーションを行ったものです。



「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。



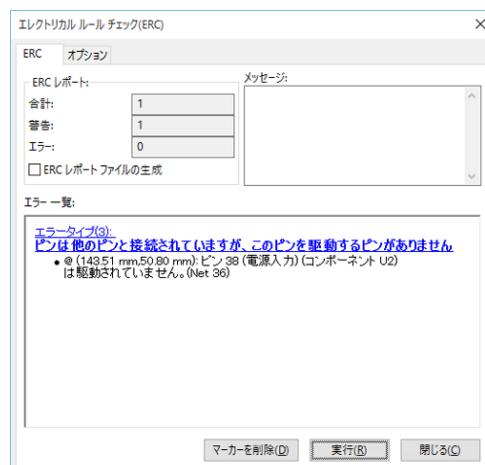
Chapter 9

ERC (エレクトリカル・ルール・チェック) による設計検証


9.1 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、設計エラーを検出するソフトウェアは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するので非常に便利です。

実際には、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になるよう修正する必要があります。ERC の質は、シンボルライブラリー作成中にピンの電氣的なプロパティをどれだけ細かく指定したかに依ります。ERC の出力は“エラー”または“警告”として報告されます。



9.2 ERC の使用法

アイコン  をクリックすると ERC を開始します。

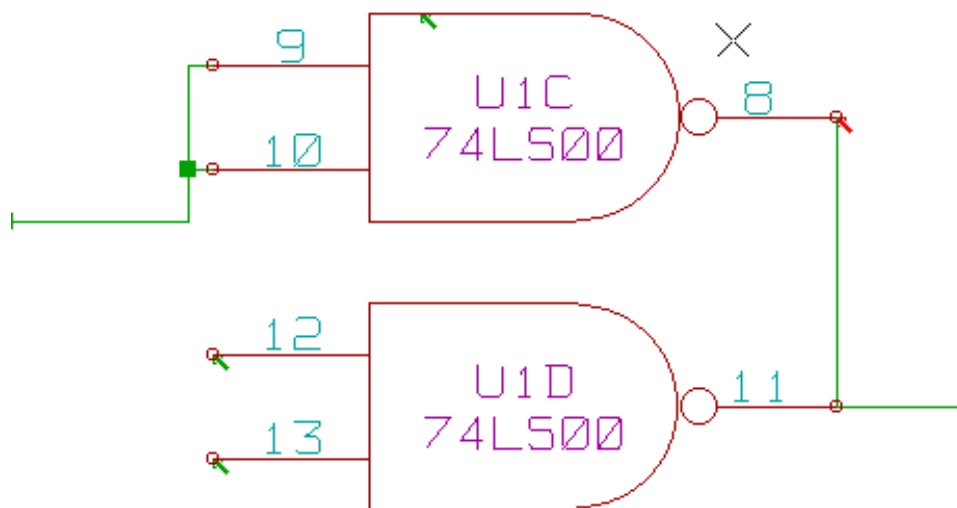
マーカーは、ERC エラーを出力した回路図の要素（ピンまたはラベル）上に配置されます。

注意

- “電気的・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）

9.3 ERC の例



エラーが4つ見られます：

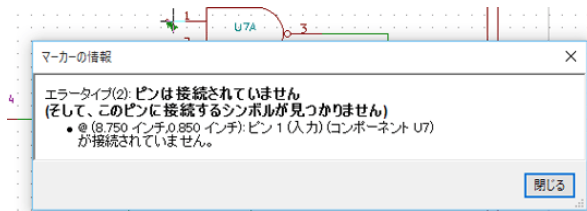
- 2本の出力が誤接続されています（赤の矢印）。
- 入力が2本未接続のままです（緑の矢印）。
- 非表示電源ポートのエラーで、電源フラグがありません（上部に緑の矢印）。

9.4 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。

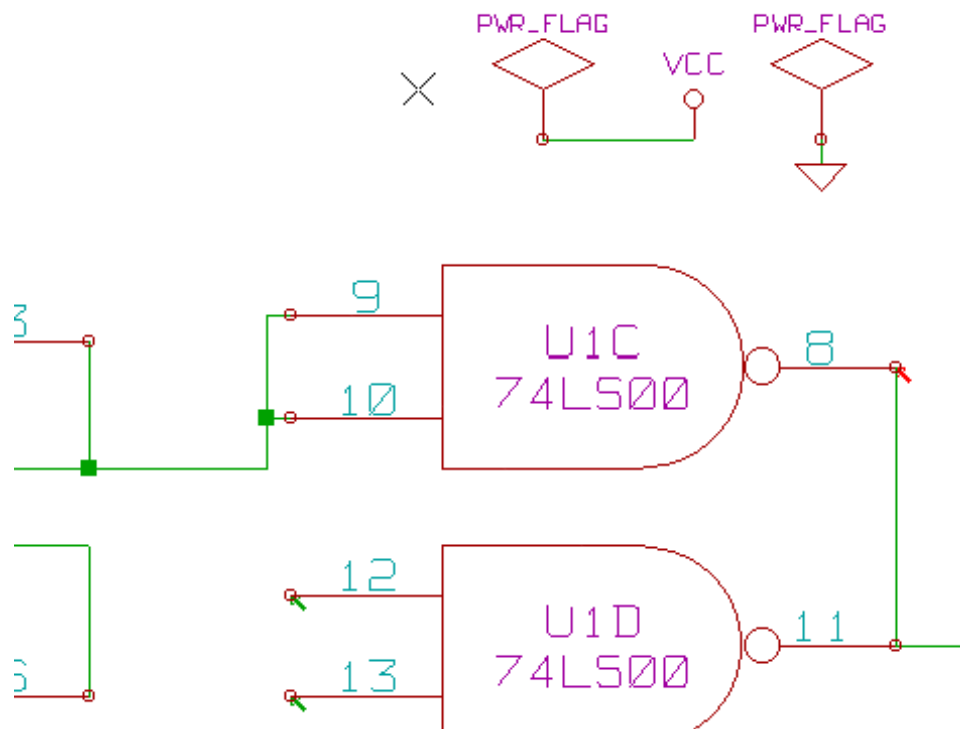


9.5 電源ピンと電源フラグ

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。:

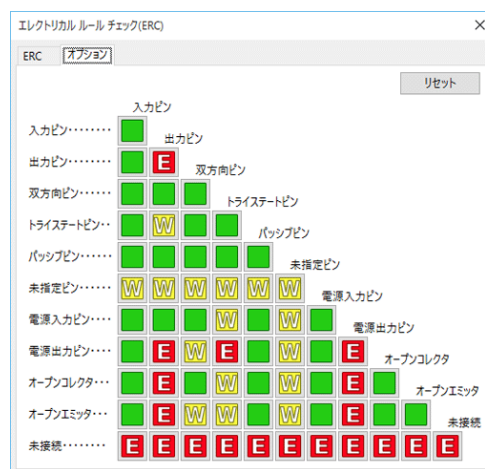


このようにすると、エラーマーカーが消えます。

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。なぜなら、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い（通常の属性は電源入力）ため、PWR_FLAG に接続しなければ電源に接続されたことにはならないからです。

9.6 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。

9.7 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 10

ネットリストの作成

10.1 概要

ネットリストはシンボル間の電氣的接続を記述したファイルです。これらの接続はネットとして参照されます。ネットリストのファイルには、次のものが含まれます:

- シンボルのリスト
- シンボル間の接続（ネット）一覧


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。シンボルのリストとネットのリストが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。なぜなら、ネットリストは次のような他の電子系 CAD ソフトウェアを紐付けるからです。:

- PCB レイアウトソフトウェア。(基板設計ソフト)
- 回路および電氣的な信号シミュレータ。
- CPLD（および他のプログラマブル IC の）コンパイラ。

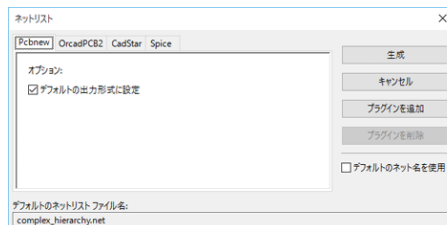
Eeschema はいくつかのネットリストのフォーマットをサポートしています。

- 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)。
-

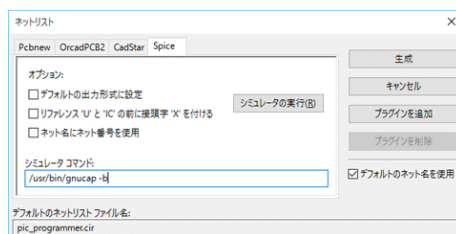
10.2 ネットリストフォーマット

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



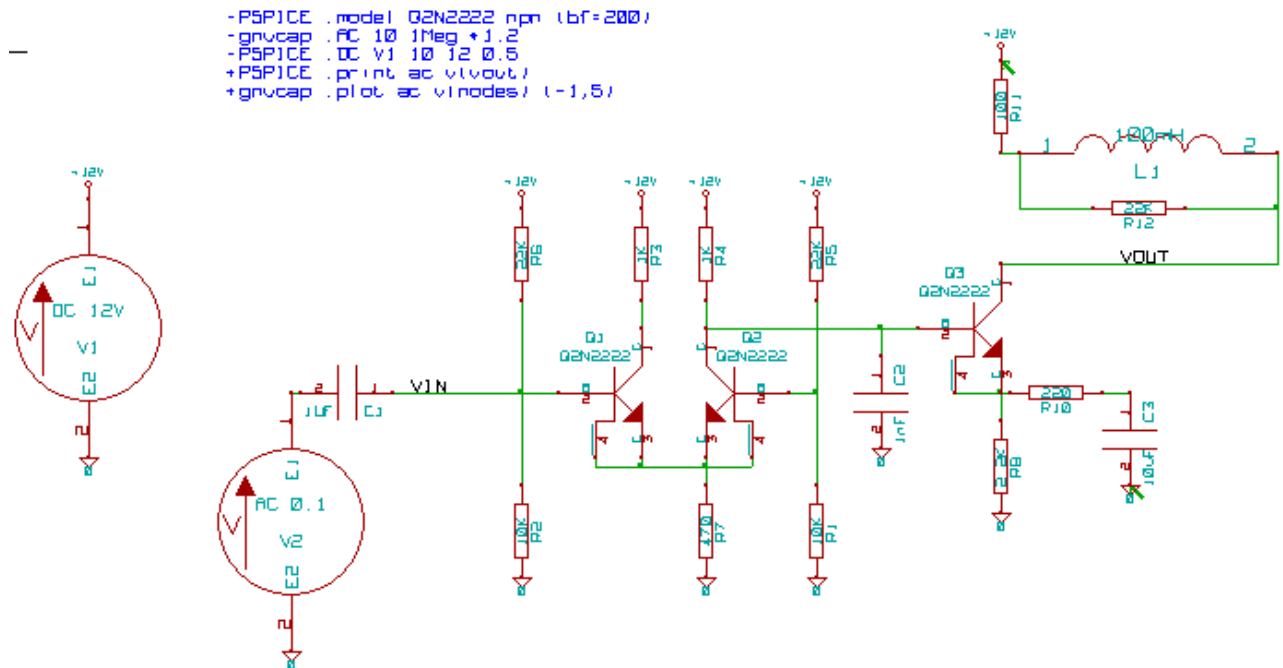
それぞれのタブで希望するフォーマットを選択できます。Spice フォーマットでは、可読性を上げるためにネットの名前を付けたり、古い Spice で使われるネット番号のどちらかを付けて、ネットリストを生成することが可能です。ネットリストボタンをクリックすると、ネットリストファイルの名前を入力するよう促されます。

注意

大きな回路図のネットリストの生成には、数分かかることがあります。

10.3 ネットリストの例

PSpice ライブラリー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 genereee 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
```

10.4 ネットリストについての注釈

10.4.1 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、ネットあるいは他の情報に半角スペースを利用できません。互換性を保つために、ラベル、コンポーネントの名称や値のフィールド、ピン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。

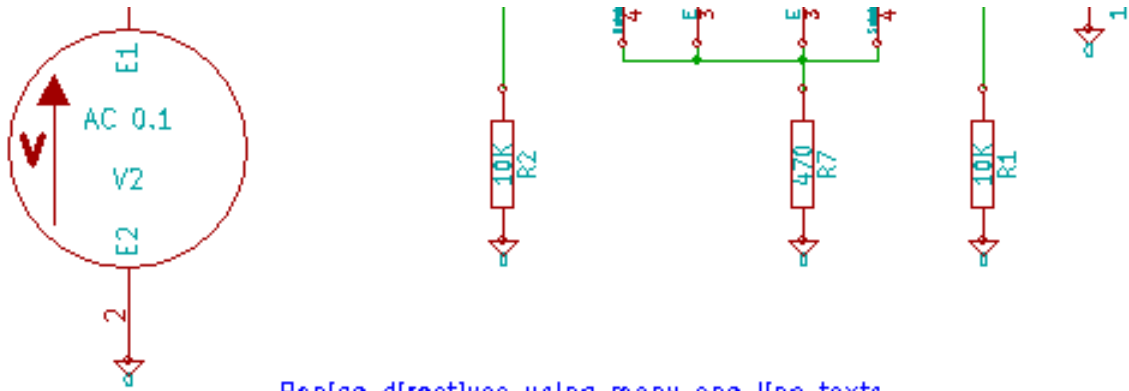
10.4.2 PSPICE ネットリスト

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 **-gnuicap** 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 **+gnuicap** 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:



Pspice directives using many one line texts

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

Pspice directives using one multiline text:

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

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、GND のネットは 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。

10.5 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。

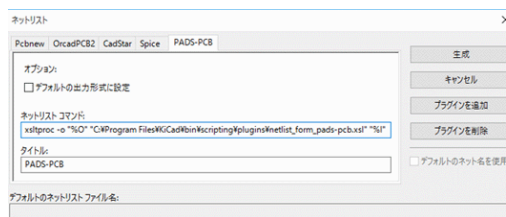
コンバーターはテキストファイル（xsl フォーマット）ですが、Python のような他の言語を使用することもできます。xsl フォーマットを使用する場合、ツール（xsltproc.exe あるいは xsltproc）は Eeschema が生成した中間ファイルとコンバーターファイルを読み込んで、出力ファイルを生成します。この場合、コンバーターファイル（シートスタイル）は非常に小さく記述が容易です。

10.5.1 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。

10.5.2 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	xsl ファイルを読み込み、変換するツール
-o %O.net	出力ファイル: %O で出力ファイルを定義
f:/kicad/bin/plugins/netlist_form_pads-pcb.xsl	ファイル名コンバーター (シートスタイル, xsl フォーマット)
%I	Eeschema が生成した中間ファイル (*.tmp) で置き換える

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.
```

