Introduction to LPB Format

リリース ***1.0***

lpb-forum.com

2021 年 **05** 月 **09** 日

目次

第 1 章　はじめに　1

第 2 章　LPB フォーマットとは何か？　5
2.1 LPB Format とは？　5
2.2 LPB Format の構成　6
2.2.1 C-Format　6
2.2.2 R-Format　7
2.2.3 M-Format　8
2.2.4 G-Format　8
2.2.5 N-Format　9

第 3 章　XML の概要　10
3.1 XML 宣言　11
3.2 要素　11
3.2.1 要素の構文規則　11
3.2.2 要素の階層構造　11
3.2.3 トップ要素　12
3.3 属性　12
3.3.1 属性の構文規則　12

第 4 章　C-Format クイックスタート　14
4.1 コンデンサ　15
4.1.1 C-Format の構成　16
4.1.2 <header>要素　16
4.1.3 <global>要素　17
4.1.4 <module>要素　21
4.2 水晶振動子　30
4.2.1 <header>要素　31
4.2.2 <global>要素　31
4.2.3 <module>要素　33
4.3 MOSFET　37
4.3.1 C-Format の構成　39
4.3.2 <header>要素　39

4.3.3 <global>要素 . . . 39
4.3.4 <module>要素 . . . 44
4.4 DDR3 . . . 50
4.4.1 パッケージ形状の定義 . . . 50
4.4.2 制約の定義 . . . 55
4.4.3 電源設定 . . . 62
4.4.4 端子の交換 . . . 64
4.4.5 3 次元データの参照 . . . 65

第 5 章 LPB Format 構文チェッカー 69
5.1 LPB デザインキット Syntax Checker . . . 69
5.1.1 インストール . . . 69
5.1.2 アンインストール . . . 69
5.1.3 起動 . . . 70
5.1.4 画面の説明 . . . 70
5.1.5 CFORMAT の構文チェック . . . 70
5.2 GemCheck によるフォーマットチェック . . . 72
5.2.1 機能・特徴 . . . 72
5.2.2 ダウンロード・インストール . . . 72
5.2.3 パスの設定 . . . 72
5.2.4 アクティベート（ライセンス設定） . . . 72
5.2.5 使用方法 . . . 72

第 6 章 用語集・索引・検索 74
6.1 その他の用語 . . . 74
6.2 Todos . . . 74

第 1 章

はじめに

LPB Format は設計と検証に必要な情報を記述するための国際標準規格（IEC 63055/IEEE2401-2019）です。設計情報を交換するためのファイルフォーマットを統一することで、情報交換時の誤解を防ぎ、設計ツールの設定を自動化することを目的としています。

LPB Format は 5 種類のファイル (M-Format/N-Format/C-Format/R-Format/G-Format) で構成されています。図 1.1 、図 1.2 、図 1.3 にその中の M-Format,C-Format と R-Format の概要を示します。この概要に示すように LPB Format には多くの要素が含まれます。このため初心者には全体像をつかむことが難しくなっています。本書は初めて LPB Format に触れる方を対象として LPB Format の概要を理解してもらうことを目的としています。

本書は LPB Format の詳細を説明するものではありません。可能な限り実例を用いて最も基本的な部分を解説していきます。既に幾つかの部品ベンダーより LPB Format が提供されていますが、本書を読むことでそれらと同等のものを作成することが可能となります。本書が読者にとって有益なものとなり、LPB Format の提供や設計フロー・EDA ツールの開発の手助けになることを期待してます。

