Ver0.2/マニュアル

http://checkerwiki.scoutlabo.com/index.php?Ver0.2/%E3%83%9E%E3%83%8B%E3%83%A5%E3%82%A2%E3%83%AB

Ver0.2 マニュアル

おすすめ

- このツールで現在一番有効な情報は接続状況がひと目で分かることなので。
  一度、手元の回路図でチェックを通して、気になるデバイスのオブジェクトページを見て頂くのがオススメです。
- また、回路図とNETリストの不整合について
  NETリストから生成されるこのツールの結果と回路図を照らし合わせ、確認していただくこともオススメです。
- FPGAのピンアサインの確認。デバッグ時コネクタアサインの確認など、NETリストを追わなくても、全体を見渡すことが出来ます。
- チェックについては、まだ最低限のレベルになっています。設計にも関わることなので、よく吟味してお使いください。

エラー報告

i. 現在、チェックをスタートするとシステムディレクトリの削除の失敗なのか、作成の失敗なのかエラーが出る場合があるようなので
その場合はシステムディレクトリを削除後、再度チェックをしてみてください。サンプルプロジェクト内に削除するバッチファイルを入れました。

インストール

- 公式サイトからダウンロードしてください。

```
http://download.scoutlabo.com/top/scoutchecker/
```

サンプルプロジェクト

サンプルプロジェクト

- 回路はほんと適当に書いたものです。もう少し余裕ができたら、もう少しマシな回路に変更します。
- システム側でアサインされてない電源の出力を手動で補った部分も入っています。
- 設定ファイル(ETCファイル)の書き方など、これを参考にしてください。

使い方

- ここで使い方を示す。

[準備段階]

1. 解析したい回路図のネットファイルと部品表ファイルを用意。
2. 設定ファイルを作成。
3. プロジェクトディレクトリを作り、その下に子ディレクトリを作り、回路図ネットリストと部品表ファイル、設定ファイルを保存。

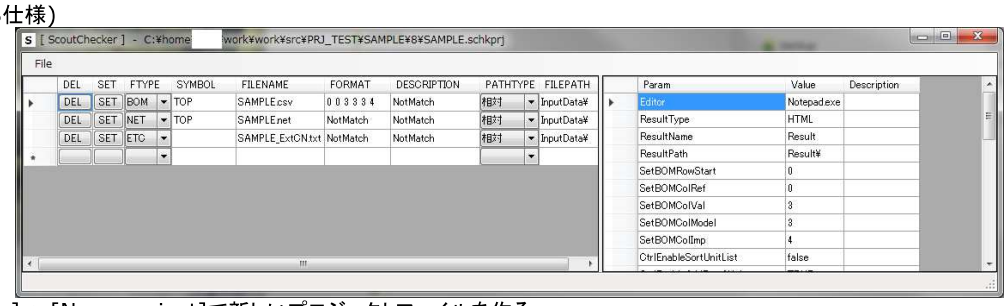
↑

[プロジェクトファイル作成段階] †

- 新規にプロジェクトファイル作成

1. ScoutChecker?を起動すると、以下のような画面が出る。

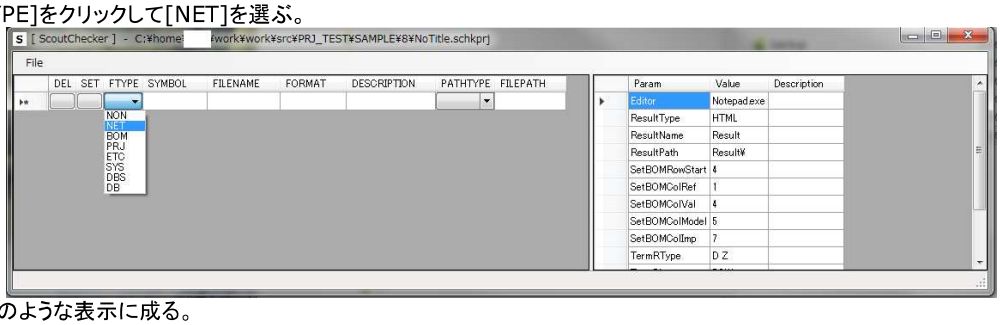
2. [File]->[Load project]をクリックすると、以下のような画面が出る。(前に動かしていたプロジェクトファイルで立ち上がる仕様)



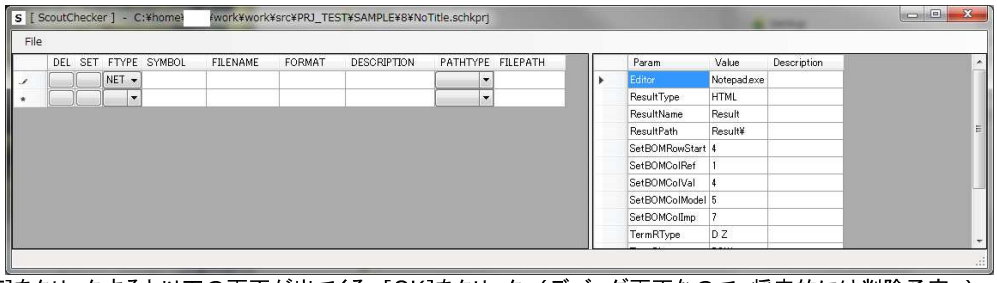
3. [File]->[New project]で新しいプロジェクトファイルを作る。