10.5.3 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。

10.5.4 中間ネットリスト・ファイル・フォーマット

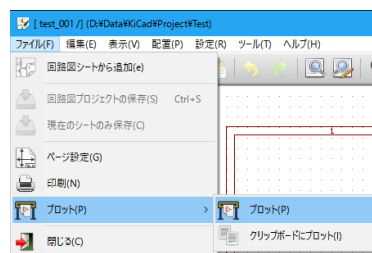
xslproc についてのさらに多くの説明、中間ファイルフォーマットの記述内容、各コンバーターの場合のシートスタイルの例は 14 章を参照して下さい。

Chapter 11

プロットと印刷

11.1 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。

11.2 プロットの共通コマンド

現在のページをプロット

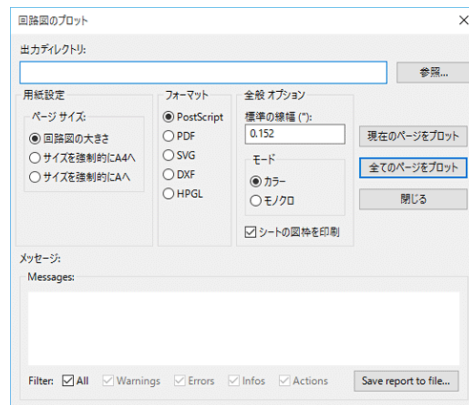
現在のシートのみプロットします。

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）。

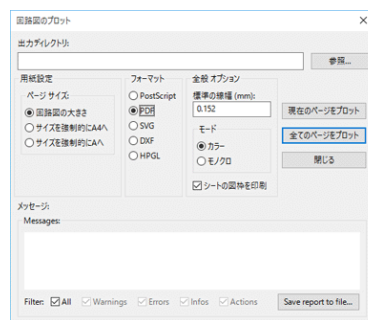
11.3 Postscript のプロット

“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。



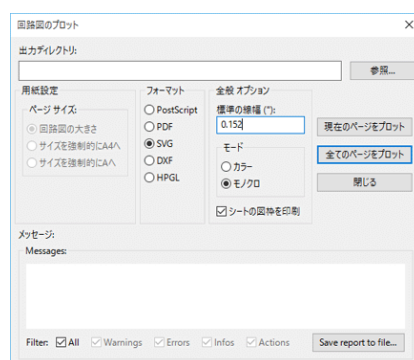
ファイル名はシート名に拡張子.ps を付加したものになります。“シート図枠を印刷” オプションを無効にすることが可能です。これは文書編集ソフトウェアで図を挿入するためにしばしば使用されるカプセル化ポストスクリプトファイル (.eps フォーマット) を生成する場合に便利です。メッセージウィンドウは生成されたファイル名を表示します。

11.4 PDF のプロット



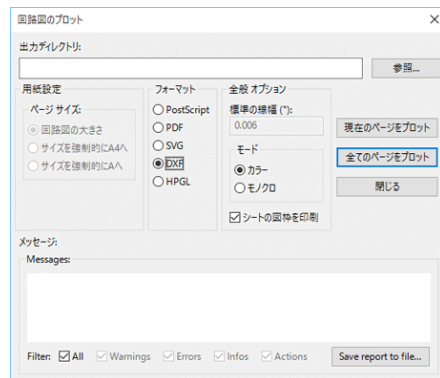
“回路図のプロット” 内のフォーマットで PDF オプションを指定します。ファイル名はシート名に拡張子.pdf を付加したものになります。

11.5 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。

11.6 DXF のプロット



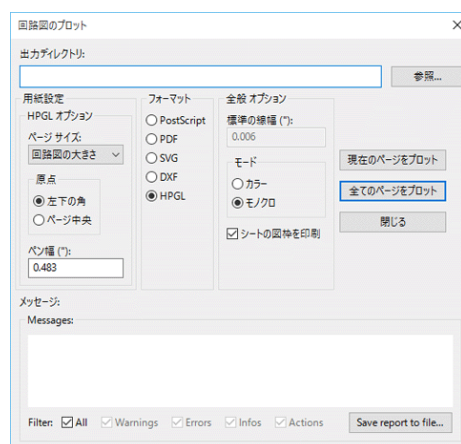
“回路図のプロット” 内のフォーマットで DXF オプションを指定します。ファイル名はシート名に拡張子.dxf を付加したものになります。

11.7 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。

11.7.1 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。

11.7.2 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。


一般的には：

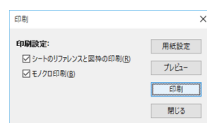
- シートの中央を原点とするプロッターの場合、オフセットは負の値で、シート寸法の $1/2$ に設定しなければなりません。
- シートの左下角を原点とするプロッターの場合、オフセットは 0 に設定しなければなりません。

オフセットを設定するには：

- 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）

11.8 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からです。

Chapter 12

シンボル・ライブラリー・エディター

12.1 シンボル・ライブラリーに関する一般情報

シンボルは、グラフィカル表現、電氣的な接続、シンボルを定義するフィールドを含んだ回路図の構成要素です。回路図で使われるシンボルは、シンボル・ライブラリーに保管されます。Eeschema は、ライブラリーの作成、ライブラリーへのシンボルの追加と削除、シンボルのライブラリー間での移動、ファイルへのシンボルのエクスポート、ファイルからのシンボルのインポートができるように、シンボル・ライブラリー編集ツールを備えています。ライブラリー編集ツールはシンボル・ライブラリー・ファイルを管理するためのシンプルな方法を提供します。

12.2 シンボル・ライブラリーの概要

シンボル・ライブラリーは 1 つ以上のシンボルから構成されています。基本的にシンボルは、機能、種類、あるいは製造元によって論理的にグループ化されます。

シンボルは次の要素で構成されています:

- グラフィカル要素 (線、円、円弧、テキストなど): 記号の定義を規定する
- ピン: 図形のプロパティ (線、クロック、反転、Low レベル・アクティブなど) とエレクトリカル・ルール・チェック (ERC) ツールが使用する電氣的プロパティ (入力、出力、双方向など) の両方を持つ
- フィールド: リファレンス、値、基板 (PCB) 設計のための対応するフットプリント名など
- エイリアスは、7400 のような通常のシンボルと共に、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ー・シンボルを共有します。

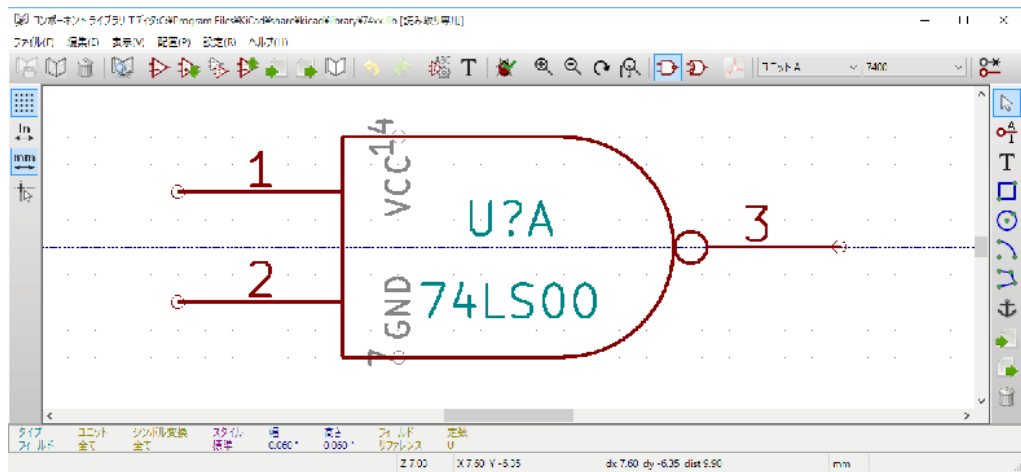
適切なシンボルの設計には以下のことが要求されます:

- 一つ以上のユニットで構成されるようにシンボルを定義する。
 - ド・モルガン表現として知られる代替ボディ・スタイルをシンボルが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、電氣的プロパティ（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。
- 他のシンボル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。あるいは他のシンボルから新たにシンボルを作成した場合、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。
- 説明文やデータシートへのリンクなどを付加してシンボルに記録する。
- 適切な (desired) ライブラリーにそれを保存する。

12.3 シンボル・ライブラリー・エディターの概要




シンボル・ライブラリー・エディター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つのツールバーとシンボルの閲覧／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。





















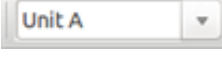





12.3.1 メイン・ツールバー

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、ライブラリー管理ツール、取り消し／やり直しコマンド、ズーム・コマンドとシンボル・プロパティ・ダイアログから成ります。














	選択されているライブラリーをハードディスクに保存する。(ライブラリーが選択されていないか、選択中のライブラリーに変更がない場合、このボタンは無効です。)
	編集するライブラリーを選択する。
	選択されているライブラリー (ライブラリーが選択されていない場合にはプロジェクトで定義されたライブラリー) からシンボルを削除する。

	シンボル・ライブラリー・ブラウザを開く。(編集するシンボルとライブラリーの選択)
	新しいシンボルを作成する。
	選択されているライブラリーから編集用にシンボルを読み込む。
	現在ロードされているシンボルから新しいシンボルを作成する。
	現在のシンボルに対する変更をメモリーに保存する。ハード・ディスク上のライブラリー・ファイルは変更されません。
	ファイルからシンボルを 1 つインポートする。
	シンボルをファイルへエクスポートする。
	現在のシンボルを含んだ新規ライブラリー・ファイルを作成する。注: 新しいライブラリーは自動的にプロジェクトに追加されません。
	最後の変更を元に戻す。
	最後の変更をやり直す。
	シンボルのプロパティを編集する。
	シンボルのフィールドを編集する。
	シンボルの設計エラーをテストする。
	ズームイン。
	ズームアウト。
	画面の再描画。
	シンボルの大きさを画面に合わせる。
	通常のボディ・スタイルを選択する。(シンボルが代替ボディ・スタイルを持っていない場合、このボタンは無効です)
	別のボディスタイルを選択する。(シンボルが代替ボディ・スタイルを持っていない場合、このボタンは無効です)
	関連ドキュメントを表示する。(シンボルに関連ドキュメントが定義されていない場合、このボタンは無効です)
	表示するユニットを選択する。(シンボルが複数のユニットを派生させていない場合、ドロップ・ダウンは無効です)

	エイリアスを選択する。(シンボルがエイリアスを持っていない場合、ドロップ・ダウンは無効です)
	ピンを編集する: 複数ユニット・コンポーネントで代替シンボルを持つ場合、ピンの形状と位置を個別に編集
	ピンテーブルを表示する。


12.3.2 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、シンボルの設計で必要とされる全ての要素を画面に配置できます。ツールバーの各ボタンの定義は下表のとおりです。

	選択ツール。(右クリックでカーソルの下にコンテキストメニューが開きます。左クリックでカーソルのあるオブジェクトの属性がメインウィンドウの下にあるメッセージパネルへ表示されます。左ダブルクリックでカーソルのあるオブジェクトのプロパティダイアログが開きます。)
	ピンツール。(左クリックで新しいピンを追加します)
	グラフィックテキストツール。(左クリックで新しいグラフィックテキストを追加します)
	矩形ツール。(左クリックで矩形の最初のコーナーから描画し始めます。次の左クリックで矩形の対頂点のコーナーが置かれます。)
	円ツール。左クリックで中心から円の描画を始めます。次の左クリックで円の半径が決まります。
	円弧ツール。(左クリックで中心から円弧の描画を始めます。次の左クリックで円弧の最初の点が決まります。更に次の左クリックで円弧の二つ目の点(終点)を決めます。
	ポリゴンツール。(左クリックで新しいポリゴンの描画を始めます。左クリックする毎にポリゴンの辺が追加されます。左ダブルクリックでポリゴンの描画を終了します。
	アンカーツール。(左クリックでシンボルのアンカー位置がセットされます。)
	ファイルからシンボルを 1 つインポートする。
	シンボルをファイルへエクスポートする。
	削除ツール。(左クリックで現在のシンボルからオブジェクトを削除します)