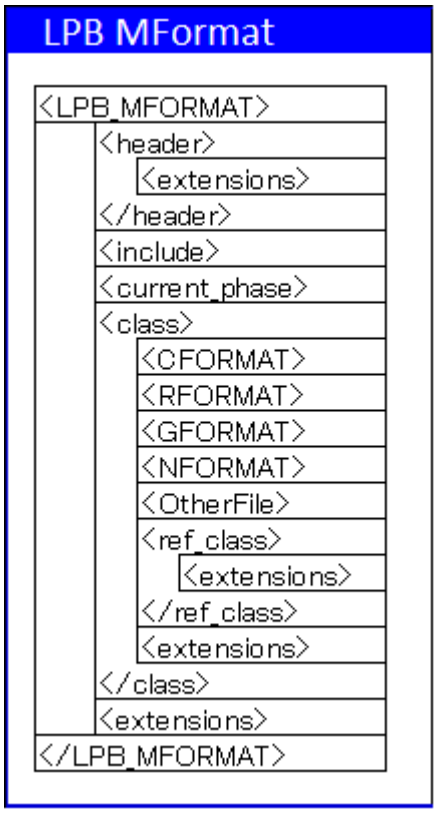


図 1.1　LPB M-Format 概要

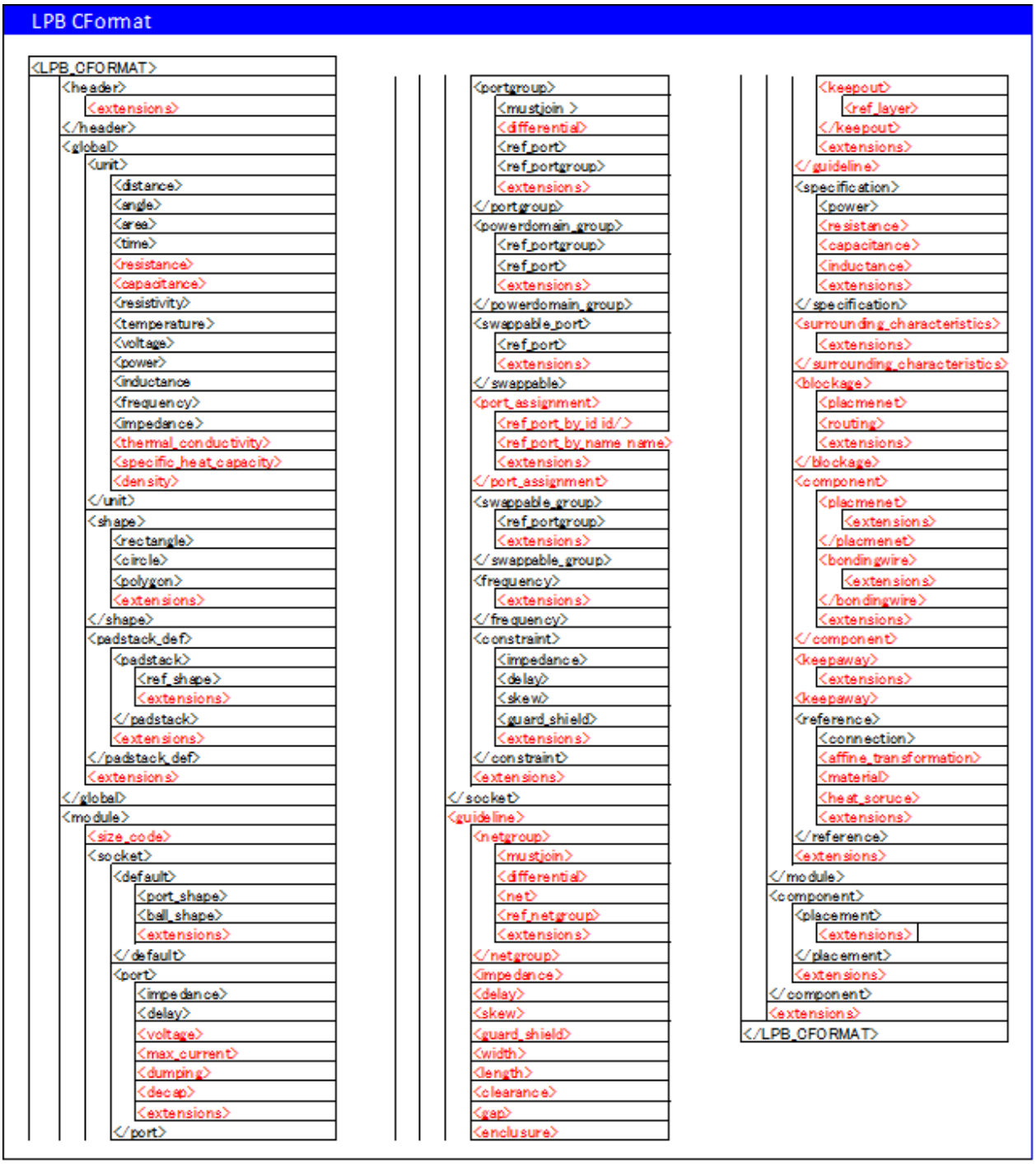


図 1.2　LPB C-Format 概要

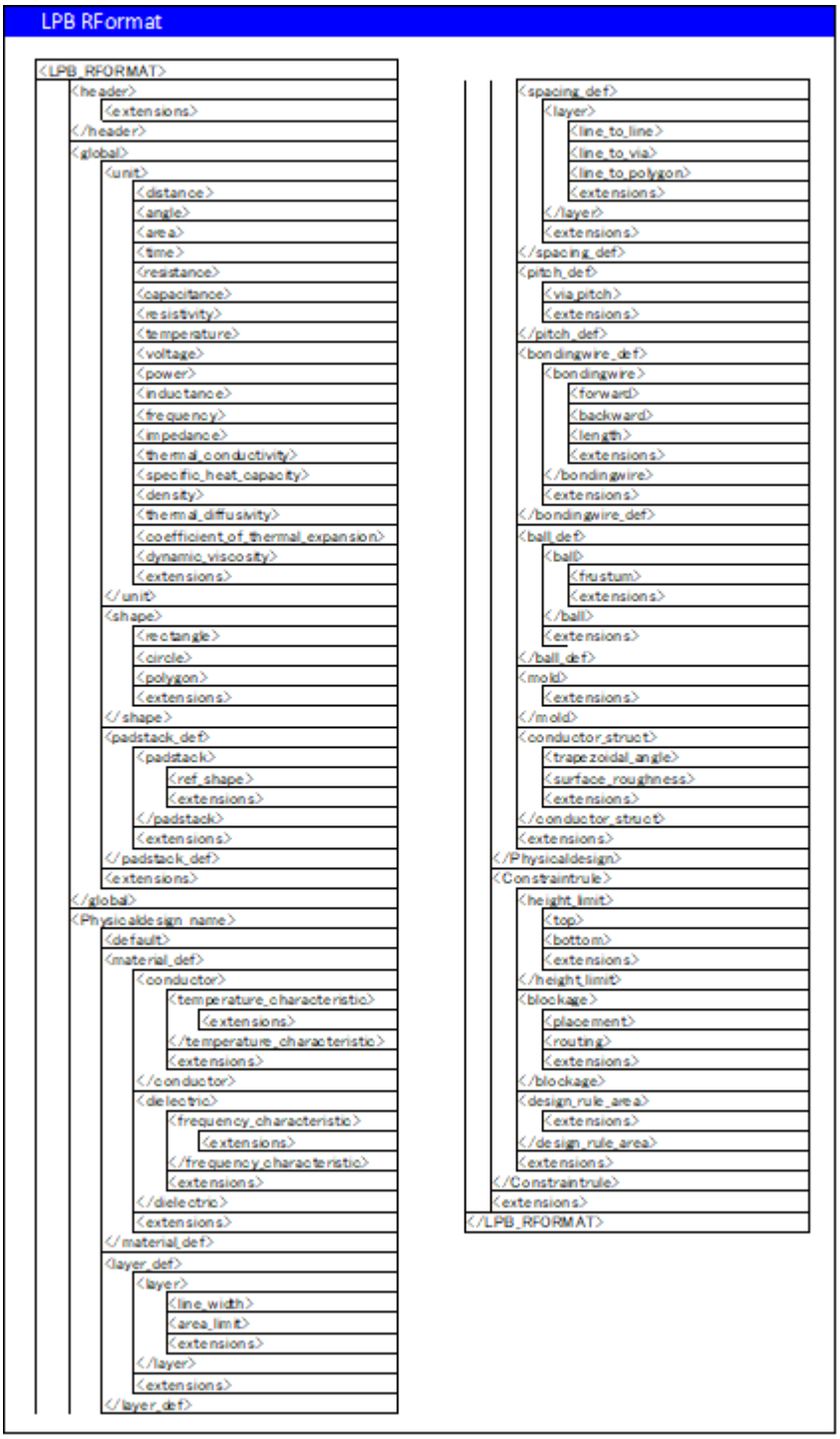


図 1.3　LPB R-Format 概要

第 2 章

LPB フォーマットとは何か？

2.1 LPB Format とは？

LPB Format は設計と検証に必要な情報を記述するための国際標準規格（IEC 63055/IEEE2401-2019）です。設計情報を交換するためのファイルフォーマットを統一することで、情報交換時の誤解を防ぎ、設計ツールの設定を自動化することを目的としています。LPB Format は以下の 5 つのファイルで構成されています。

- M-Format - プロジェクト管理用のファイルです。　LPB Format ファイルのバージョンや組み合わせを管理します。
- N-Format - 部品間の接続を定義するネットリストファイルです。
- C-Format - 設計制約や部品のフットプリント、シミュレーション用のモデルを　定義します。LPB Format の中核となるファイルです。
- R-Format - 基板構造や Line & Space 等のデザインルールを定義します。
- G-Format - プリント基板や IC パッケージなどの 2 次元の Layer Stackup 構造の　図形データを定義します。