- NETファイルを登録する。

1. [FTYPE]をクリックして[NET]を選ぶ。



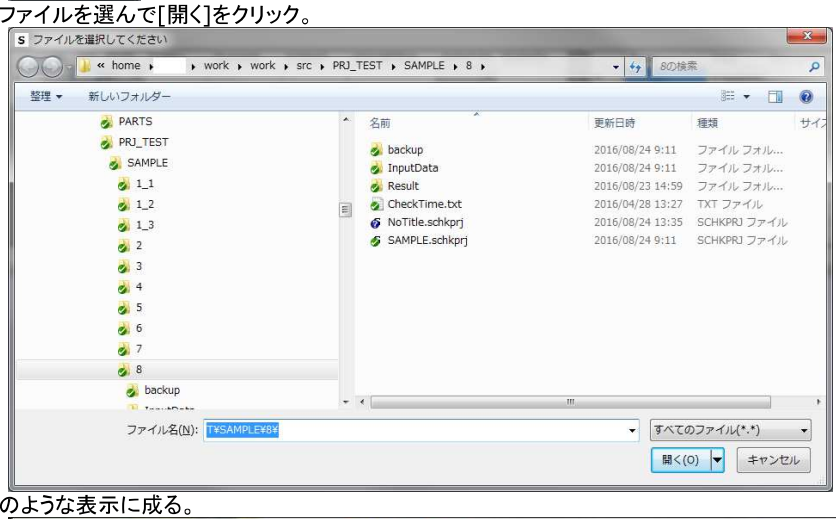
2. 以下のような表示に成る。



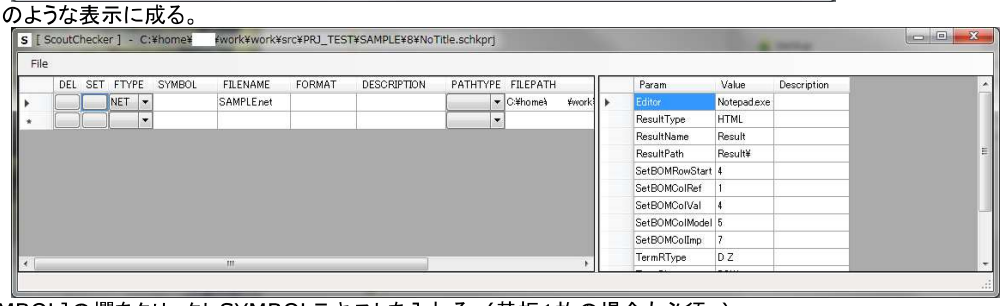
3. [SET]をクリックすると以下の画面が出てくる。[OK]をクリック。(デバッグ画面なので、将来的には削除予定。)

4. [OK]をクリック。(デバッグ画面なので、将来的には削除予定。)

5. NETファイルを選んで[開く]をクリック。

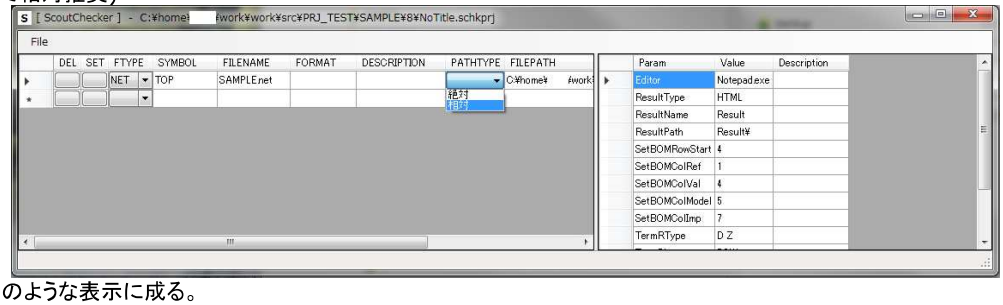


6. 以下のような表示に成る。

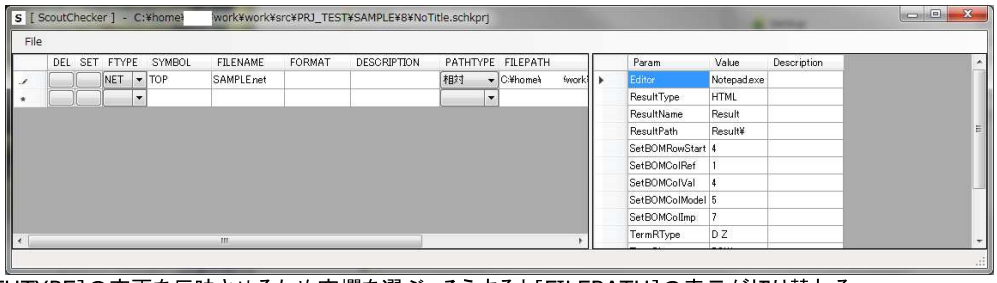


7. [SYMBOL]の欄をクリックしSYMBOLテキストを入れる。(基板1枚の場合も必須。)