12.3.3 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、エディター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。



	グリッド表示の切替。(グリッドの表示/非表示が交互に切り替わります)
---	------------------------------------

	単位をインチに設定。
	単位を mm に設定。
	フル・スクリーン・カーソルの切替。(カーソルが通常/フル・スクリーンと交互に切り替わります)

12.4 ライブラリーの選択および保守

メイン・ツールバーのアイコン  を使って使用可能な全てのライブラリーから一つ選び、現在のライブラリーを選択します。シンボルの読み込みと保存は、このライブラリー (現在のライブラリー) に対して行われます。シンボルのライブラリー名は、自身の定数フィールドの値となります。


注意

- シンボルの読み込みと保存を行うには、Eeschema にライブラリーをロードしなければなりません。
- 現在のライブラリー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。
- シンボルを現在のライブラリーから削除するには  をクリックします。

12.4.1 シンボルの選択と保存

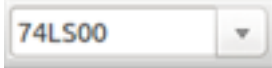
シンボルの編集時には、実際にハード・ディスクのライブラリー内のコンポーネントに対して作業しているのではなく、ローカルのメモリー内にあるそのコピーに対して作業をしています。従って、どのような編集作業も容易に元に戻すことが可能です。また、シンボルはローカル・ライブラリーや既存のシンボルから読み込むことができます。

12.4.1.1 シンボルの選択



メイン・ツールバーのアイコン  をクリックすると、現在選択されているライブラリーから選択して読み込むことができる利用可能なシンボルのリストが表示されます。

注意

シンボル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シンボルの名前が常にウィンドウのタイトルに表示されます。シンボルのエイリアスのリストは常に各シンボルと共に読み込まれ、編集


することができます。  から現在のシンボルのエイリアスを選択することで、新しいシンボルを作ることができます。エイリアス・リストの最初の項目はシンボルのルート名です。


注意


もう一つの方法として、エクスポート・コマンド  で前回保存したシンボルを、インポート・コマンド  により読み込むことができます。

12.4.1.2 シンボルの保存

変更後、シンボルを現在のライブラリーまたは新規ライブラリーへ保存、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。

変更されたシンボルを現在のライブラリーに保存するには、更新コマンド  をクリックします。更新コマンドはローカル・メモリー内にシンボルの変更を保存するだけという点に注意して下さい。このように、実際にライブラリーに保存する前に変更してみるすることができます。


シンボルの変更をライブラリー・ファイルに永続的に保存するには、保存アイコン  をクリックして、ハード・ディスク上の既存のライブラリー・ファイルを変更されたシンボルで上書きします。

このシンボルを含んだ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






新しいライブラリーは現在のプロジェクトへは自動的に追加されません。

回路図で使用したい新しいライブラリーはすべて Eeschema の“設定” → “シンボル・ライブラリーを管理”でダイアログを開いて、シンボル・ライブラリー・ファイルのリストへ追加しなければなりません。

現在のシンボルだけを含んだファイルを作成するには、 をクリックします。このファイルは一つのシンボルだけを含んだ標準ライブラリー・ファイルとなります。このファイルは、別のライブラリーへシンボルをインポートするのに使うことができます。実際、新規ライブラリー作成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。

12.4.1.3 別のライブラリーへのシンボルの移動

コピー元のライブラリーからコピー先のライブラリーへ簡単にシンボル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：

-  をクリックして、コピー元のライブラリーを選択します。
 -  をクリックして、移動するシンボルを読み込みます。シンボルが編集エリアに表示されます。
 -  をクリックして、コピー先のライブラリーを選択します。
 -  をクリックして、ローカル・メモリーにある新しいライブラリーに現在のシンボルを保存します。
 -  をクリックして、(ハード・ディスク上にある) 現在のローカル・ライブラリー・ファイルにシンボルを保存します。
-

12.4.1.4 シンボル変更の取り消し

シンボルでの作業時、編集されたシンボルはライブラリーにある実際のシンボルの作業用コピーです。これは、シンボルを保存していない限り、全ての変更を無効にして再ロードできることを意味しています。既にローカル・メモリー上のシンボルを更新していてもライブラリー・ファイルに保存していないなら、いつでも中止してもう一度開始することができます。Eeschema は全ての変更を元に戻すでしょう。

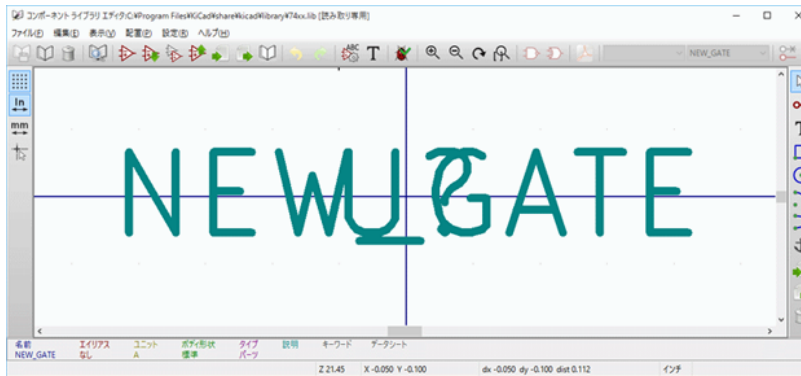
12.5 ライブラリー・シンボルの作成

12.5.1 新規シンボルの作成








をクリックして、新規シンボルを作成することが可能です。シンボル名 (この名前は回路図エディター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 …)、パッケージ内のユニット数 (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つのユニットを持っている)、そして代替ボディ・スタイル (ド・モルガンと呼ばれることもある) が必要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だと、デフォルトの “U” になります。これらのプロパティは後で設定することも可能ですが、シンボルの作成時に設定した方がいいでしょう。

上のプロパティを使って新しくシンボルを作成すると、シンボル・ライブラリー・エディターの画面は次のようになります。




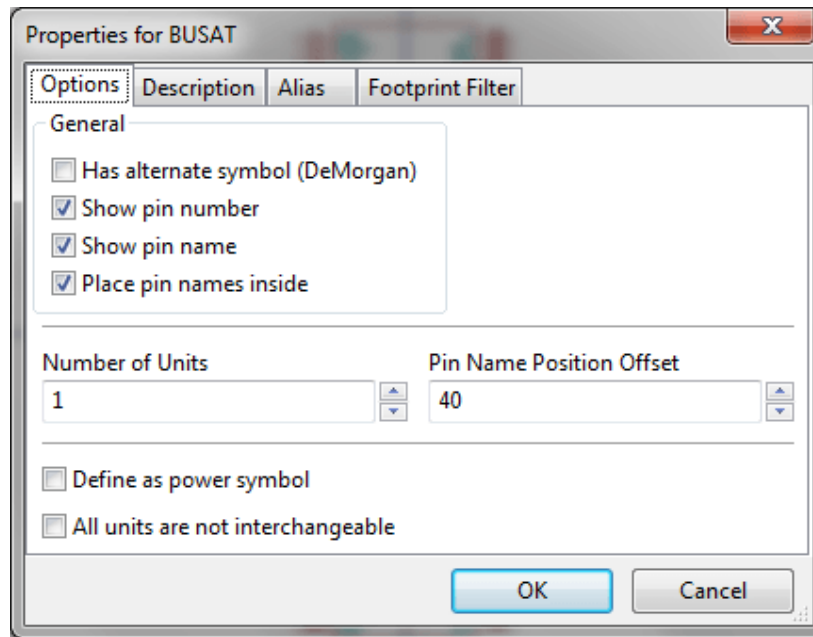
12.5.2 他のシンボルからシンボルを作成

作成したいと思っているシンボルが、シンボル・ライブラリーに既にあるものと似ているということはよくあります。この場合、既存のシンボルを読み込んで変更すると簡単です。

- 出発点として使うシンボルを読み込みます。
-  をクリックするか、または定数フィールド上で右クリックしてテキストを編集し、その名前を変更します。もし変更しようとした名前が既に使われていたなら、新しいシンボル名を求められます。
- 雛型となったシンボルにエイリアスがある場合、新規シンボルから現在のライブラリーで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、新規シンボルの作成は中止されます。シンボル・ライブラリー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。
- 必要に応じて新規シンボルを編集します。
-  をクリックして現在のライブラリーにある新規シンボルを更新するか、または  をクリックして新規ライブラリーに保存します。あるいは他の既存ライブラリーに新規シンボルを保存したい場合には、 をクリックして保存したいライブラリーを選択し、新規シンボルを保存します。
-  をクリックして、ディスクに現在のライブラリー・ファイルを保存します。

12.5.3 シンボル・プロパティ

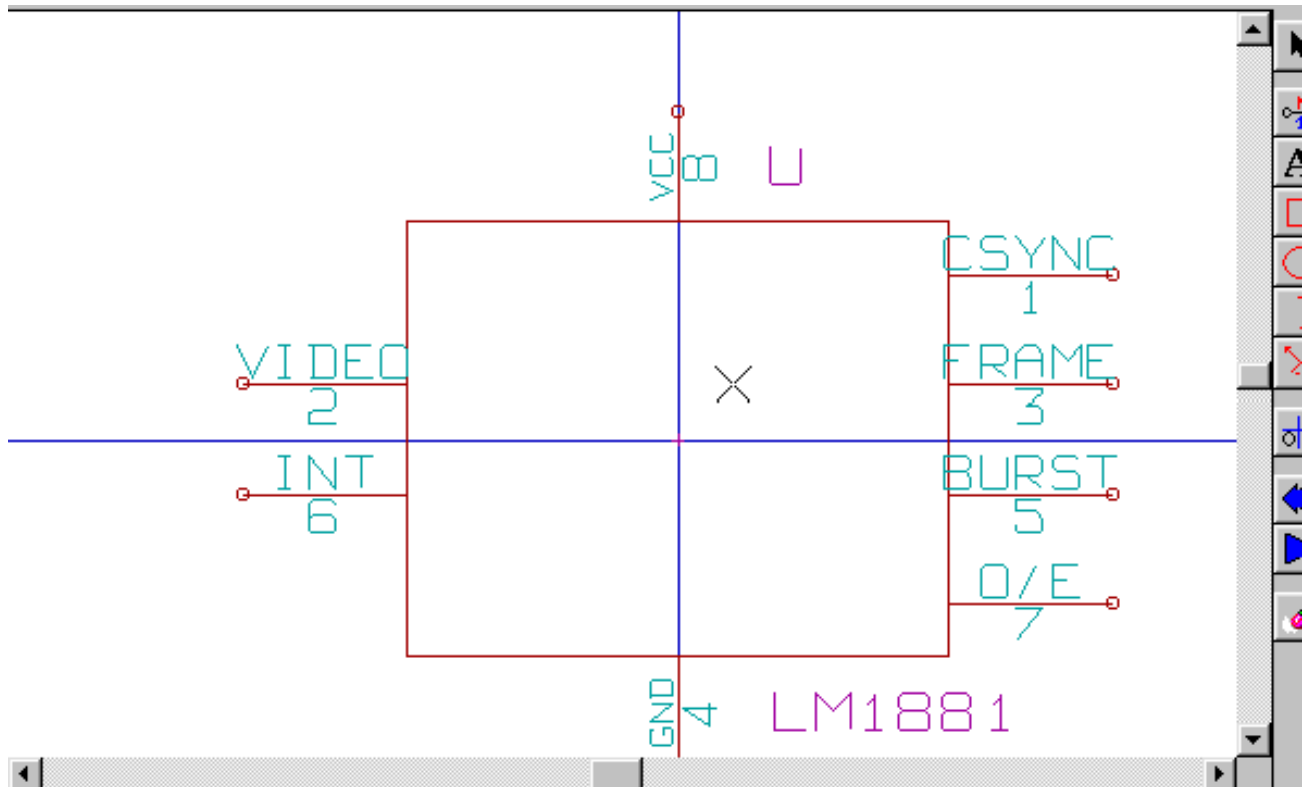
シンボルの新規作成時や既存のシンボルから継承してシンボルを作成する時には、シンボルのプロパティは注意深く設定しなければなりません。シンボルのプロパティを変更するには、 をクリックして、次のダイアログを表示します。




パッケージあたりのユニット数と代替シンボル表現を正しく設定することは大変重要です。もし設定が有効なら、ピンの作成または編集時にピンは各ユニットに正しく反映されるでしょう。ピンの作成／編集後にパッケージあたりのユニット数を増やすと、新しいユニットにピンとシンボルを追加するという余計な仕事をする破目になります。例えそうだとしても、これらのパラメータはいつでも変更可能です。


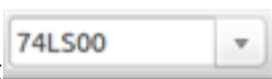
グラフィック・オプション”ピン番号を表示”と”ピン名を表示”は、ピン番号とピン名のテキストの可視性を定義します。対応するオプションがチェックされていたなら、そのテキストは表示されます。オプション”ピン名を内側に配置”は、ピンに対するピン名の位置を定義します。オプションがチェックされている場合、そのテキストはシンボル・アウトラインの内側に表示されます。この場合、”ピン名の位置オフセット”プロパティはピンの外形端から内側方向へのテキストの変位量を定義します。値は (1/1000 インチ単位で) 30 ~ 40 が妥当です。