図 2.1 は、LPB Format を使用して設計情報を交換する例を示しています。この例では LSI 設計、パッケージ設計、ボード設計の 3 種類の設計者が存在しています。すべての設計者は LPB Format で記載された設計情報を交換しています。

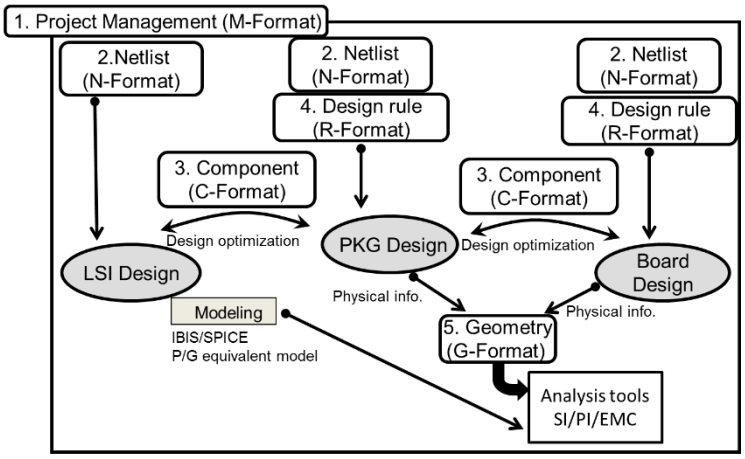


図 2.1　LPB フォーマットを使用した設計情報の交換例

2.2 LPB Format の構成

2.2.1 C-Format

C-Format の第 1 の目的は、LSI、パッケージ、ソケットなどのコンポーネントの外部仕様に統一された表記法を提供することです。外側仕様とは以下で定義するコンポーネントを使用するために必要な情報です。

- 部品の物理的形状
- 部品に対する入力および出力する信号の名前とタイプ
- 端子の物理的形状や位置、スワップ可能な端子定義などの入出力 (i/o) 仕様
- 遅延やスキューの上限などの設計上の制約
- 端子の入力インピーダンスや消費電力などの設計仕様

C-Format の 2 番目の目的は SPICE ネットリスト、IBIS、S パラメータなどのシミュレーションモデルの入出力ノードとコンポーネントの端子間の相互参照を提供することです。シミュレーションツールは C-Format ラップされたモデルファイルを入力することでモデルを自動的にプラグインできます。図 2.2 はシミュレーションモデルの交換例を示したものです。