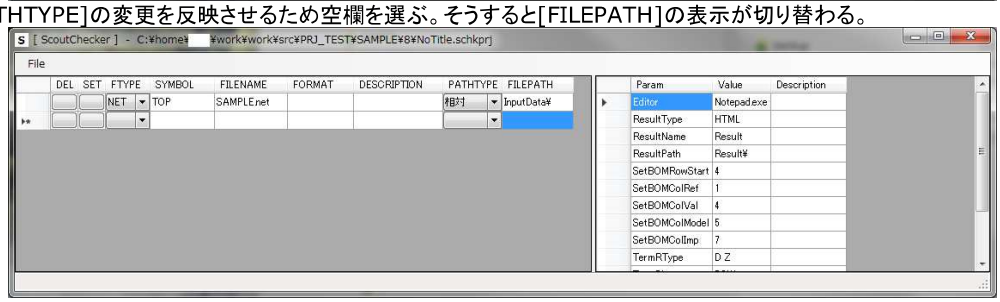
8. [PATHTYPE]をクリックして相対を選ぶ。(これは必要な場合のみ行う。プロジェクトディレクトリごと移動するのに便利なので相対推奨)



9. 以下のような表示に成る。

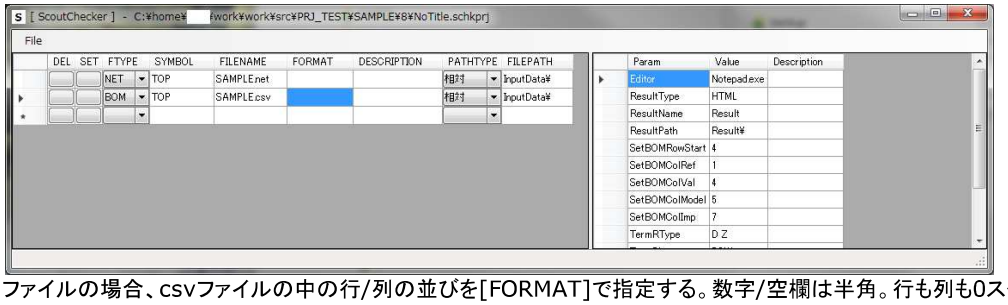


10. [PATHTYPE]の変更を反映させるため空欄を選ぶ。そうすると[FILEPATH]の表示が切り替わる。



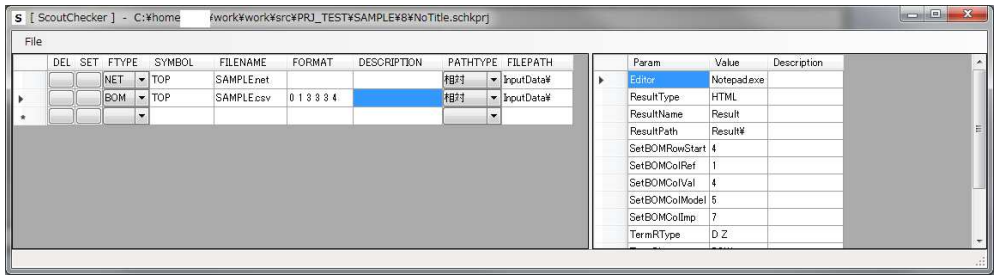
- BOMファイルを登録する。

1. 同様の手順でBOMファイルも登録する。対応するNETファイルとBOMファイルのSYMBOLテキストは同じに揃えてください。



2. BOMファイルの場合、csvファイルの中の行/列の並びを[FORMAT]で指定する。数字/空欄は半角。行も列も0スタート(最初の列/行は0)。

1文字目	2文字目	3文字目	4文字目	5文字目	6文字目
スタート行	空欄	配置番号の列	空欄	VALUEの列	空欄	MODEL(型番)の列	空欄	メーカーの列	空欄	員数の列

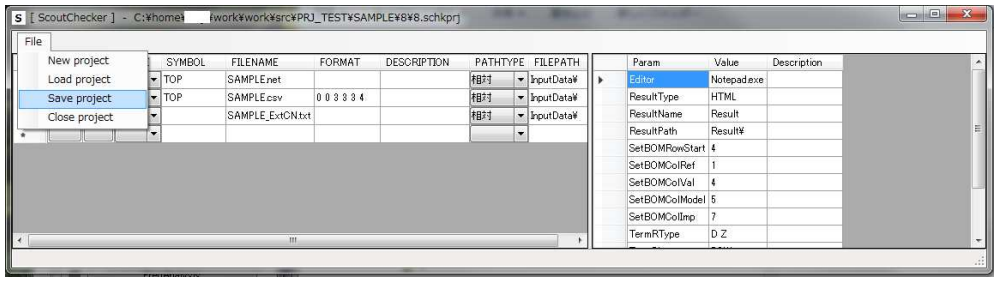


- ETCファイルを登録する。

1. ETCファイルもNETファイルと同様の手順で登録する。

- プロジェクトファイルを保存する。

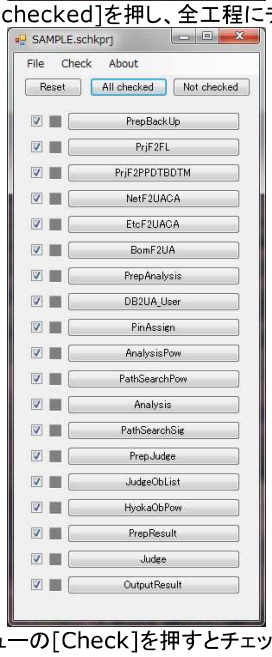
1. プロジェクトファイルを保存する。



[チェック段階] [+]