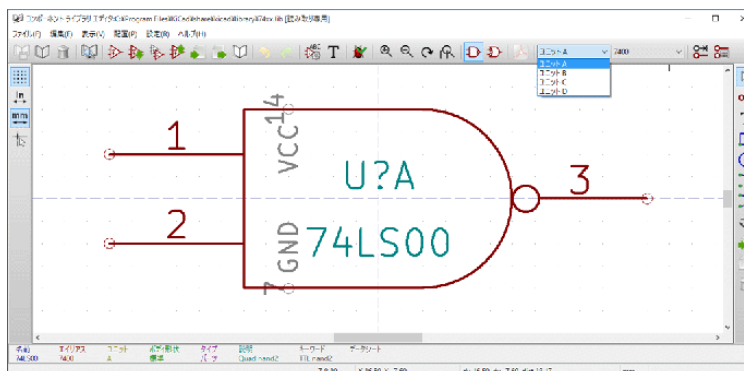
以下の例は、”ピン名を内側に配置”オプションにチェックをつけない状態のシンボルを示しています。ピン名とピン番号の位置に注意して下さい。



12.5.4 代替シンボル表現をもつシンボル

もしシンボル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シンボル表現を一つ選択しなければならないでしょう。通常のシンボル表現を編集するには、 をクリックします。

代替シンボル表現を編集するには、 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。次に画面例を示します。



12.6 グラフィック要素

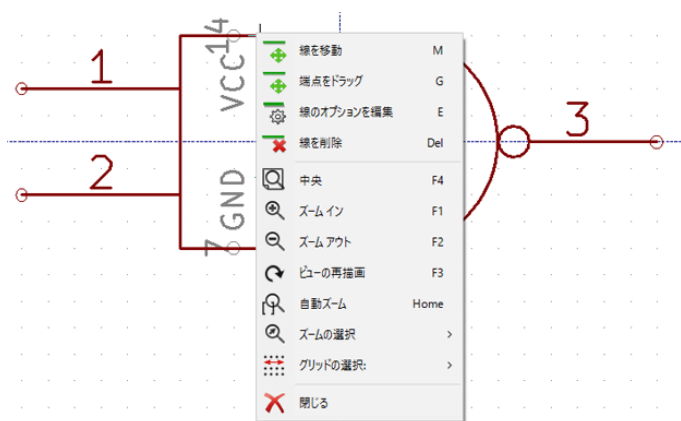
グラフィック要素はシンボル表現を作成し、電気的接続の情報は含みません。この設計には次のツールが使用可能です:

- 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。)

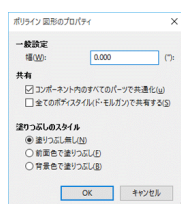
メイン・ウィンドウの右側にある縦ツールバーには、シンボルの表現をデザインするために必要な全てのグラフィック要素を配置できます。

12.6.1 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、全てのユニットで共通/全てのボディスタイルで共通/ユニット毎/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、現在表示しているユニットにおけるライン要素の幅を定義します。
- “シンボルにある全てのユニットで共通”設定は、パッケージに一つ以上のユニットを持つシンボルの各ユニットにグラフィック要素を描く、または現在のユニット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。
- “全てのボディ・スタイル (ド・モルガン) で共通”設定は、代替ボディ・スタイルを持つシンボルの各ボディ・スタイルへ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。
- 塗りつぶしスタイル設定は、グラフィック要素で定義されたシンボルが、塗りつぶし無し、背面色で塗りつぶし、前面色で塗りつぶし、の何れで描画されるかを決定します。

12.6.2 グラフィック・テキスト要素

Tにより、グラフィック・テキストを作成がすることができます。グラフィック・テキストは、シンボルが反転 (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・テキストはフィールドではないことに注意して下さい。

12.7 複数ユニット・シンボルと代替ボディ・スタイル

シンボルは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つ以上のユニットを持っています。いくつかのシンボルは、異なったシンボル表現と異なるピン定義を持つユニットを1つのパッケージ内に複数持っています。

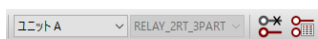
例として、3つの異なるユニット (コイル、スイッチ 1、スイッチ 2) を持つシンボルとして設計することが可能な2つのスイッチを持つリレーを考えてみましょう。複数ユニット・シンボルと代替ボディ・スタイルを持つシンボルの設計はとても柔軟に行えます。ピンまたはボディ・スタイルは、それぞれのユニットに共通または特定のユニットのみ、もしくは両方のシンボル表現に共通または特定のシンボル表現のみ、に適用できます。

デフォルトでは、ピンはそれぞれのユニットのそれぞれのシンボル表現に固有です。それは、ピンの数は各ユニットで異なり、シンボル表現に依存した形状となるからです。ピンが各ユニットまたは各シンボル表現で共通の場合には、全てのユニットとシンボル表現に対して一度作成するだけで済みます (電源ピンの場合は一般的)。これは、それぞれのユニット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)。

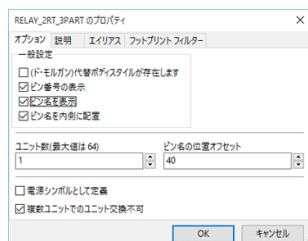
12.7.1 異なったシンボル表現で複数ユニットを持つシンボルの例:

これはパッケージ毎に3つのユニット (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。:

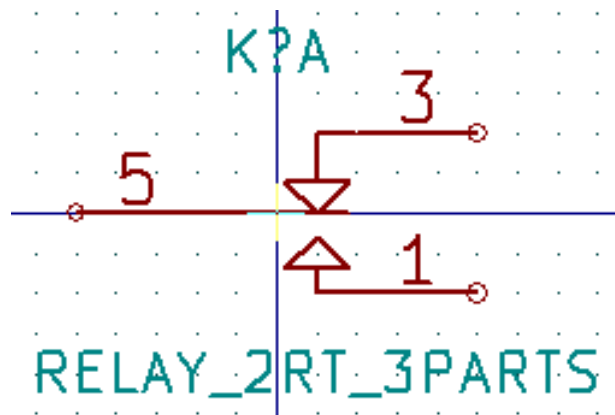
オプション: ピンはリンクしていない。別のユニットのいかなるピンとも同期せず、各ユニットごとに単独でピンを追加/編集できる。



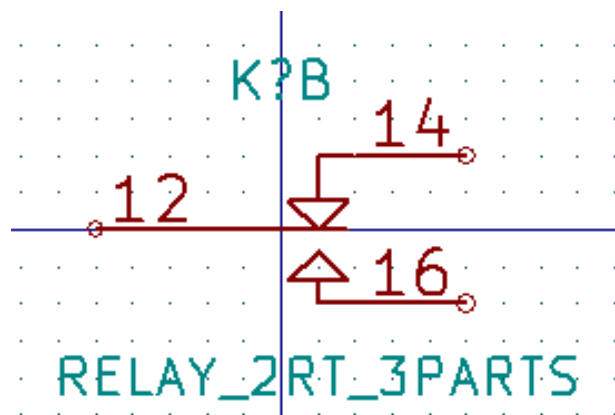
選択されている全てのユニットは入れ替えできない。



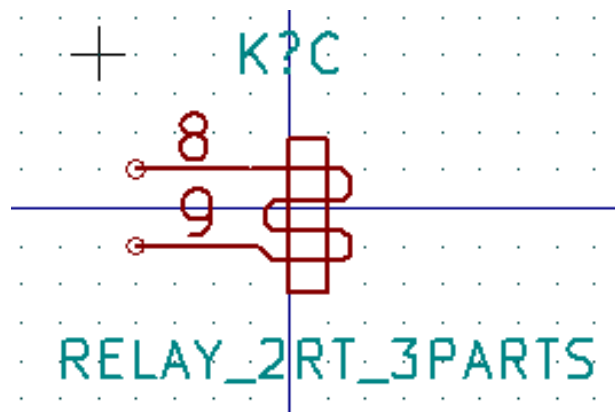
ユニット 1



ユニット 2



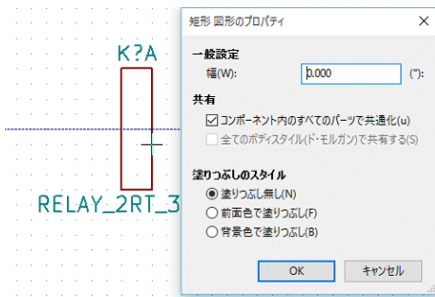
ユニット 3




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。

12.7.1.1 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つのユニットは異なったシンボル表現を持っています。そのため、各ユニットは個別に作成され、グラフィック・シンボル要素は”シンボルの全てのユニットで共通”を無効としなければなりません。



12.8 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。

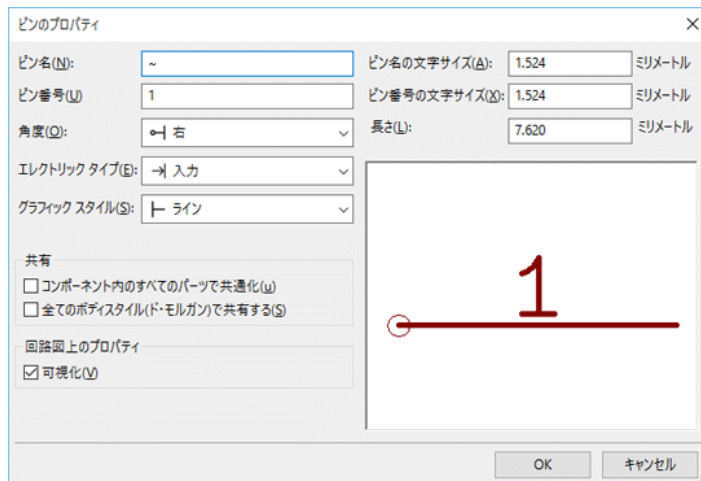
12.8.1 ピンの概要

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、“電氣的”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 “ ”) 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~0 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 など有効です。
- シンボルで重複したピン “番号” を使うことはできません。

12.8.2 ピンのプロパティ

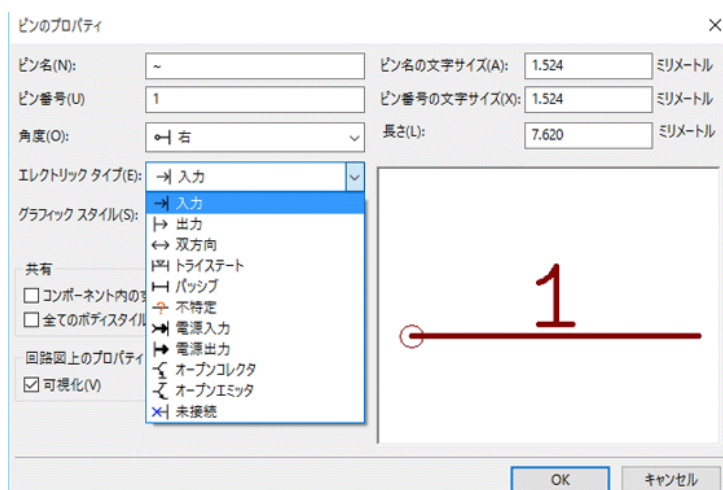


ピンプロパティダイアログでピンの全ての特性を編集することができます。このダイアログは、ピンを作成したり、あるいは既存のピンをダブルクリックすると、自動的にポップアップします。このメニューで以下の定義または変更が可能です。:

- “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ユニットで共通” および “全てのボディスタイル (ド・モルガン) で共有する (S)”
- “可視性 (V)”。(電源ピンで使用される。)

12.8.3 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、電氣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。)



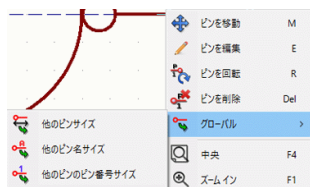
12.8.4 ピンの電気的タイプ

電気的タイプの選択を正しく行うことは ERC ツールにとって重要です。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シンボル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。シンボル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。シンボル内部で接続されていないピンに使用されます。


12.8.5 ピンのグローバル・プロパティ

すべてのピンの長さ、および名前とパーツ番号のテキスト・サイズは変更可能です。右クリックでポップアップされるピンのコンテキスト・メニューにあるグローバル・コマンドを使用して設定します。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。その値は現在のシンボル表現のピンすべてに適用されます。



12.8.6 複数ユニットおよび代替シンボル表現におけるピン定義

複数ユニットおよび代替シンボル表現を持つシンボルは、ピンの作成、編集を行う場合、特に問題となります。ピンの多くは、個別のユニット (ピン番号はユニットごとに指定される) 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。ピンの作成と編集は、パッケージ毎に複数ユニットを持っていたり代替シンボル表現があるシンボルに対して、問題となり得ます。シンボル・ライブラリー・エディター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、複数ユニット・シンボルの全てのユニットと代替シンボル表現を持つシンボルの両方の表現に対して行われます。


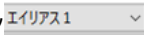
これに対して、ピンの形状 (グラフィック・スタイル) と名前は例外です。この依存関係は多くの場合、ピンの作成、編集を容易にします。また、この依存関係はメイン・ツールバーの  で無効にできます (トグル動作)。これにより完全に独立して、各ユニットと個別のシンボル表現に対してピンを作成することができます。

シンボルは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ユニットから構成することができます。あるシンボルでは、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。[前のセクション](#)にあるリレーのサンプルのように、リレーは3つの別のユニット(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。