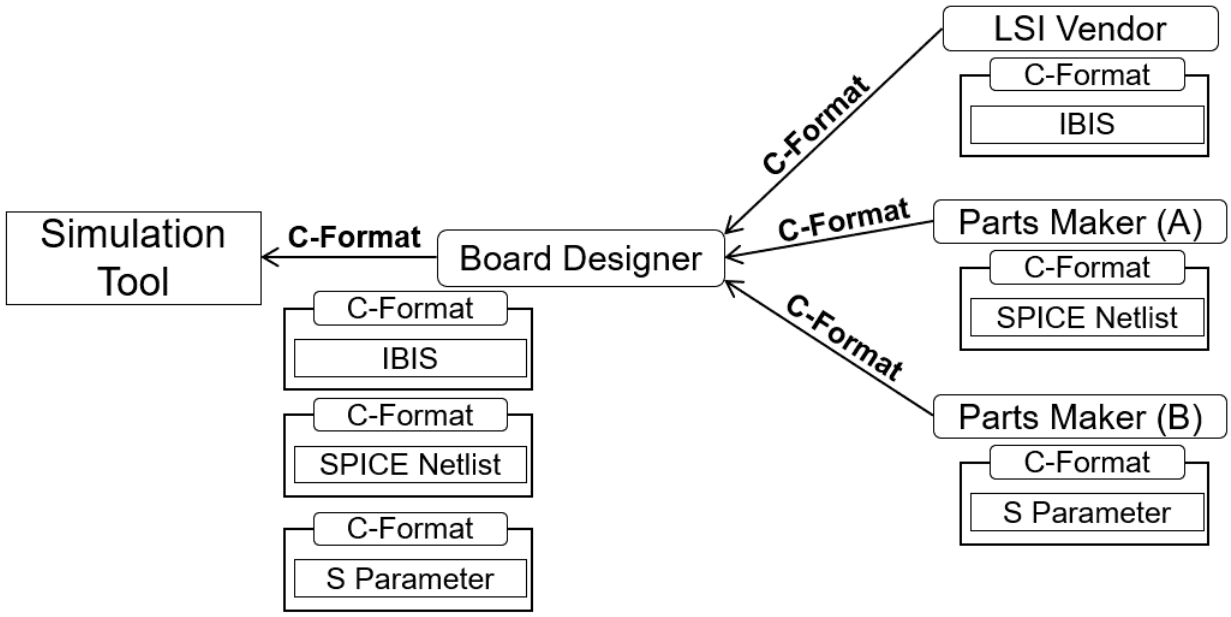


図 2.2　C-Format を使ったシミュレーションモデルの流通例

村田製作所は下記 URL でコンデンサ部品の C-Format と SPICE モデルおよび S パラメータファイルを公開しています。

https://www.murata.com/ja-jp/tool/c-format/mlcc

LPB デザインキットを使ってダウンロードした C-Format と S パラメータファイルを ANSYS/SI-Wave 用のライブラリに変換することができます。詳細は下記 URL を参照してください。

LPB デザインキット：　http://jeita-sdtc.com/lpb-open-source-project/lpbdesignkit/

LPB C-Froamt の使い方：　http://jeita-sdtc.com/wp/wp-content/uploads/2019/06/HowToUseCFormat1.2.pdf

2.2.2 R-Format

R-Format の第一の目的はプリント基板とパッケージの設計ルールの表記を統一することです。R-Format では次の設計規則が定義されています。

- パッケージとプリント基板の層スタックアップ
- 導電層/絶縁層の厚さ
- 各層に使用される材料
- 導電率、誘電定数、損失正接などの材料パラメータ
- 線幅と線のスペース

- ビア間隔とビア形状

通常、プリント基板やパッケージのメーカーは独自の表記法を使用して設計ルールを提供しています。設計者は異なる表記で説明されている設計規則を理解し設計ツールをセットアップする必要があります。これは誤解によるヒューマンエラーのリスクを含んでいます。R-Format で表記を統一することで設計ツールの自動的設定を可能とし、ヒューマンエラーを防ぎます。図 2.3 は RFomrat を使用して設計ルールを交換する例を示しています。すべての製造業者は統一された表記法（すなわち R-Format）を使用して設計規則を提供します。設計ツールは R-Format を入力することで自動的に設定されます。

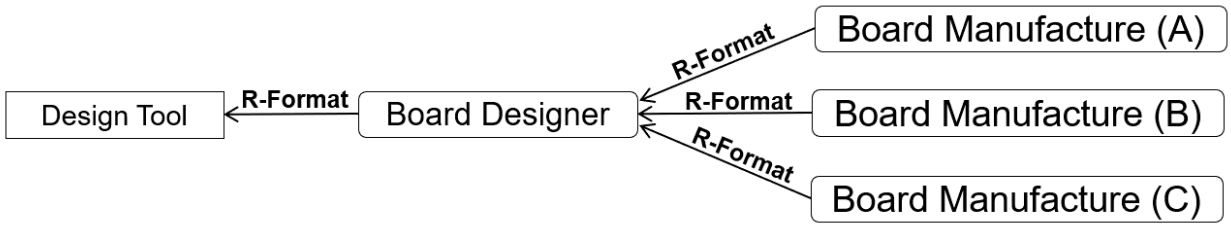


図 2.3　R-Format を使った設計ルールの流通例

R-Format の第 2 の目的はプリント基板とパッケージの物理的な設計上の制約を定義することです。物理的な設計上の制約とは取り付けられた部品の高さの制限とデフォルト以外の設計規則領域を意味します。

2.2.3 M-Format

LPB Format ファイルの内容は設計の進捗状況に応じて動的に更新されます。M-Format は LPB Format のファイルを交換するときのエラーを防ぐために、各ファイルのバージョンを管理することを目的としています。

2.2.4 G-Format

G-Format はプリント基板やパッケージなどのスタックアップ構造のレイアウトデータを表記するフォーマットです。標準化されたレイアウトデータを使用することで解析ツールとレイアウトツール間でデータをシームレスに交換することができます。G-Format ファイルには、次の物理情報が含まれています。

- プリント基板
- 材料の層スタックアップおよび物理パラメータ
- 配置された部品の形状と位置
- 端子の形状と位置
- ネットの経路またはパターン
- ビアの形状と位置

- ボンディングワイヤーの形状

2.2.5 N-Format

N-Format は設計に使用されるネットリストの表記を統一するを目的としています。従来はネットリストを表記するフォーマットがツールごとに異なるためデータの流通に問題が発生していました。N-Format は Verilog-HDL (IEEE Std 1364) に準拠し電源およびグランドを識別するためのキーワードを追加したものです。同じフォーマットのネットリストを使用することで回路設計ツールとレイアウトツール間でデータをシームレスに交換することが可能になります。

通常、回路設計者がボード設計者に設計を依頼する場合、ネットリストを提供します。ネットリストの表記が統一されていないと、その解釈でヒューマンエラーが発生する恐れがあります。統一された表記のネットリストを使用することで、ツール設定時の人為的ミスを防ぐことができます。

第 3 章

XML の概要

M-Format、C-Format、および R-Format は、[*1] XML を使用して定義されています。本節では、XML の簡単な構文規則について説明します。以下は、XML で記述された簡単な C-Format の例です。

```
<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
  <header project="LPB documentation project" design_revision="1.0"/>
  <global>
    <unit>
      <distance unit="mm"/>
    </unit>
    <shape>
      <rectangle id="1" height="0.3" width="0.6"/>
      <rectangle id="2" height="0.3" width="0.2"/>
    </shape>
    <padstack_def>
      <padstack id="1">
        <ref_shape pad_layer="BOTTOM" shape_id="2" x="0" y="0"/>
      </padstack>
    </padstack_def>
  </global>
  <module name="AMK063C6474_P-F"
          shape_id="1" thickness="0" type="C" x="0" y="0">
    <socket name="socket">
      <default>
        <port_shape padstack_id="1"/>
      </default>
      <port id="p1" x="0.2" y="0"/>
      <port id="p2" x="-0.2" y="0"/>
    </socket>
```