1. プロジェクトファイルのダブルクリックで、ScoutChecker?を起動。すると以下のような画面が出る。

2. [All checked]を押し、全工程にチェックが入ったことを確認する。

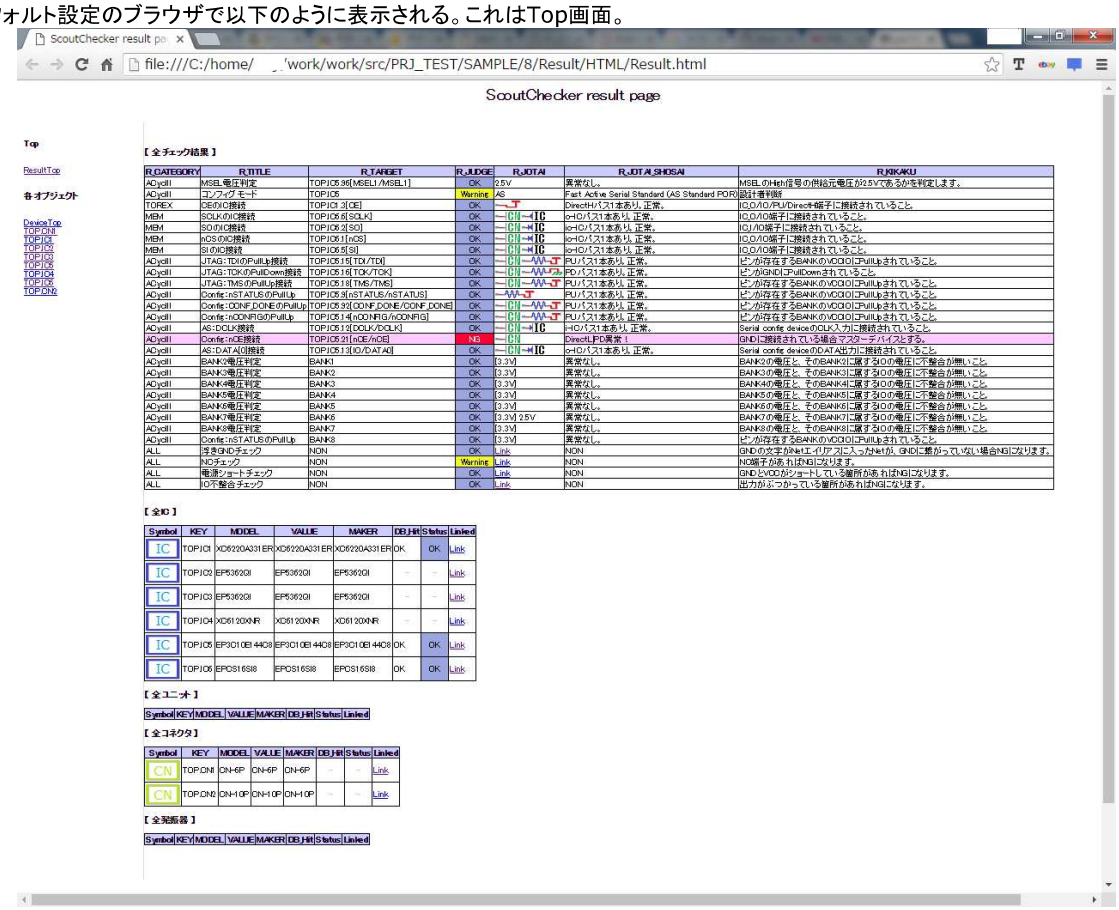


3. メニューの[Check]を押すとチェックがスタートする。

4. 全部の工程に緑のインジケータが着いたら。チェック終了。
5. デフォルト設定のブラウザが自動で立ち上がり結果が表示される。

[結果確認段階] †

- デフォルト設定のブラウザで以下のように表示される。これはTop画面。



  - 全チェック結果
    - ここに各デバイスの電源端子チェック以外の結果が表示される。

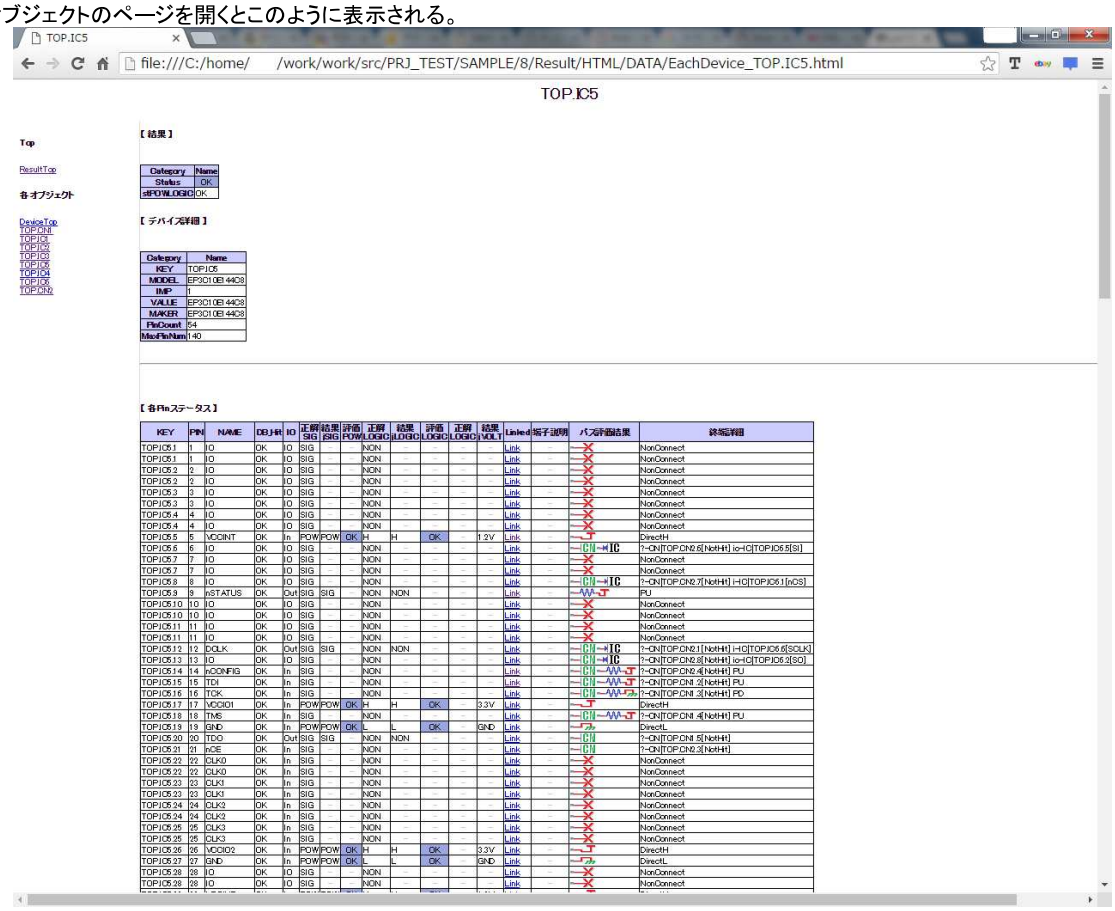
項目名	内容
R_CATEGORY	チェックの分類カテゴリー
R_TITLE	チェックタイトル
R_TARGET	チェック対象
R_JUDGE	判定結果
R_JOTAI	チェック状態
R_JOTAI_SHOSAI	チェック結果の説明

R_KIKAKU　　チェックの規格

- 各オブジェクトの表の項目
  - 各オブジェクトのページに飛ぶことが出る。

項目名	内容
Symbol	シンボルアイコン
KEY	プロジェクト内で一意のコード
MODEL	型番(BOMから引用)
VALUE	値(BOMから引用)
MAKER	メーカー名(BOMから引用、またはシステムから引用)
DB_Hit	システム側で情報が用意されてた場合OK
Status	電源端子の総合判定結果
Linked	各オブジェクトページへのリンク

- 各オブジェクトのページを開くとこのように表示される。



  - 【 結果 】

項目名	内容
Status	電源端子の信号タイプと論理の総合判定結果
stPOWLOGIC	電源端子の論理の総合判定結果

  - 【 デバイス詳細 】

項目名	内容
KEY	プロジェクト内で一意のコード
MODEL	型番(BOMから引用)
IMP	員数。0:未実装、1:実装
VALUE	値
MAKER	メーカー名