複数ユニット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、複数ユニットで共通または各ユニット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、ピンは各ユニットのそれぞれの表現に固有です。それは、ピンの数は各ユニットで異なり、それら(ユニット)のデザインは各シンボル表現で異なるからです。全てのユニット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、シンボルの定義では8個の出力ピンを別々に定義します。新しい 7400 シンボルが作られると、通常シンボル表現のユニット A がライブラリー・エディターに表示されるでしょう。代替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。各ユニットのピン番号を編集するには、ドロップダウン **イリアス1** を使ってユニットを選択します。

12.9 シンボルのフィールド

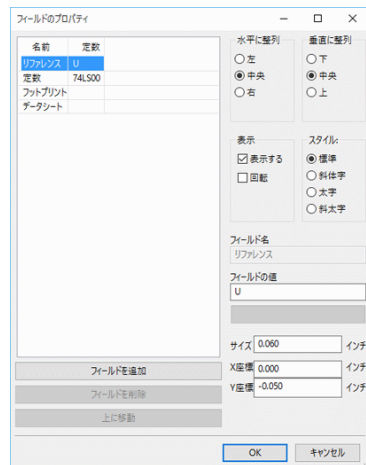
全てのライブラリー・シンボルには4つのデフォルト・フィールドが定義されています。リファレンス、定数、フットプリント、データシートのフィールドは、シンボルが作られるか、コピーされると必ず生成されます。必須項目は、リファレンス、定数のフィールドだけです。既存のフィールドに対しては、ピンを右クリックすることでコンテキスト・メニューのコマンドを使うことができます。ライブラリーにあるシンボルには、通常これら4つのデフォルト・フィールドが定義されています。製造元、品番、価格などのような追加フィールドをライブラリー・シンボルに追加できますが、一般的にこれは、追加フィールドを回路図にある全てのシンボルに適用できるように回路図エディターで行われます。

12.9.1 シンボル・フィールドの編集

既存シンボル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下のフィールド・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、以下に示すフィールドのプロパティダイアログを開きます。



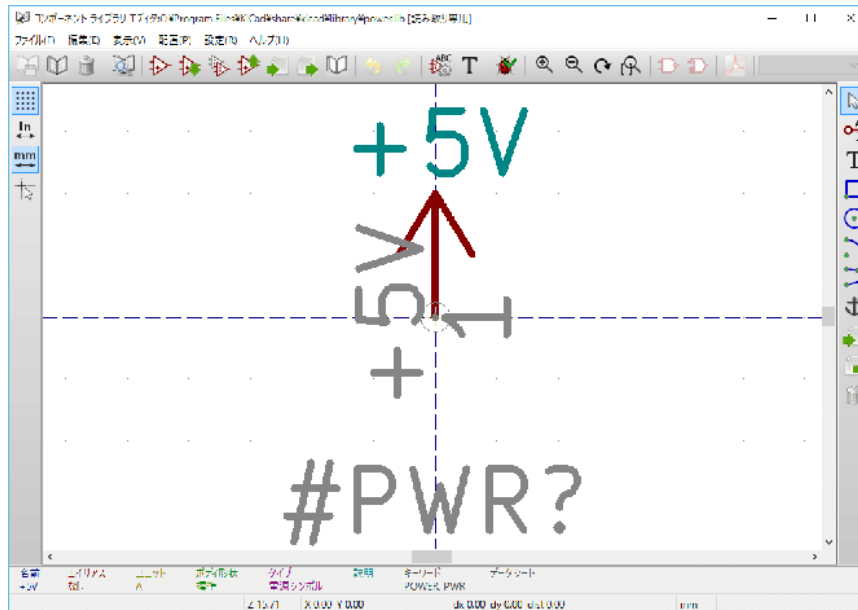
フィールドはシンボルに関連付けされたテキスト・セクションです。シンボルのグラフィック表現に属するテキストと混同すべきではありません。

重要な注意

- 定数フィールドのテキストを変更することは、雛型として現在のシンボルを使って新規シンボルを作成することに相当します。選択されているライブラリーに保存する場合、この新規シンボルは定数フィールドに入っている名前を持っています。
- 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、上のフィールド編集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・ライブラリー・テーブル中で定義されたフットプリント・ライブラリーの名前であり、FPNAME は LIBNAME ライブラリー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・ライブラリー・テーブル” を参照)

12.10 電源シンボル

電源シンボルは通常のシンボルと同様に作成します。Power.lib のような専用のライブラリーにそれらを集めると便利かもしれません。電源シンボルはグラフィカルなシンボルと “非表示電源” タイプのピンから構成されます。電源ポートのシンボルは回路図入力ソフトウェアで他のシンボル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 (名前は +5V ネットへの接続を確立するために重要) 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。
- シンボルの定数を +5V とします。
- シンボルのリファレンスは \#+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。慣例では、# で始まるリファレンス・フィールドを持つ全てのシンボルは、シンボル・リストとネットリストのどちらにも現れず、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、他のシンボルを雛型として使用すると簡単に作成できます。:

- 既存の電源シンボルを読み込む。
- 新しい電源シンボルの名前でピン名を編集する。
- 電源ポートの値を表示したい場合、定数フィールドをピンと同じ名前に編集する。
- 新しいシンボルを保存する。

Chapter 13

LibEdit - シンボル

13.1 概要

シンボルは、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・プロセッサーで使われる関連テキスト、フィールド：ネットリスト、シンボル・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。シンボル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、シンボル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、部品の調査、ライブラリーのメンテナンスや使用が容易になります。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シンボル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、特にライブラリーからシンボルを選択する場合に表示されます。

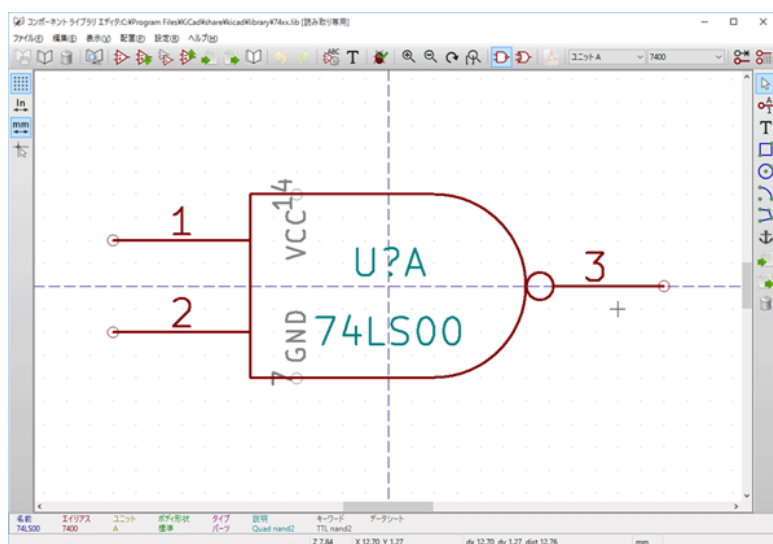
シンボルはまた、アンカー・ポイントを持っています。回転やミラーはこのアンカー・ポイントに対して相対的に行われ、配置時にはこの点が基準位置として使用されます。このように、このアンカーを正確に配置することは重要です。


シンボルは、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シンボルの数をかなり減らすことができます。(例えば、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、最終的にシンボルはライブラリー (メーカーやテーマ別に分類された) として配布されます。

13.2 シンボル・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。




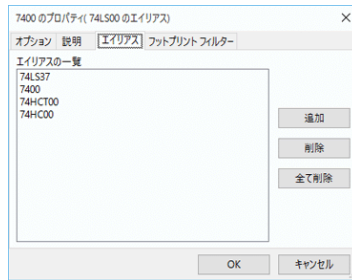
アンカーは、アイコン  を選択し、新しくアンカーを配置したい位置を左クリックすることで再配置できます。図は自動的に新しいアンカーポイントで再センタリングされます。

13.3 シンボルのエイリアス

エイリアスとは、ライブラリー内での同じシンボルに対応した別名のことで、類似したピン配列、表現を持つシンボルは、いくつかのエイリアスを持った 1つのシンボル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ーを構築することができます。エイリアスを使うことで、よりライブラリー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。



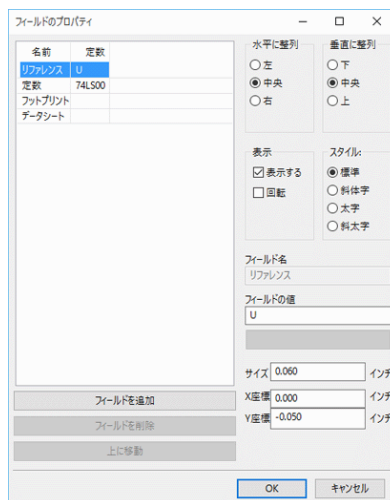
このように希望のエイリアスを追加したり、削除することができます。ただし、現在のエイリアスは編集されているので、明示的に削除することはできません。

全てのエイリアスを削除するには、まずルート・シンボルを選択することが必要です。エイリアスの最初のシンボルは、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。

13.4 シンボルのフィールド

フィールド編集ツールは **T** アイコンから実行できます。


4つの特殊フィールド (シンボルに付属するテキスト) と、設定可能なユーザー・フィールドがあります。

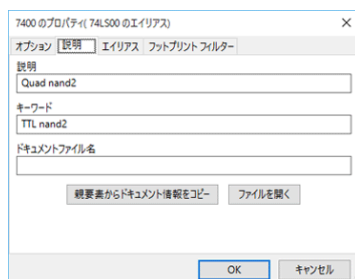


特殊フィールド

- リファレンス (Reference) : 部品の参照番号 (接頭辞)。
- 定数 (Value) : ライブラリー内のシンボル名であり、回路図中でのデフォルトの定数フィールド。
- フットプリント (Footprint) : 基板で使用す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。

13.5 シンボルのドキュメント

ドキュメント情報を編集するためには、メイン・ツールバーのアイコン  を左クリックしてシンボルのプロパティを開く必要があります。“説明”タブを開き、“ファイルを開く”ボタンを押してドキュメントを選択すると、“ドキュメント・ファイル名”にドキュメントの絶対パスが格納されます。



“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・シンボル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・シンボルからドキュメントの情報をコピーすることができます。

13.5.1 シンボルのキーワード

“キーワード”により、特定の選択基準（機能、技術的ファミリーなど）に従って、シンボル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。ライブラリー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・ファミリー
- 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・コレクター出力を持つ論理ゲートです。例えば、回路図エディター内で NAND2 OpenCol というキーワードでシンボルを検索すると、Eeschema はこれら 2 つのキーワードを持つシンボルのリストを表示します。

13.5.2 シンボルのドキュメント・ファイル名 (Doc)

特に ViewLib メニューおよびライブラリーに表示されたシンボルー覧からシンボルを選択する場合、様々なメニューに“説明”、“キーワード”、“ドキュメント・ファイル名”が表示されます。

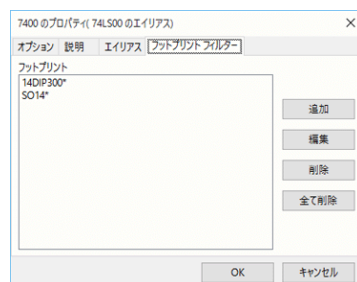
シンボルのプロパティに“ドキュメント・ファイル名”が存在する場合、回路図エディター・ソフト内でドキュメント・ファイルへのアクセスが可能であり、シンボルを右クリックして表示されるコンテキスト・メニューから利用できます。

13.5.3 関連付けられたドキュメント・ファイル (DocFileName)

次のようなものが相当します。添付ファイル (技術文書、アプリケーション回路例) 有効なファイル (pdf ファイル、回路図など)

13.5.4 CvPcb のフットプリント・フィルター

シンボル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・フィルター”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) フィルターとして働きます。空のリストは何もフィルターしません。



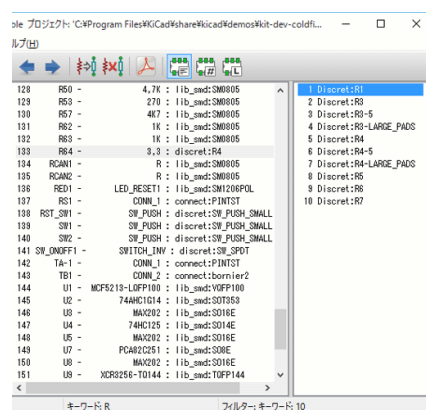
ワイルドカード文字が使用できます。

SO14* 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。

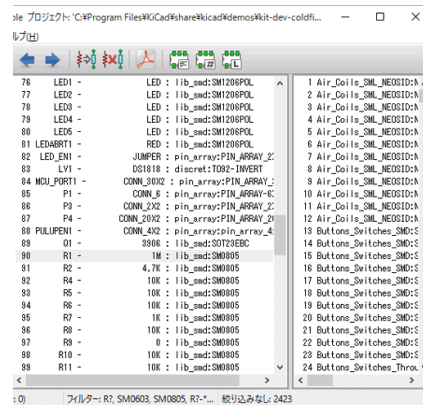
抵抗については、R? により R で始まる 2 文字の全てのフットプリントを示すことができます。

フィルター有り、無しのサンプルを示します。

フィルター有り



フィルターなし




13.6 シンボル・ライブラリー

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ー・ファイルを簡単に編集することができます。これはシンボル (三角形、AND、OR、ExOR ゲートなどの形状) の作成やその後の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ー・ディレクトリーに保存されます。これらのシンボルは一般的にそう多くないので、通常のシンボルのようにライブラリーには集められません。