(次のページに続く)

[*1] XML（eXtensible Markup Language）は 1998 年 2 月に W3C で勧告が出された言語の仕様（XML1.0）です。XML の言語仕様は以下の URL から入手することができます。
https://www.w3.org/XML/

(前のページからの続き)

```
  </module>
</LPB_CFORMAT>
```

3.1 XML 宣言

XML ファイルはの以下の XML 宣言から始めます。

```
<?xml version="1.0" ?>
```

3.2 要素

XML ファイルは複数の要素（エレメントとも呼ばれる）で構成されます。以下に示すように要素の名前（要素名、タグとも呼ばれる）は三角括弧<> で囲みます。要素名は大文字と小文字を区別します。

```
<element>
```

3.2.1 要素の構文規則

各要素は、以下に示すように開始タグ (<element>) で始まり、終了タグ (</element>) で閉じます。

```
<element>....</element>
```

要素の終了タグは開始タグと同じ名前を付ける必要があります。下は終了タグと開始タグが異なるため誤ったタグの例です。

```
誤り ： <element1>....</element2>
```

3.2.2 要素の階層構造

要素は、その子として複数の要素を含むことができます。下の要素の階層構造の例です。

```
<?xml version = "1.0"?>
<element>
   <subelement1>
       <subelement2>...</subelement2>
       <subelement3>...</subelement3>
   </subelement1>
</element>
```

子の要素を持たない要素は、単純に以下の形に書くことができます。

```
<element/>
```

3.2.3 トップ要素

XML は、トップ要素を一つだけ持つことができます。

```
<?xml version = "1.0"?>
<top>
    <element> .... </element>
</top>
```

下のように複数のトップ要素を持つことはできません。

```
誤り : <?xml version = "1.0"?>
        <top>
            <element> .... </element>
        </top>
        <top>
            <element> .... </element>
        </top>
```

3.3 属性

属性（プロパティとも呼ばれる）は、以下に示すように属性名="属性値"で定義します。属性名は大文字と小文字を区別します。 属性値はダブルクオート (") で囲みます。下に示す例の name は属性名、propertyvalue は属性値です。

```
<element name="propertyvalue">
```

3.3.1 属性の構文規則

要素は 1 つ以上の属性を持つことができます。

```
<element a="value" b="value~>
```

2 つ以上の同じ属性名の属性を持つことはできません。次の例は属性 a が 2 回指定されているため誤った構文です。

誤り：`<element a="value" a="value">`

第 4 章

C-Format クイックスタート

C-Format は LSI、パッケージ、ソケットなどのコンポーネントの外部仕様を定義するものです。ここで外部仕様とは以下に示す情報です。

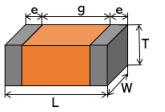
- 部品の物理的形状
- 部品に対する入力および出力する信号の名前とタイプ
- 端子の物理的形状や位置、スワップ可能な端子定義などの入出力 (i/o) 仕様
- 遅延やスキューの上限などの設計上の制約
- 端子の入力インピーダンスや消費電力などの設計仕様
- SPICE、IBIS、S パラメータなどのシミュレーションモデルの入出力ノードとコンポーネントの端子間の相互参照

C-Format には多くの情報を定義することが可能ですが、それらの大半は省略可能です。最も単純な C-Format は部品のフットプリントだけを定義したものです。本書では実存する部品を例として C-Format の最も基本的な構成を説明していきます。

なお、本書では実際に存在する部品を例にして **LPB CFormtat** の書式を説明しますが、例示する **LPB C-Format** は一般に公開されている情報を基に **JEITA SDTC** が独自で作成したものです。例示されている部品の提供会社は本文章の内容を保障するものではありません。本文章の内容に関するご質問は **JEITA SDTC** までお願いします。例示されている部品の提供元にお問い合わせることはご遠慮頂きますようお願いします。

4.1 コンデンサ

本節では村田製作所のチップコンデンサ（GRM188R60J226MEA0）を例にして最も簡単な C-Format を見ていきます。



L寸法	1.6mm
W寸法	0.8mm
T寸法	0.8mm
外部電極寸法e	0.2～0.55mm
外部電極間距離g	0.6mm
サイズコード	1608M

図 4.1　GRM188R60J226MEA0

下に GRM188R60J226MEA0 の C-Format の例を示します。この C-Format では部品の FootPrint と、その部品の SPICE モデルおよび S パラメータを定義しています。以後、この C-Format の詳細を見ていきます。

```
<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
  <header design_revision="1.0" project="GRM"
          company="MURATA" date="Friday Nov. 22 2019"/>
  <global>
    <unit>
      <distance unit="mm"/>
      <capacitance unit="uF"/>
    </unit>
    <shape>
      <rectangle id="1" height="0.8" width="1.6"/>
      <rectangle id="2" height="0.8" width="0.375"/>
    </shape>
    <padstack_def>
      <padstack id="1">
        <ref_shape shape_id="2" x="0" y="0" pad_layer="BOTTOM"/>
      </padstack>
    </padstack_def>
  </global>
  <module name="GRM188R60J226MEA0"
          shape_id="1" x="0" y="0" thickness="0.8" type="C">
    <size_code imperial="0603" metric="1608"/>
    <socket name="pins">
      <default>
        <port_shape padstack_id="1"/>
      </default>
      <port id="1" x="-0.6125" y="0.0"/>
      <port id="2" x="0.6125" y="0.0"/>
    </socket>
    <specification>
      <capacitance typ="22"/>
    </specification>
```

(次のページに続く)