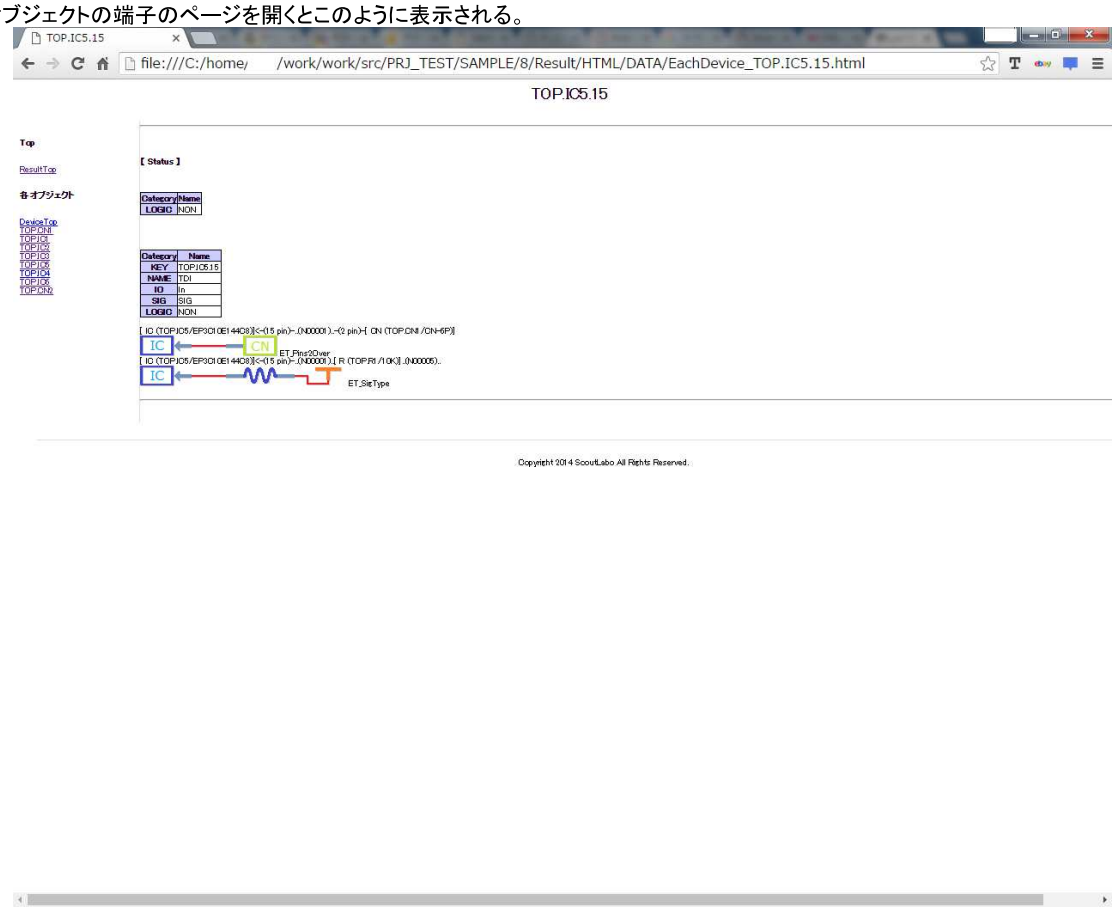
PinCount? ネットリスト上にあるこのオブジェクトに属するピン数

MaxPinNum?

- 【 各Pinステータス 】

項目名	内容
KEY	プロジェクト内で一意のコード
PIN	ピン番号
NAME	ピン名
DB_Hit	システム側で情報が用意されてた場合OK
IO正解	端子の方向性(In/Out/IO)
SIG結果	正しい信号タイプ(POW/SIG/LVDS)
jSIG評価	チェック結果の信号タイプ
POW正解	信号タイプがPOWの場合のみ、信号タイプが一致したらOKの判定が出る。
LOGIC正解	正しい論理(L/H)
jLOGIC評価	チェック結果の論理
LOGIC結果	論理の判定結果
jVOLT	チェック結果の電圧
Linked	端子ページへのリンク
端子説明	端子の説明
パス評価結果	この端子に繋がるパスリスト(アイコン)
終端詳細	接続先情報(テキスト)

- 各オブジェクトの端子のページを開くとこのように表示される。



- 各オブジェクトの表の項目
  - 各オブジェクトのページに飛ぶことが出る。

項目名	内容

きちんと結果を出すまで

- 正直、上記結果はある程度、手を加えた状態です。
- おそらく、最初はScoutChecker?側のデータベースが不十分だったりでなかなか思うように出てきません。
- なので、次の手順に従って情報を付与してください。

- まず、電源パスが全部通るようにする。

1. 欠けてる電源情報を追加
   - 電源ICでシステム側のアサインがされてない場合、アサインをETCファイルに書いて追加します。
   - 例:GigE.U6.2に3.7Vをアサイン

```
[>]KEY:GigE.U6.2,;IO:Out,SIG:POW,LOGIC:H,VOLT:3.7V,jIO:Out,jSIG:POW,jLOGIC:H,jVOLT:3.7V,;
```

2. 電源パスを遮る要素にスルー設定を入れる。
   - 電源パスを遮るダイオードなどにスルー設定をETCファイルに書いて追加します。
   - ダイオードは2pinのデバイスとは限らないためScoutChecker?側でスルーさせることが出来ません。
   - 例:IO.D1.CATHODEとIO.D1.ANODEを接続

```
[&]KeyA:IO.D1.CATHODE,KeyB:IO.D1.ANODE,;
```

- とりあえず、電源パスが全て通れば、そこそこの結果が出るようになります。お手間だとは思いますが、最初だけですので何卒よろしくお願いします。

詳細説明

語句説明

デバイス
: 部品のこと。回路上の部品全てがこれに当てはまります。

オブジェクト
: デバイス/ネットAliasをまとめてオブジェクトと呼ぶ。

端子(ピン)
: デバイスの端子。デバイスは必ず端子でネットAliasに接続されていることを前提にしています。

ネット**Alias**(ネット)
: ネットのこと。デバイスの端子間の接続のこと。

プロジェクト
: ScoutChecker?でチェックを行いたい回路(複数基板間接続も見たいなら、その全部)に対して1個定義する。

BOM及び**BOM**ファイル
: Bill of materialの略。部品構成表。回路図のシンボルに対して詳細情報を記載したもの。

NET及び**NET**ファイル
: ネットリスト。回路図CADが吐き出す結線情報。

ETCファイル
: 設定ファイル。ユーザーが設計に基いて、必要な情報を付与するためのファイル。

チェック
: ScoutChecker?で回路の検査を行うこと。

パス
: ここではデバイス間の接続をパスと呼ぶことにします。端子から終端条件のあるデバイスまたはネットAliasまでの接続を1つのパスとします。パスはネットAliasを共有出来ます。
(例:VCC端子には電源ICの出力端子、パスコンなどが繋がっていますが、VCC端子-電源出力端子/VCC端子-パスコン それぞれが一つのパスになります。)

終端
: パスを終わらせること。パスを終わらせるデバイス。

アサイン
: 端子やデバイス、ネットAliasなどに対して情報を付与すること。

スルー
: 3pin以上のデバイスでパスを通過させたい場合。スルー設定を入れます。

基板シンボル(**=SYMBOL**テキスト)
: 基板を特定するシンボル名。
プロジェクトファイルで定義します。

主に基板間の配置番号の被り対策に使います。

Key
このプロジェクト内でデバイス/ピン/ネットAliasを特定するために付ける名前。
配置番号のようなもの。
フォーマットは

```
基板シンボル. 部品配置番号. ピン番号
```

```
例：CMOS. CN3. 1
```