13.6.1 シンボルの作成、エクスポート



シンボルは  ボタンでエクスポートすることができます。一般的には1つのグラフィックだけを作成することが可能で、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。

13.6.2 シンボルのインポート




インポートすることで、編集しているシンボル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。

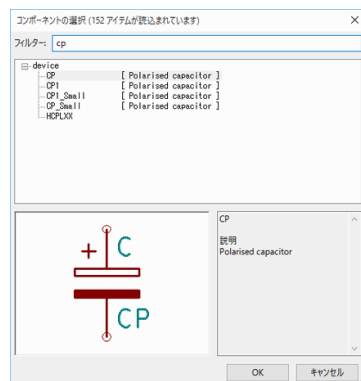
Chapter 14

シンボル・ライブラリー・ブラウザー

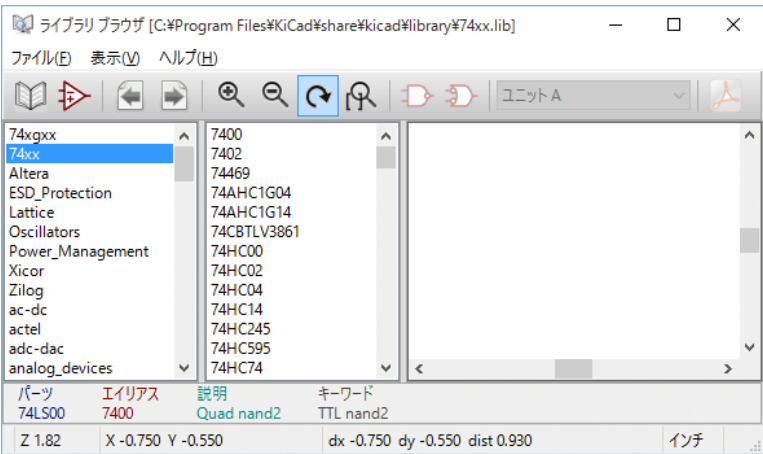
14.1 はじめに

シンボル・ライブラリー・ブラウザーを使うと、シンボル・ライブラリーの内容を素早く確認することができます。

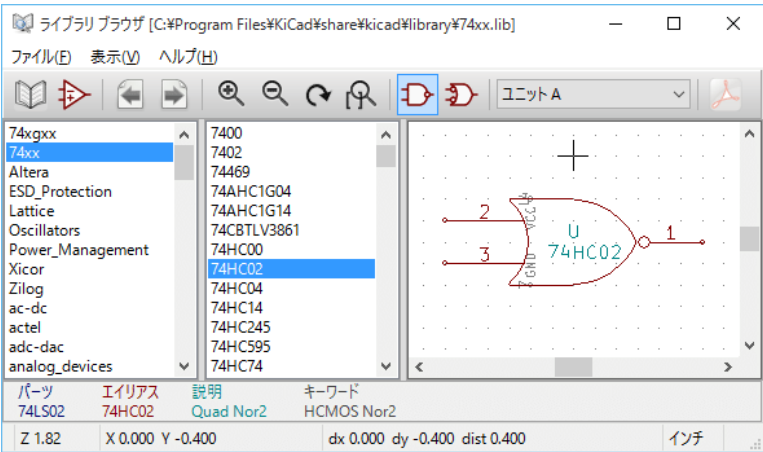
シンボル・ライブラリー・ブラウザーは、メイン・ツールバーにある  アイコンをクリックするか、“表示”メニューの“ライブラリー・ブラウザー”の選択、あるいは“シンボルを選択”ウィンドウにあるシンボルの画像をダブル・クリックして呼び出すことができます。



14.2 Viewlib - メイン・スクリーン



ライブラリーの内容を確認するには、左側のリストからライブラリーを選択する必要があります。選択したライブラリーに含まれる利用可能な全てのシンボルが左から 2 番目のパネルに表示されます。シンボル名を選択すると、シンボルの内容を確認できます。









14.3 シンボル・ライブラリー・ブラウザーの上部ツールバー

シンボル・ライブラリー・ブラウザー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:

	一覧より表示するライブラリーを選択します。
	一覧からシンボルを選択します。
	リストで選択されている、1 つ前のシンボルを表示します。

	リストで選択されている、1 つ次のシンボルを表示します。
	拡大／縮小（ズーム）ツールです。
	（代替シンボル表現が存在する場合）表現を選択します。
	複数のユニットを持つシンボルの場合、ユニットを選択します。
	関連ドキュメントが存在する場合、それ表示します。Eeschema の “シンボルの配置” ダイアログから呼び出された場合のみ有効です。
	ライブラリー・ブラウザーを閉じ、選択されたシンボルを Eeschema 上に配置します。このアイコンは、ライブラリー・ブラウザーが Eeschema から呼び出された時だけ表示されます。（“シンボルを選択” の画面でシンボルをダブルクリック）

Chapter 15

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

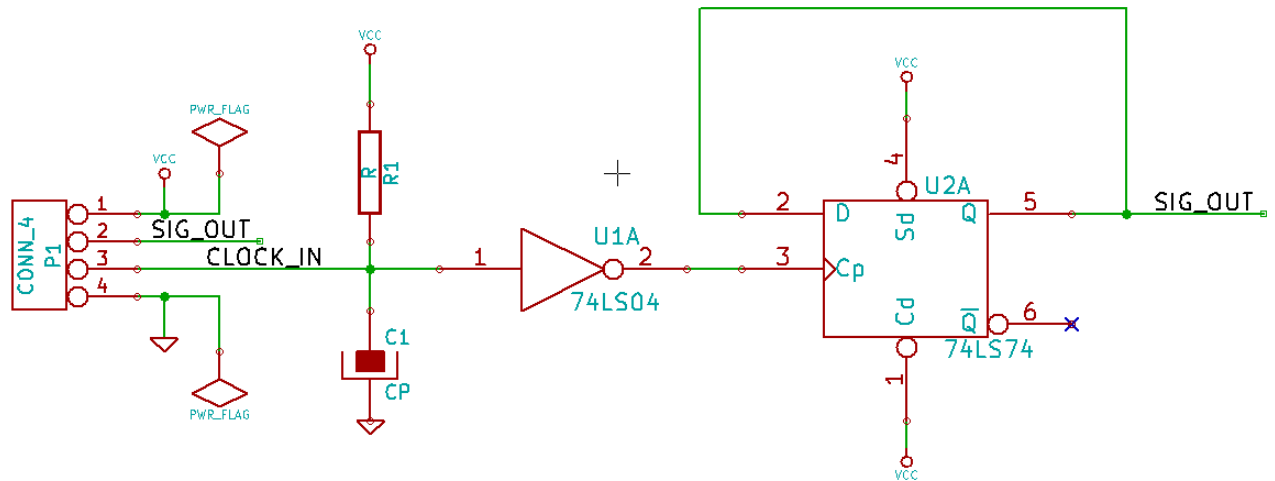
15.1 中間ネットリスト

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。

15.1.1 回路図のサンプル



15.1.2 中間ネットリストのサンプル

上記の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>
  </sheetpath>
</comp>
<comp ref="R1">
  <value>R</value>
  <libsource lib="device" part="R"/>
  <sheetpath names="/" tstamps="/">
    <tstamp>4C6E208A</tstamp>
  </sheetpath>
</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 & 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"/>
  </net>
</nets>
```

```
<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>
```

15.2 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ー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ーは、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。

15.3 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>

15.3.1 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.xml test. ↵
tmp
```

15.3.2 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: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
```

15.3.3 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    "
"> <!--new line CR, LF -->
]>
```

```

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

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

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

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

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

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

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

```

```

<xsl:template match="comp">
  <xsl:text> ( </xsl:text>
  <xsl:choose>
    <xsl:when test = "tstamp != '' ">
      <xsl:apply-templates select="tstamp"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>00000000</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text> </xsl:text>
  <xsl:choose>
    <xsl:when test = "footprint != '' ">
      <xsl:apply-templates select="footprint"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>$noname</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text> </xsl:text>
  <xsl:value-of select="@ref"/>
  <xsl:text> </xsl:text>
  <xsl:choose>
    <xsl:when test = "value != '' ">
      <xsl:apply-templates select="value"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>"~"</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:text>&nl;</xsl:text>
  <xsl:call-template name="Search_pin_list" >
    <xsl:with-param name="cmplib_id" select="libsource/@part"/>
    <xsl:with-param name="cmp_ref" select="@ref"/>
  </xsl:call-template>
  <xsl:text> )&nl;</xsl:text>
</xsl:template>

<!--
  This template search for a given lib component description in list
  lib component descriptions are in /export/libparts,
  and each description start at ./libpart
  We search here for the list of pins of the given component
  This template has 2 parameters:
    "cmplib_id" (reference in libparts)
    "cmp_ref"   (schematic reference of the given component)
-->
<xsl:template name="Search_pin_list" >

```

```

<xsl:param name="cmplib_id" select="0" />
<xsl:param name="cmp_ref" select="0" />
  <xsl:for-each select="/export/libparts/libpart">
    <xsl:if test = "@part = $cmplib_id ">
      <xsl:apply-templates name="build_pin_list" select="pins/pin">
        <xsl:with-param name="cmp_ref" select="$cmp_ref"/>
      </xsl:apply-templates>
    </xsl:if>
  </xsl:for-each>
</xsl:template>

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

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

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

```

```

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

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

<xsl:template name="Search_pin_netname" match="node">
    <xsl:param name="cmp_ref" select="0" />
    <xsl:param name="pin_num" select="0" />
    <xsl:param name="pin_cnt_in_net" select="0" />

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

```

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

</xsl: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 )
)
)
*
```

15.3.4 Eeschema プラグイン・インタフェース

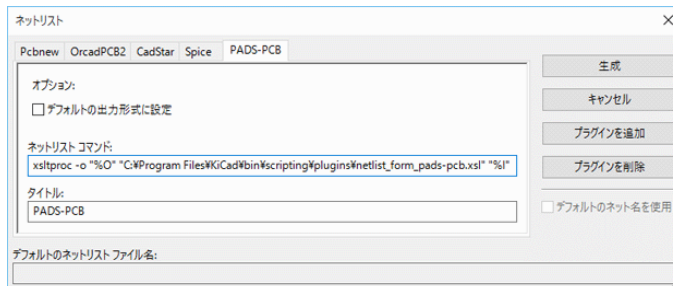
中間ネットリストの変換は Eeschema の中で自動的に実行させることができます。

15.3.4.1 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:



15.3.4.2 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。

15.3.4.3 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 ユーザー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。

15.3.4.4 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.xml "%I"
```

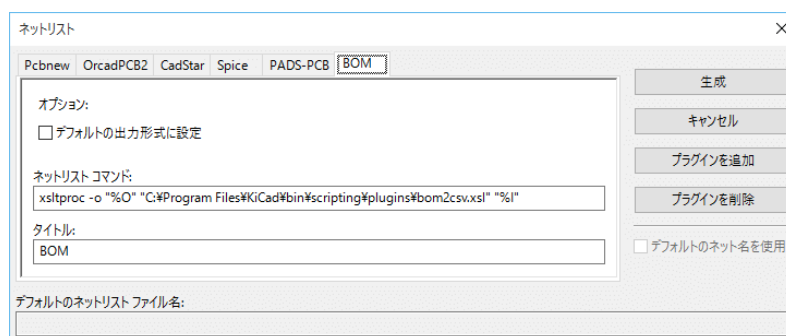
Linux 環境の場合:

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

15.3.5 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.xml 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.xml* とします。

15.4 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。)

15.5 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="/" />
    </comp>
  </components>
</export>
```

```
<tstamp>4C6E2094</tstamp>
<comp ref="R1">
  <value>R</value>
  <libsource lib="device" part="R"/>
  <sheetpath names="/" tstamp="/" />
  <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>
```

15.5.1 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
- ライブラリーパーツセクション
- ライブラリーセクション
- 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>
```

15.5.2 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。

15.5.3 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	そのコンポーネントが含まれているライブラリー名
part	ライブラリー中に登録されているコンポーネント名
sheetpath	階層内のシートのパス（回路図階層全体の中で、その回路図シートの位置を明確にするために利用される）
tstamps (time stamps)	回路図ファイルのタイムスタンプ
tstamp (time stamp)	コンポーネントのタイムスタンプ

15.5.3.1 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。

15.5.4 ライブラリーパーツ・セクション

このライブラリー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ー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：

- The allowed footprints names (names use wildcards) delimiter <fp>.
- <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、ピンの電氣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

15.5.5 ライブラリー・セクション

ライブラリーセクションは <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ー情報を含みます。

```

<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>