(前のページからの続き)

```
    <reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
               format="SPICE"
               reffile="GRM188R60J226MEA0.mod">
      <connection socket_name="pins" port_id="1">
        <spice:ref_port subckt="GRM188R60J226MEA0" portid="1"/>
      </connection>
      <connection socket_name="pins" port_id="2">
        <spice:ref_port subckt="GRM188R60J226MEA0" portid="2"/>
      </connection>
    </reference>
    <reference xmlns:touchstone="http://www.jeita.or.jp/LPB/touchstone"
               format="TOUCHSTONE"
               reffile="GRM188R60J226MEA0.s2p">
      <connection socket_name="pins" port_id="1">
        <touchstone:ref_port portid="1"/>
      </connection>
      <connection socket_name="pins" port_id="2">
        <touchstone:ref_port portid="2"/>
      </connection>
    </reference>
  </module>
</LPB_CFORMAT>
```

4.1.1 C-Format の構成

C-Format のトップ要素は<LPB_CFORMAT>です。トップ要素の下に<header>, <global>, <module>の 3 つの要素を配置します。

<LPB_CFORMAT>要素の **version** 属性は、LPB Format のバージョンです。現在の最新バージョンは **2020** です。

```
<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
  <header>要素
  <global>要素
  <module>要素
</LPB_CFORMAT>
```

4.1.2 <header>要素

<header>要素には LPB Format ファイルの管理情報を定義します。**design_revision** と **project** は必須の属性です。その他の属性はオプションです。

```
<header design_revision="1.0" project="GRM"
        company="MURATA" date="Friday Nov. 22 2019"/>
```

project 必須属性です。プロジェクト名を記載します。任意の文字列を入力することができます。サンプルの C-Format ではコンデンサのシリーズ名 (GRM) を設定しています。

design_revision 必須属性です。リビジョン番号を設定します。任意の文字列が設定できます。サンプルでは"1.0"と設定していますが、"1.0.1"でも"1.0-beta1"でも問題ありません。管理しやすいリビジョン番号を設定してください。

company オプション属性です。C-Format のオーナを設定します。任意の文字列が設定可能です。

date オプション属性です。C-Format を作成した日付を設定ます。任意の文字列が設定可能です。サンプルでは"Wednesday Dec. 19 2018"としていますが、"2018/12/19(W)"でも問題ありません。管理しやすい書式で設定してください。

4.1.3 <global>要素

<global>要素には C-Format で使用する単位系 (<unit>要素)、形状 (<shape>要素) およびパッドスタック (<padstack_def>要素) を定義します。<global>要素のに設定された事項の有効範囲は、それが含まれるファイルに限られます。

```
<global>
  <unit>要素
  <shape>要素
  <padstack_def>要素
</global>
```

<unit>要素

<unit>要素には C-Format で使用する単位系を定義します。サンプルでは長さ・座標の単位系 (distance) と容量値の単位系 (capacitance) が定義されています。これ以外にもインダクタンス (inductance) や周波数 (frequency) 等の単位系も定義できます。C-Format 内で使用される全ての数値の単位系を、ここで定義してください。

```
<unit>
  <distance unit="mm"/>
  <capacitance unit="uF"/>
</unit>
```

<distance>要素

unit="mm"は、この C-Format 内での全ての長さ・座標を表す数値の単位はミリ・メートル (mm) であることを意味します。mm 以外に以下の設定が可能です。

pm　ピコ・メートル

nm　ナノ・メートル

um　マイクロ・メートル

mm　ミリ・メートル

m　メートル

inch　インチ

mil　ミル

<capacitance>要素

同様に unit="uF"は、この C-Format 内での全ての容量を表す数値の単位はマイクロ・ファラッド (uF) であることを意味します。 uF 以外に以下の設定が可能です。

fF　フェムト・ファラッド

pF　ピコ・ファラッド

nF　ナノ・ファラッド

uF　マイクロ・ファラッド

mF　ミリ・ファラッド

F　ファラッド

kF　キロ・ファラッド

MF　メガ・ファラッド

GF　ギガ・ファラッド

<shape>要素

<shape>要素には C-Format で使用されている図形を定義します。このサンプルではコンデンサ部品の外形と電極（端子）の形状 （共に矩形 (rectangle) です）を定義しています。

```
<shape>
    <rectangle id="1" height="0.8" width="1.6"/>
    <rectangle id="2" height="0.8" width="0.375"/>
</shape>
```

<rectangle> は矩形を定義する要素です。以下の必須属性を持ちます。

id

この shape の識別子です。識別子はユニークである必要があります。以下の例は id がユニークではないため誤った構文です。

```
誤り： <shape>
          <rectangle id="1" height="0.8" width="1.6"/>
          <rectangle id="1" height="0.8" width="0.375"/>
       </shape>
```

サンプルでは整数値を使用していますが、任意の文字を使用することができます。

```
<shape>
    <rectangle id="rect_1" height="0.8" width="1.6"/>
    <rectangle id="rect_2" height="0.8" width="0.375"/>
</shape>
```

height

矩形の高さです。長さの単位は<unit>要素の子要素である<distance>で定義されてます。このサンプルでは<distance unit="mm"/>と定義されているので、長さの単位はミリ・メートルです。

width

矩形の幅を定義します。

原点は矩形の中心点となります (`rectangle`)。

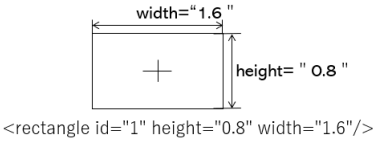


図 4.2　<rectangle>

<padstack_def>要素

<padstack_def>要素では C-Format で使用する端子、VIA、PAD 等のパッドスタックを定義します。その子要素として複数個の<padstack>要素を持つことができます。

このサンプルではコンデンサ部品の電極（端子）を定義しています。GRM188R60J226MEA0 は電極を 2 つ持ちますが、その形状は一種類だけなのでパッドスタックの定義も一つだけです。

```
<padstack_def>
  <padstack id="1">
    <ref_shape shape_id="2" x="0" y="0" pad_layer="BOTTOM"/>
  </padstack>
</padstack_def>
```