DEVKEY
端子が属するデバイスのKey。端子が自分の親デバイスを把握するために持つ。

システム対応
デバイスがScoutChecker?側で登録されている。ピンアサインなどがScoutChecker?側で出来ている。

Unit
デバイス／ピン／ネットを個々にUnitという単位で扱います。

UA
UnitArray?の略。プロジェクト内の全てのデバイス／ピン／ネット情報が、この中に入っています。

Con
デバイスーピン／ネットーピン接続情報を個々にConという単位で扱います。(全て1:1接続です)

CA
ConArray?の略。プロジェクト内の全てのデバイスーピン／ネットーピン接続情報が、この中に入っています。

準備段階

- 以下のファイルを作成する必要があります。各ファイルの作成について説明します。

ファイルの種類

- 基本的に拡張子に制限は無く(PRJを除く)。ScoutChecker?上で各ファイルに対して以下の属性を定義します。

ファイルタイプ	拡張子	名称	詳細
NET		ネットファイル	回路図のネットファイル。各CADの操作手順に従ってネットリストを作成ください。telesisフォーマットです。テキストファイル。
BOM		部品表ファイル	部品構成表のCSVファイル。部品構成表をcsv形式(,区切り、1行1部品)にして作成してください。
ETC		設定ファイル	プロジェクトに依存したScoutChecker?で必要な設定を書くテキストファイル。
PRJ	.schkprj	プロジェクトファイル	ScoutChecker?で使用するプロジェクトファイル。ScoutChecker?を起動して作成します。テキストファイル。

プロジェクトディレクトリ構造

- プロジェクトごとにプロジェクトディレクトリを作り、その下に子ディレクトリを作り必要なファイルを入れて使用します。
  プロジェクトファイルはプロジェクトディレクトリの直下に入れてください。

```
プロジェクトディレクトリ(ユーザー作成)
|. 「プロジェクトファイル」
|. プロジェクト全体を保管するディレクトリ。
|. プロジェクトファイルをここの直下に置いてください。
|. ディレクトリ名に制限はありません。
├ InputData?(ユーザー作成)
|.. 「ネットファイル」「部品表ファイル」「設定ファイル」
|.. チェックに必要なユーザーが用意するファイルを保存。
|.. ディレクトリ名は下記自動生成ディレクトリ名と被らなければ制限はありません。
|.. 複数作成可。
├ Result(自動生成)
|. |..・結果を出力するディレクトリ。
|. |..・以下全て相対パスなのでこのディレクトリごと移動しても表示に支障は出ません。
|. └ HTML(自動生成)
|.... |.. 「Result.html」
```

```
| .... | ..・HTML形式の結果を保存するディレクトリ。
| .... | ..・ここの直下にResult.htmlというファイルが生成されるので、
| .... | ..・結果を見る場合はまずこれを開いてください。
| .... ┗ DATA(自動生成)
| .....・各詳細ページファイル及びCSSファイルなどを格納
┗ backup(自動生成)
.....・プロジェクトが途中結果を吐き出すディレクトリ。。
```

↑

設定ファイルのフォーマット †

- 設定ファイルの書式は以下に従ってテキストファイルで作成します。
  行タイプによって、Unit数や何番目のUnitは何の意味など、個々に異なります。

		Unit								**Unit**			
行タイプ	**Info**		**Info**		･･･	**Info**		･･･	**Info**			行末
[ ]	ｶﾃｺﾞﾘ	**:**	値	**,**	ｶﾃｺﾞﾘ	**:**	値	**,**	･･･	ｶﾃｺﾞﾘ	**:**	値	**,** **;**
例 [&]	TYPE	:	Pin	,	TYPE	:	Pin	,	･･･	TYPE	:	Pin	, ; ･･･ TYPE : Pin , ;

ｶﾃｺﾞﾘ
: ScoutChecker?上で一意の項目を指す。

値
: ｶﾃｺﾞﾘに入れる値を指す。

Info
: ｶﾃｺﾞﾘと値の1セットをInfoと定義する。

Unit
: 複数Infoの集合体(Info1個でも可)をUnitと定義する。

行タイプ
: その行の扱い方タイプをここで定義。タイプを示す文字を[]で括る。

区切り文字
: : ; , [ ] など項目や値の区切りとなる決められた文字。

[]
: 行タイプを示す文字の区切り。

:
: Infoのｶﾃｺﾞﾘと値を区切る。

,
: Infoの終了を示す。

;
: Unitの終了を示す。

- 注意事項
  - 情報は1行で完結する。複数行にまたがらない。
  - 行が入れ替わっても意味は変わらない。(影響しても表示順だけ)
  - 区切り文字は全て半角英数字
  - 区切り文字,:;[]はユーザー内容部分では使用禁止。
  - 1行内に1個以上複数のUnit記載。
  - 1Unit内に1個以上複数のInfo記載。

↑

設定ファイルの作成方法 †

- 設定ファイルに入れるべき情報としては以下があります。
  - 外部電源アサイン(必須)
    - この回路に外部から供給される電源をアサインします。
  - 基板間接続コネクタアサイン/スルーアサイン
    - 基板同士のコネクタ接続をピン単位でアサインします。
    - 基板同士のコネクタは終端しない場合が多いので「終端しない」アサインを入れます。
    - ダイオードやEMIフィルタは3pin以上のモノもあるためパスをスルーさせるためここでピン間にスルー設定をアサインします。
  - 内部電源アサイン
    - ScoutChecker?が回路内の電源ICを認識出来なかった場合(未対応の場合)、手動でそのデバイスの出力端子に電源アサインを付与して電源パスを実現します。
      (電源パスは回路内全て実現しないとチェックは有効な結果を吐きません)