```

15.5.6 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

15.6 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>

15.6.1 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。

15.6.2 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* ]  
[ *....* ]
```

15.6.3 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、`xseltproc` 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。

15.6.4 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

15.6.5 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>

Chapter 16

シミュレーター

Eeschema は、**ngspice** をシミュレーション・エンジンとして使用した電気回路シミュレーターを内蔵しています。

シミュレーターを使う場合には、公式 *pspice* ライブラリーが役に立ちます。これには、電圧源や電流源、ngspice ノード順の仕様に合ったピン番号を持つトランジスターのような、シミュレーションに使われる共通シンボルが含まれています。

シミュレーターができることを説明するためのデモ・プロジェクトも用意されています。これらは *demossimulation* ディレクトリーにあります。

16.1 モデルの割り当て

シミュレーションを実行する前に、Spice モデルを部品に割り当てる必要があります。

各部品はモデルを 1 つだけ持つことができます。例えば複数ユニットを持っていたとしても 1 つのみで、この場合には、最初のユニットが指定されたモデルを持つことになります。

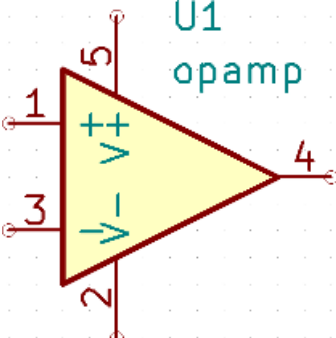
Spice 表記 (抵抗 R^* 、コンデンサー C^* 、インダクター L^*) のデバイス・タイプと一致したリファレンスを持つ受動部品は、暗黙的に割り当てられたモデルを持っており、プロパティの決定には定数フィールドの値が使われます。

注意

Spice 表記では M はミリ、 Meg はメガであることに留意して下さい。もし M をメガという接頭辞としたいのであれば、[シミュレーション設定ダイアログ](#) で設定することができます。

Spice モデルの情報はシンボルのフィールドにテキストとして保存されており、シンボル・エディターまたは回路図エディターで定義することができます。シンボル・プロパティのダイアログを開き、*Spice* モデルを編集ボタンをクリックして Spice モデル・エディターのダイアログを開いて下さい。

Spice モデル・エディターのダイアログは、モデルの種類に応じた 3 つのタブを持っています。全てのモデルの種類で共通のオプションは 2 つあります。

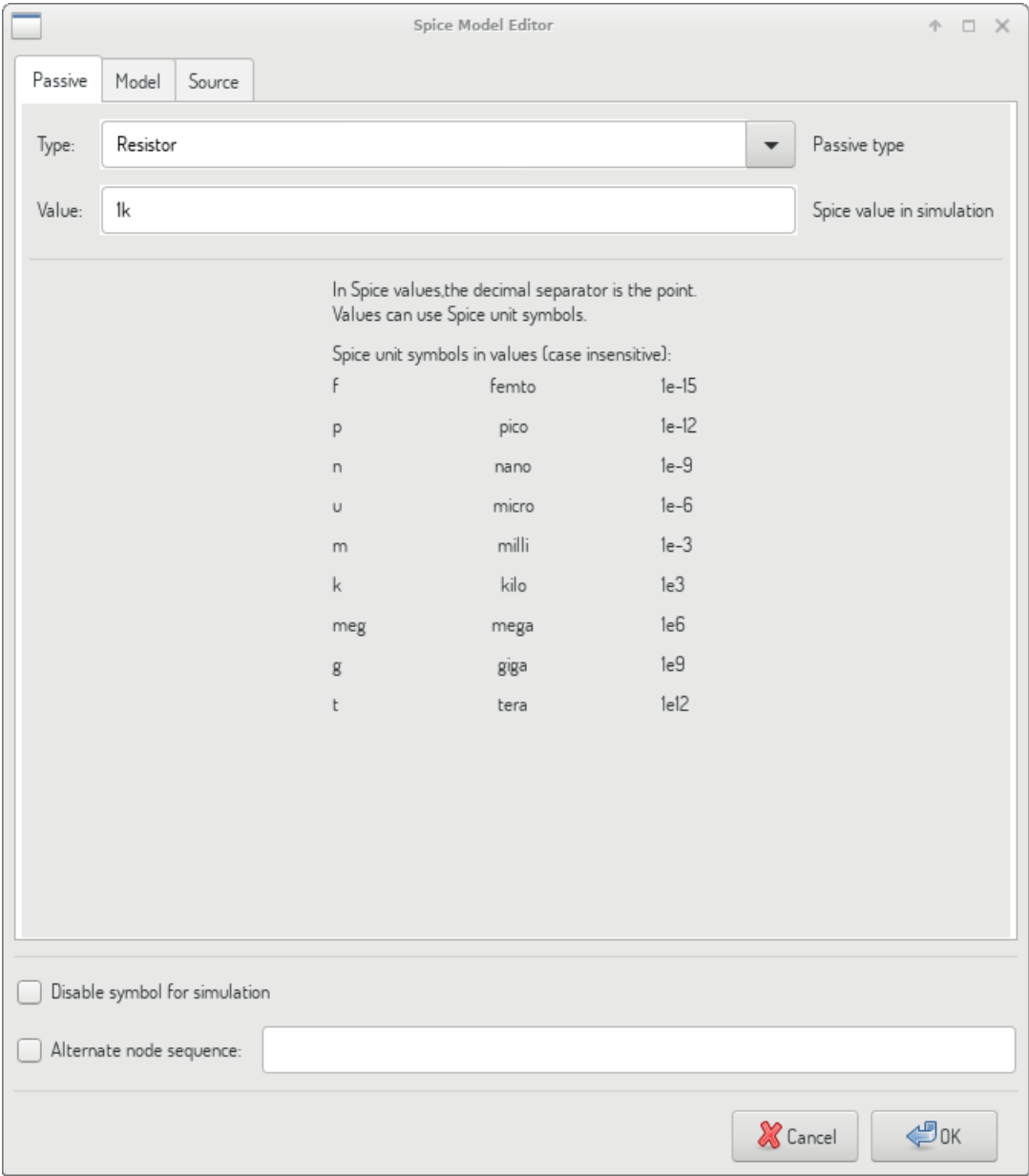
シミュレーションでシンボルを無効化	チェックされている場合、部品をシミュレーションから除外します。
代替ノード・シーケンス	<p>モデル・ノードのマッピングでシンボル・ピンの上書きを許容します。モデル設定に従うよう、ピン番号を指定して異なったマッピングを定義。</p> <p>例:</p> <ul style="list-style-type: none">* 接続:* 1:非反転入力* 2:反転入力* 3:正電源入力* 4:負電源入力* 5:出力 <pre>.subckt t1071 1 2 3 4 5</pre>  <p>上図にある Spice モデルのノードにシンボルのピンを一致させるには、次の値と一緒に代替ノード・シーケンス・オプションを使う必要があります。値: "1 3 5 2 4". これは Spice モデルのノード番号に対応したピン番号のリストです。</p>

16.1.1 パッシブ

`_パッシブ_`・タブで受動部品 (抵抗、コンデンサー、インダクター) の割り当てができます。これは、あまり使われないオプションで、部品のリファレンスが実際のデバイス・タイプと一致しない場合を除いて、通常受動部品は暗黙的に割り当てられたモデルを持っています。

注意

明示的に定義された受動部品のモデルは、暗黙的に割り当てられたモデルに優先します。これは、受動部品のモデルが割り当てられたなら、シミュレーション中はリファレンスと定数のフィールドが無視されることを意味します。従って、割り当てられたモデルの定数と回路図シートにある定数が異なっているときは、混乱を招くことがあります。

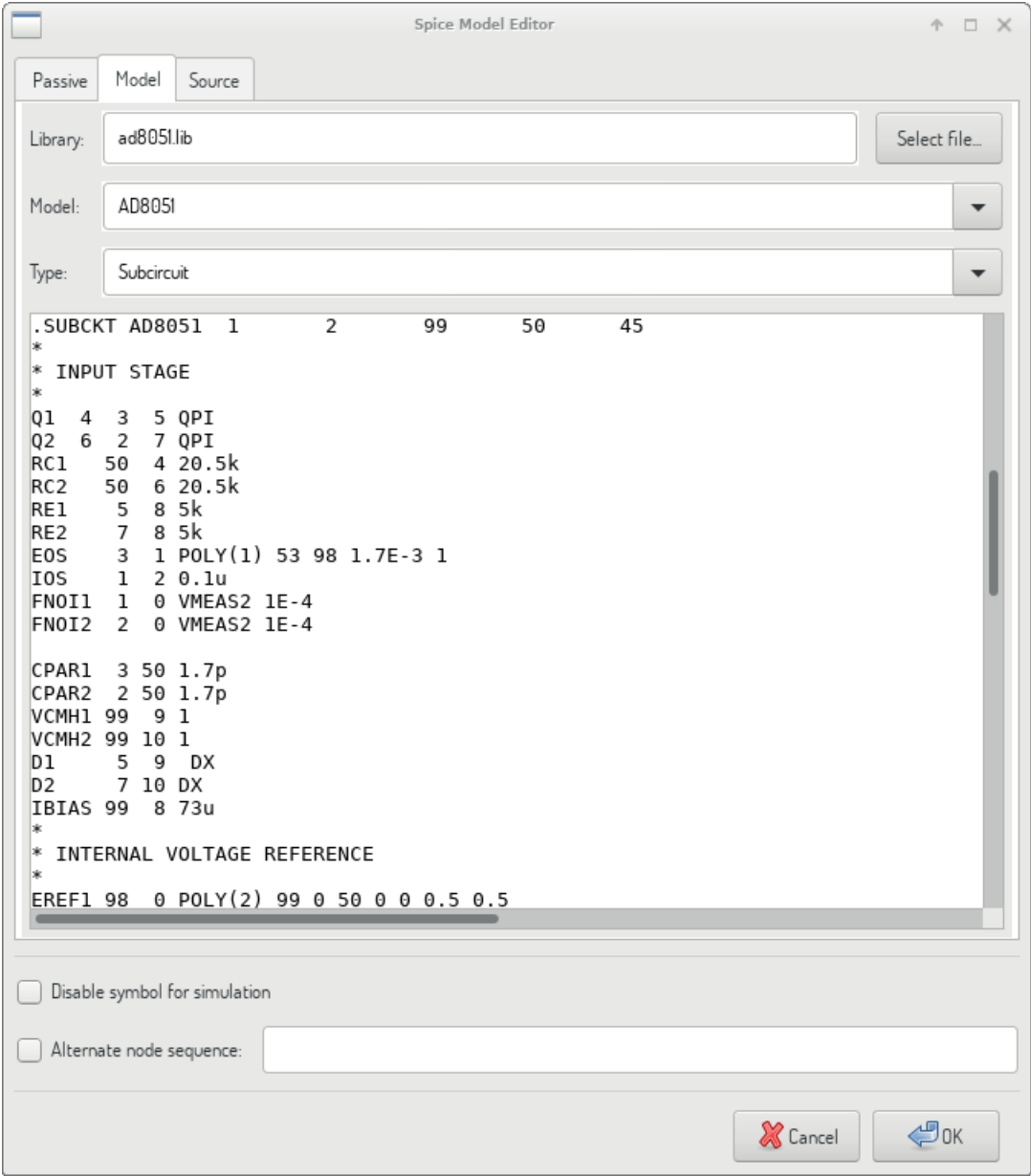


タイプ	デバイス・タイプ (抵抗、コンデンサーあるいはインダクター) を選択します。
定数	部品の属性 (抵抗値、静電容量またはインダクタンス) を定義します。値には Spice で一般的な単位の接頭辞を使えます。(テキスト入力フィールドの下にリスト表示) また小数点には"." を使わなければなりません。Spice は値の間に入っている接頭辞を正しく解釈できないことに注意して下さい。(例 1k5)

16.1.2 モデル

__モデル__・タブは半導体や外部のライブラリー・ファイルで定義された複雑なモデルの割り当てに使われます。Spice モデルのライブラリーは、多くの場合、部品メーカーによって提供されます。

メインのテキスト部分には、選択されたライブラリー・ファイルの内容が表示されます。通常は、ライブラリー・ファイル内部にある、ノード順を含んだモデル記述となります。



ライブラリー	Spice ライブラリー・ファイルのパス。このファイルは、 <code>.include</code> 指令を使って追加され、シミュレーターで使われます。
モデル	選択されたデバイス・モデル。ファイルが選択されているとき、リストには選択可能なモデルが表示されます。
タイプ	モデル・タイプ (サブ回路、BJT、MOSFET やダイオード) の選択。通常は、モデル選択時に自動でセットされます。

16.1.3 ソース

ソース _ ・タブは電源や信号源モデルの割り当てに使用されます。2つのセクション: _DC/AC 解析と 過渡応答解析があります。各々、シミュレーション・タイプに応じて、ソースのパラメーターを定義します。

_ ソース・タイプ _ ・オプションは全てのシミュレーション・タイプに適用されます。

Spice Model Editor

Passive Model Source

DC/AC analysis:

DC: Volts/Amps

AC magnitude: Volts/Amps AC phase: radians

Transient analysis:

Pulse Sinusoidal Exponential Piece-wise Linear

Initial value: Volts/Amps

Pulsed value: Volts/Amps

Delay time: seconds

Rise time: seconds

Fall time: seconds

Pulse width: seconds

Period: seconds

Source type:

☒ Voltage ☐ Current

☐ Disable symbol for simulation