<padstack>要素

複数個の図形 (<shape>) を参照してつのパッドスタックを形成します。<ref_shape>要素で参照する図形 (<shape>) を定義します。

このサンプルの電極（端子）は単純な矩形で表現されているので、参照する図形 (<shape>) は一つだけです。

```
<padstack id="1">
  <ref_shape shape_id="2" x="0" y="0" pad_layer="BOTTOM"/>
</padstack>
```

id

必須属性です。このパッドスタックの識別子です。識別子はユニークである必要があります。サンプルでは整数値を使用していますが、任意の文字を使用することもできます。

<ref_shape>要素

参照する図形と、その配置座標を定義します。

```
<ref_shape shape_id="2" x="0" y="0" pad_layer="BOTTOM"/>
```

shape_id

必須属性です。参照する shape の識別子を記載します。このサンプルでは識別子が 2 の shape を参照しています。即ち幅 0.375mm、高さ 0.8mm の矩形を参照しています。

x y

オプション属性です。参照する shape の原点を配置する座標です。省略した場合は (0, 0) が配置座標となります。座標の単位は<unit>要素の子要素である<distance>で定義されてます。

pad_layer

参照する shape を配置する層（レイヤ）を定義します。部品の端子をはんだ付けで基板に接続する場合は **BOTTOM** を設定します。部品上面に端子を配置し、ワイヤボンディングで基板と接続する場合は **TOP** を設定します (`pad_layer`)。サンプルは表面実装部品 (SMD) なので BOTTOM を設定します。

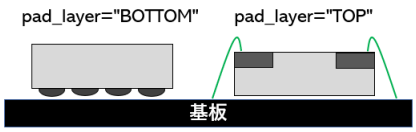


図 4.3　pad_layer

4.1.4 <module>要素

<module>要素は子要素として<size_code>要素、<socket>要素、<specification>要素および<referecen>要素を持ち、部品の FootPrint と、その部品のシミュレーションモデルが定義されたファイルとの参照関係を定義します。このサンプルでは SPICE モデルおよび S パラメータ (touch stone) との参照関係を定義しています。

```
<module name="GRM188R60J226MEA0"
        shape_id="1" x="0" y="0" thickness="0.8" type="C">
  <size_code>要素
  <socket>要素
  <specification>要素
  <reference>要素
</module>
```

<module>要素は以下の属性を持ちます。

name

必須属性です。モジュールの名前を定義します。

shape_id

このモジュールの外郭の形状を定義した<shape>の識別子を定義します。サンプルでは識別子 1 の shape を参照しています。即ち幅 1.6mm、高さ 0.8 の矩形が GRM188R60J226MEA0 の外郭形状です (`GRM188R60J226MEA0_bbox`)。

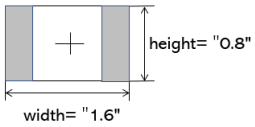


図 4.4　GRM188R60J226MEA0 の外郭形状 (TOP VIEW)

x y

shape_id で参照する shape の原点を配置する座標です。省略時は (0, 0) に配置されます。

thickness

部品の高さです。省略時は 0 となります。このサンプルでは部品の高さは 0.8 です (`GRM188R60J226MEA0_3Dbbox`)。

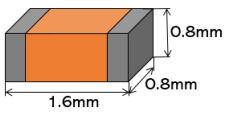


図 4.5　GRM188R60J226MEA0 の外郭形状 (3D)

type

部品のタイプを設定します。下記のいずれかの値を設定します。省略時は **OTHER** (不定）となります。

LSI　LSI

PKG　パッケージ

PWB　プリント基板

C　コンデンサ

R　抵抗

L　インダクター

OTHER　不定

サンプルではコンデンサを表す **C** を設定しています。

<size_code>要素

表面実装部品 (SMD) の<size_code>要素には部品のサイズコードを定義します。

```
<size_code imperial="0603" metric="1608"/>
```

imperial

インチベースのサイズコード、いわゆる EIA のサイズコードです。

metric

メートルベースのサイズコード、いわゆる JIS のサイズコードです。

<socket>要素

<socket>要素は子要素として<default>要素と<port>要素を持ち、部品の端子と、その形状 (FootPrint) を定義します。

socket は複数個の端子の集合です。このサンプルでは 2 つの端子を定義しています。

```
<socket name="pins">
  <default>
    <port_shape padstack_id="1"/>
  </default>
  <port id="1" x="-0.6125" y="0.0"/>
  <port id="2" x="0.6125" y="0.0"/>
</socket>
```

name

socket の名称です。socket 名は module 内でユニークである必要があります。module には複数の socket を定義することができます。socket を識別するために、個々の socket に名前を付けます。

特殊なケースを除き socket は一つだけ定義します。このサンプルでも socket は一つだけ定義しています。パッケージの上下に入出力端子を持つような POP 構造のパッケージでは、パッケージの上下に別々の socket を定義します。

<default>要素

<default>要素には socket に定義されている端子の形状を定義します。

```
<default>
  <port_shape padstack_id="1"/>
</default>
```

<port_shape>要素で端子の形状を定義した padstack を参照します。この padstack が端子のデフォルトの形状となります。

padstack_id

端子形状を定義した padstack の識別子を記載します。このサンプルでは下記の識別子 1 の padstack を参照しています。

```
<padstack id="1">
  <ref_shape shape_id="2" x="0" y="0" pad_layer="BOTTOM"/>
</padstack>
```

さらに、この padstack は下記の識別子 2 の矩形を参照しています。

```
<rectangle id="2" height="0.8" width="0.375"/>
```