- 基板間コネクタ接続アサインの場合
  - [&]KeyA:Val1,KeyB:Val2,;
  - KeyA:Val1をInfo1、KeyB:Val2をInfo2とする。

行ﾀｲﾌﾟ	説明
[&]	Info1とInfo2が繋がっていることを定義する。方向性無し。

使用ｶﾃｺﾞﾘ	値	説明
KeyA/KeyB	Keyを入れる	KeyAとKeyBは等価に扱われます。

  - 例:CMOS.CN3.1とMAIN.CN2.1を接続

```
[&]KeyA:CMOS.CN3.1,KeyB:MAIN.CN1.1,;
```

- ダイオードスルーアサインの場合
  - ダイオードの端子名はネットリスト内の名前で書いてください。
  - フォーマットは「基板間コネクタ接続アサイン」と同じです。
  - この接続に方向性はありません両方から素通りが可能になります。
  - 例:IO.D1.CATHODEとIO.D1.ANODEを接続

```
[&]KeyA:IO.D1.CATHODE,KeyB:IO.D1.ANODE,;
```

  - 使用パラメータ

行ﾀｲﾌﾟ	説明
[>]	Unit1のKEYに合致するUnitをUAから検索しUnit2をアサイン

使用ｶﾃｺﾞﾘ	値	説明
KEY	基板シンボル.配置番号.ピン番号	回路全体の中でUnitを特定するための名前
TermEach?	NotTerm?	各Unitを終端させる場合に、このｶﾃｺﾞﾘにNotTerm?をアサインする。

  - 例:KEY:GigE.CN3を終端させないアサイン

```
[>]KEY:GigE.CN3,;TermEach:NotTerm,;
```

- 外部電源アサインをする場合
  - フォーマットは「基板間コネクタを終端させないアサイン」と同じ。
  - 使用パラメータ

使用ｶﾃｺﾞﾘ	値	説明
KEY	基板シンボル.配置番号.ピン番号	回路全体の中でUnitを特定するための名前
IO	I/O/IO	この端子の正解IOタイプを入れる。
SIG	POW/SIG	この端子の正解SIGタイプを入れる。
LOGIC	L/H	この端子の正解ロジックを入れる。
VOLT	数字V	この端子の正解電圧を入れる。
jIO	I/O/IO	実際の所のIOタイプを入れる。
jSIG	POW/SIG	実際の所のSIGタイプを入れる。
jLOGIC	L/H	実際の所のロジックを入れる。
jVOLT	数字V	実際の所の電圧を入れる。

  - 例:IO.CN2.10に電源12Vをアサイン

```
[>]KEY:IO.CN2.10,;IO:Out,SIG:POW,LOGIC:H,VOLT:12V,jIO:Out,jSIG:POW,jLOGIC:H,jVOLT:12V,;
```

- 外部電源としてGNDアサインをする場合
  - 例:IO.CN2.5にGNDをアサイン

```
[>]KEY:IO.CN2.5,;IO:Out,SIG:POW,LOGIC:L,VOLT:GND,jIO:Out,jSIG:POW,jLOGIC:L,jVOLT:GND,;
```

- 内部電源アサインの場合
  - 例:GigE.U6.2に3.7Vをアサイン

```
[>]KEY:GigE.U6.2,;IO:Out,SIG:POW,LOGIC:H,VOLT:3.7V,jIO:Out,jSIG:POW,jLOGIC:H,jVOLT:3.7V,;
```

動作概念

- ScoutCheckerではデバイスを端子/デバイスに分解し、ネットAliasと合わせて3種のUnitに分解して考えます。
- 接続は端子-デバイス/端子-ネットAliasの2種の1:1Unit間接続情報(Con)に分解して考えます。
  (ユーザーの設定する基板間コネクタ接続やスルー接続の端子-端子を合わせると3種)
- まず、ネットファイルからUnitリスト(UA:UnitArrayの略)とConリスト(CA:ConArrayの略)を生成します。
- ユーザー情報側からは部品表ファイル、設定ファイル。システム側からはMODELで判断してピン名やIO情報などをUAやCAに追加します。
- UAの電源アサインや出力端子を起点にCAを探索し終端するまでConを積み上げて行きます。これで一つのパスが生成されます。
- このパスリストから得られる情報を基に回路の評価を行います。

チェック

- チェックの詳細を説明していきます。おおまかに分けると2種類。各デバイス固有のチェックと汎用のチェックがあります。
- 各デバイスチェック
  - 各デバイス固有の端子ごとに、必要な接続があるか？必要な論理切り替えが可能か？をチェックします。
  - 各デバイスの設定端子などからデバイスがどういう状態になっているか？をチェックします。
- 汎用チェック
  - NC(Non connect)チェック、GNDネットが複数あるかチェックなど。

NCチェック

- ネットリストにNCは出てきません。なので、システム側で対応出来ているデバイスについてのみ、NC端子のチェックを行います。

入力端子オープンチェック**(Ver0.2.0.3**以降対応予定**)**

- 端子に入力端子アサインがある場合のみ、オープンのチェックを行います。