☐ Alternate node sequence:

Cancel OK

ソースに関するより詳細な情報は、[ngspice ドキュメント](#)、chapter 4 (Voltage and Current Sources) を参照して下さい。

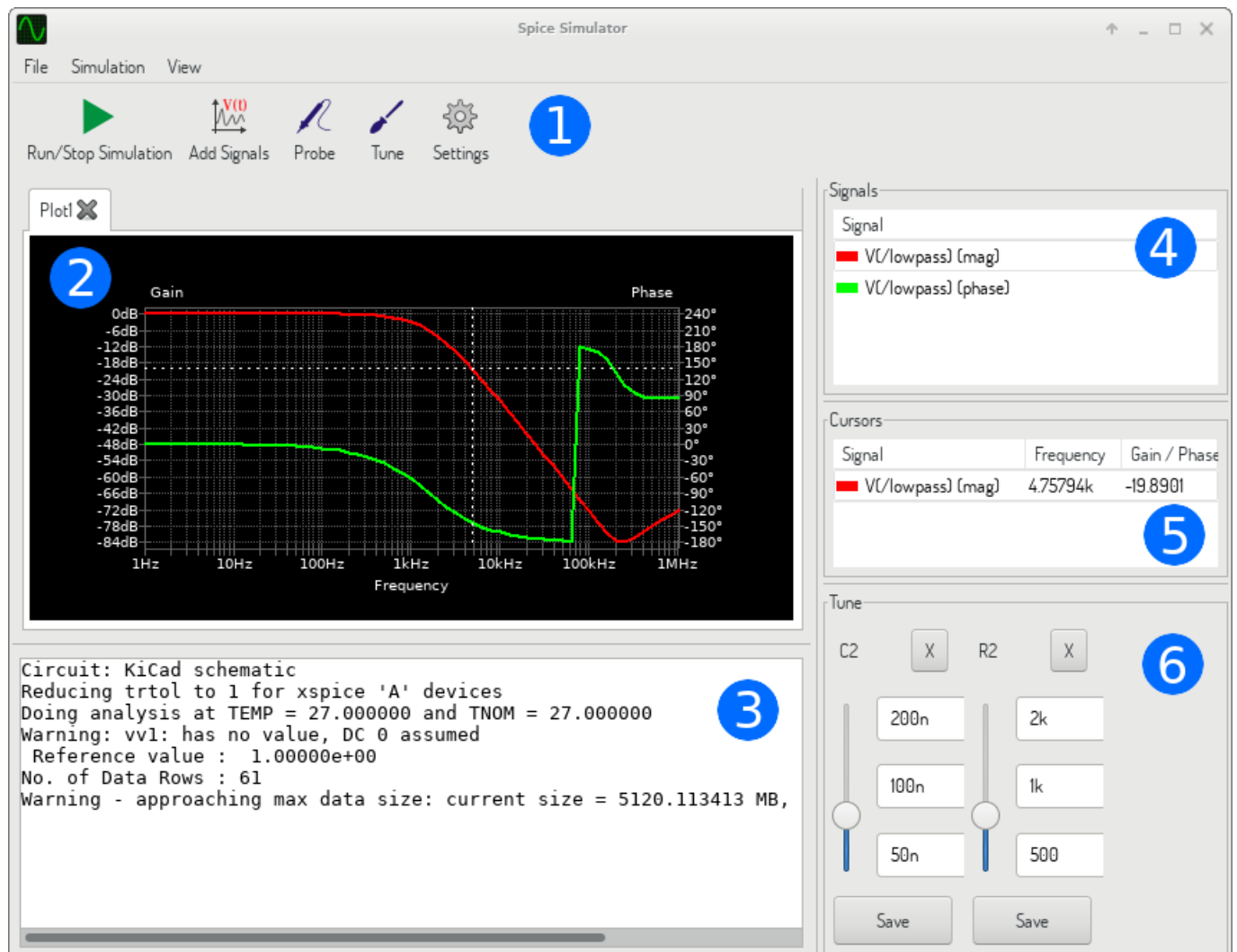
16.2 Spice 指令

回路図シートのテキスト・フィールドで Spice 指令を追加することが可能です。この方法はデフォルトのシミュレーション・タイプを定義する時に便利です。この機能はドットで始まる Spice 指令 (例".tran 10n 1m") に限定され

ます。テキスト・フィールドを使って追加の部品を配置することはできません。

16.3 シミュレーション

シミュレーションを開始するには、回路図エディターのメニューから ツール → シミュレーターを選択して `_Spice
シミュレーター _` ダイアログを開きます。



ダイアログはいくつかのセクションに分かれています。

- ツールバー
- プロット・パネル
- 出力コンソール
- 信号リスト
- カーソル・リスト
- 調整パネル

16.3.1 メニュー

16.3.1.1 ファイル

新規プロット	プロット・パネルに新しいタブを作成する。
ワークブックを開く	プロットされた信号のリストを開く。
ワークブックを保存	プロットされた信号のリストを保存する。
画像として保存	アクティブなプロットを png ファイルにエクスポートする。
.csv ファイルとして保存	アクティブなプロットの元のポイント・データを .csv ファイルにエクスポートする。
シミュレーションを終了	ダイアログを閉じる。

16.3.1.2 シミュレーション

シミュレーションを実行	現在の設定を使ってシミュレーションを実行する。
信号を追加…	プロットする信号を選択するダイアログを開く。
回路図からプローブ	回路図の プローブ ・ツールを起動する。
コンポーネント定数を調整	調整 ・ツールを起動する。
SPICE ネットリストを表示 …	回路シミュレーション用に生成されたネットリストを表示するダイアログを開く。
設定…	シミュレーション設定ダイアログ を開く。

16.3.1.3 表示

ズーム・イン	アクティブなプロットをズーム・イン。
ズーム・アウト	アクティブなプロットをズーム・アウト。
スクリーンにフィット	全てのプロットを表示するようにズームを調整。
グリッドの表示	グリッドの可視性をトグルで切り替え。
凡例の表示	プロット凡例の可視性をトグルで切り替え。

16.3.2 ツールバー



上部ツールバーは頻繁に繰り返し実行される機能を呼び出すためのものです。

実行/停止	シミュレーションの開始と停止。
信号を追加	プロットする信号を選択するダイアログを開く。
プローブ	回路図の プローブ ・ツールを起動する。
調整	調整 ・ツールを起動する。
設定	項16.3.10を開く。

16.3.3 プロット・パネル

シミュレーション結果をプロットして可視化する。開かれている別のタブにも複数プロットすることができますが、シミュレーション実行時に更新されるのはアクティブなタブだけです。このように、異なったシミュレーションの実行結果を比較できるようになっています。

プロットは、[表示](#)・メニューを使って、グリッドと凡例の可視性をトグル動作でカスタマイズ可能です。凡例が表示されているときは、ドラッグで位置を変更することができます。

プロット・パネルでの操作:

- マウス・ホイールのスクロールで、ズーム・イン/ズーム・アウト
- コンテキスト・メニューを開いて右クリックで、表示の調整
- 矩形選択で、選択された領域をズーム
- カーソルのドラッグで、座標の変更

16.3.4 出力コンソール

出力コンソールは、シミュレーターからのメッセージを表示します。エラーや警告がないことの確認用に出力コンソールをチェックすることをお勧めします。

16.3.5 信号リスト

アクティブなプロットで表示された信号のリストを表示します。

信号リストでの操作:

- コンテキスト・メニューを開いて右クリックで、信号の非表示またはカーソルの ON/OFF
- ダブル・クリックで、信号の非表示

16.3.6 カーソル・リスト

カーソルとその座標のリストを表示します。各信号には表示されたカーソルが 1 つあります。カーソルの可視性は[信号](#)・リストを使って設定されます。

16.3.7 調整パネル

[調整](#)・ツールでピックアップされた部品を表示します。調整パネルでは、部品定数の変更によるシミュレーション結果への影響を迅速に調べることができます。(部品定数が変更される毎に、シミュレーションが再実行されてプロットが更新されます。)

各部品には、関係する幾つかのコントロールがあります:

- 最上部のテキスト・フィールドには、部品定数の最大値がセットされます。

- 中間のテキスト・フィールドには、実際の部品定数がセットされます。
- 最下部のテキスト・フィールドには、部品定数の最小値がセットされます。
- スライダーを使うと、部品定数をスムーズに変換できます。
- `_保存_` ボタンを押すと、回路図の部品定数はスライダーで選択された値に変更されます。
- `_X_` ボタンを押すと、調整パネルから部品が削除されて部品定数は元の値に復元されます。

3つのテキスト・フィールドでは、Spice での単位の接頭辞を使用できます。

16.3.8 調整ツール

調整ツールは、調整する部品をピックアップするためのものです。

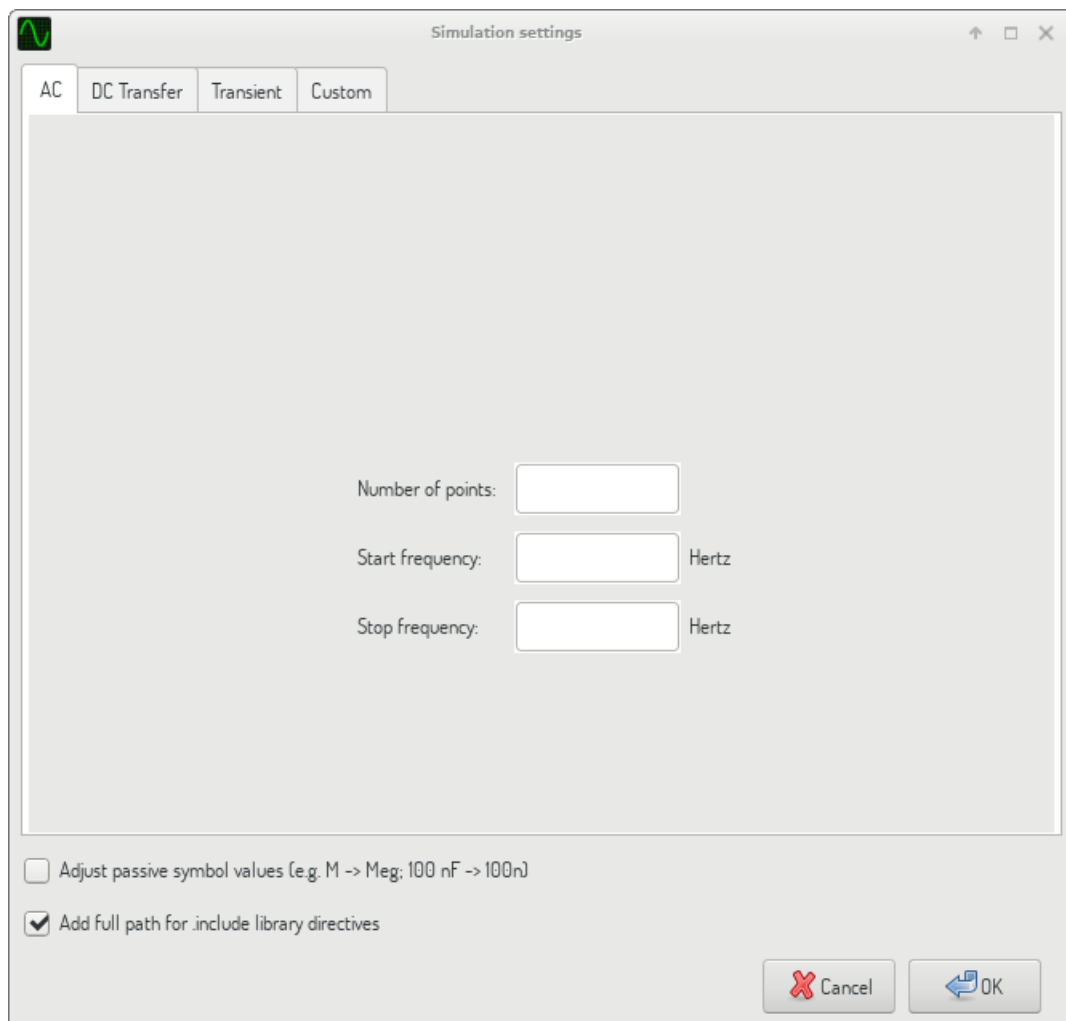
調整する部品を選択するには、ツールを起動してから回路図エディターで部品をクリックします。選択された部品は、[調整](#)・パネルに表示されます。調整できるのは受動部品だけです。

16.3.9 プローブ・ツール

プローブ・ツールを使うと、プロットする信号を簡単に選択できます。

プロットする信号を追加するには、ツールを起動してから回路図エディター上で対応する線をクリックします。

16.3.10 シミュレーションの設定



シミュレーション設定ダイアログは、シミュレーション・タイプとパラメーターを設定するためのものです。4つのタブがあります:

- AC
- DC 伝送
- 過渡応答
- カスタム

最初の3つのタブがシミュレーションのパラメーターを指定するためのフォームです。最後のタブは、ユーザーがカスタマイズしたシミュレーション設定用の Spice 指令を入力するためのものです。シミュレーション・タイプとパラメーターについてのより詳細な情報は、[ngspice documentation](#), chapter 1.2. で見つけることができます。

シミュレーションを設定する別の方法は、回路図上のテキスト・フィールドに [Spice 指令](#)を入力することです。シミュレーション・タイプに関連したテキスト・フィールドでの指令は、常にダイアログで選択された設定で上書きされます。これは、シミュレーション・ダイアログを使ってシミュレーションを開始したなら、シミュレーターを再起動するまでダイアログが回路図の指令を上書きすることを意味します。

全てのシミュレーション・タイプで共通のオプションが2つあります:

受動部品シンボルの定数を調整	一般的な部品定数の表記を Spice 表記へ変換するために受動シンボルの定数を置換する。
.include ライブラリー指令の絶対パスを追加	前にパスが追加された Spice モデル・ライブラリー・ファイルの名前、ライブラリー・ファイルのフル・パス。通常、ngspice はライブラリー・ファイルにアクセスするためにフル・パスを必要とします。