従って端子形状は **BOTTOM** に配置された幅 0.375mm、高さ 0.8 の矩形となります。

<port>要素

<port>要素で端子の座標を定義します。このサンプルでは 2 つの端子を定義しています。それぞれの端子の形状は<default>要素で定義されています。

```
<port id="1" x="-0.6125" y="0.0"/>
<port id="2" x="0.6125" y="0.0"/>
```

id

端子の識別子です。同一 socket 内で識別子はユニークである必要があります。このサンプルでは整数値を識別してして使用していますが任意の文字列も使えます（下例参照）。

```
<port id="G8" x="2.5" y="-1.5"/>
<port id="G9" x="3.5" y="-1.5"/>
<port id="G10" x="4.5" y="-1.5"/>
```

x y

端子の配置座標です。端子形状の padstack の原点を、ここで指定された座標に配置します。サンプルの端子座標は (-0.6125, 0.0) と (0.6125, 0.0) となります (`GRM188R60J226MEA0_ports`)。

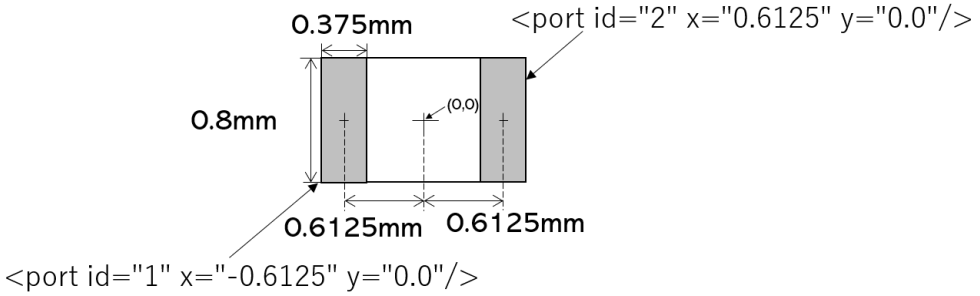


図 4.6　GRM188R60J226MEA0 の端子

<specification>要素

<specification>要素は部品の特性を定義します。サンプルの例ではコンデンサ部品の容量の公証値を定義しています。

```
<specification>
  <capacitance typ="22"/>
</specification>
```

子要素の<capacitance>要素が容量値です。

typ

容量値です。容量の単位は<unit>要素の子要素である<capacitance>で定義されてます。サンプルの例では単位は <capacitance unit="uF"/> と定義しているので、22uF が容量値となります。

<specification>要素はオプショナルな要素です。未定義でも問題ありません。

<reference>要素

<reference>要素で、この部品のシミュレーションモデルと端子との参照関係を定義します。サンプルは SPICE モデルの定義用と S パラモデルの定義用の 2 つの<reference>要素を含んでいます。

<reference>要素の構文規則

<reference>要素は子要素として一つ以上の <connection>要素を含みます。

```
<reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
           format="SPICE"
           reffile="GRM188R60J226MEA0.mod">
  <connection>要素
</reference>
```

<reference>の属性で参照するシミュレーションモデルの種類とファイルを定義します。以下の 3 つの属性を持ちます。

xmlns format

参照するシミュレーションモデルにより **xmlns** と **format** の記述方法は以下のように固定されています。

SPICE モデル

```
xmlns:spice="http://www.jeita.or.jp/LPB/spice"
format="SPICE"
```

S パラメータ（touch stone 形式)

```
xmlns:touchstone="http://www.jeita.or.jp/LPB/touchstone"
format="TOUCHSTONE"
```

reffile

シミュレーションモデルのファイル名です。参照するファイル名を記述してください。

<connection>の属性でシミュレーションモデルの入出力ノードと端子とのクロスリファレンスを定義します。**socket_name port_id** の 2 つの属性を持ちます（下例参照)。

```
<connection socket_name="pins" port_id="1">
```

socket_name

シミュレーションモデルの入出力ノードに対応する部品の端子が定義されている socket の名前（識別子）を記載します。

port_id

シミュレーションモデルの入出力ノードに対応する部品の端子が定義されている端子 (port) の識別子を記載します。ここで参照されている識別子の端子 (port) は、 **socket_name** で指定された socket に属している必要があります。

<connection>要素の子要素は参照するシミュレーションモデルにより子要素が変化します。詳細は以降の節で説明します。

SIPCE モデル用の<reference>要素

以下に 2 端子部品の SPICE モデルファイルを参照する例を示します。SPICE モデルの場合の **xmlns** と **format** は下記の例のように記述します。**reffile** は参照する SPICE モデルのファイル名です。

```
<reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
           format="SPICE"
           reffile="GRM188R60J226MEA0.mod">
  <connection socket_name="pins" port_id="1">
    <spice:ref_port subckt="GRM188R60J226MEA0" portid="1"/>
  </connection>
  <connection socket_name="pins" port_id="2">
    <spice:ref_port subckt="GRM188R60J226MEA0" portid="2"/>
  </connection>
</reference>
```

<connection>要素は、SPICE の I/O ノードと端子との対応（クロスリファレンス）を定義します。SPICE モデルを参照する場合の子要素は<spice:ref_port>です。下記にその例を示します。

```
<connection socket_name="pins" port_id="1">
  <spice:ref_port subckt="GRM188R60J226MEA0" portid="1"/>
</connection>
```

subckt

<reference>要素の **reffile** で指定された SPICD モデルファイルに記載されている subckt 名です。

portid

SPICE モデルの I/O ノードの定義順番です。上記例では portid="1" と記載されているので、第一番目に定義されている I/O ノードを意味します。

上記の<connection>要素の例では、SPICE モデルファイルの GRM188R60J226MEA0 という subckt の第一番目の I/O ノードは、CFormat 中の pins という名前の socket に含まれる id が 1 の port とリンクします。

再度、サンプルの CFormat を見てみましょう。`SPICE_reference` は CFormat の port と SPICE モデルの I/O ノードの関係を示しています。GRM188R60J226MEA0 の id="1"の端子は SPICE モデルの Port1 ノードに、id="2" の端子は Port2 ノードにリンクしています (`SPICE_reference`)。

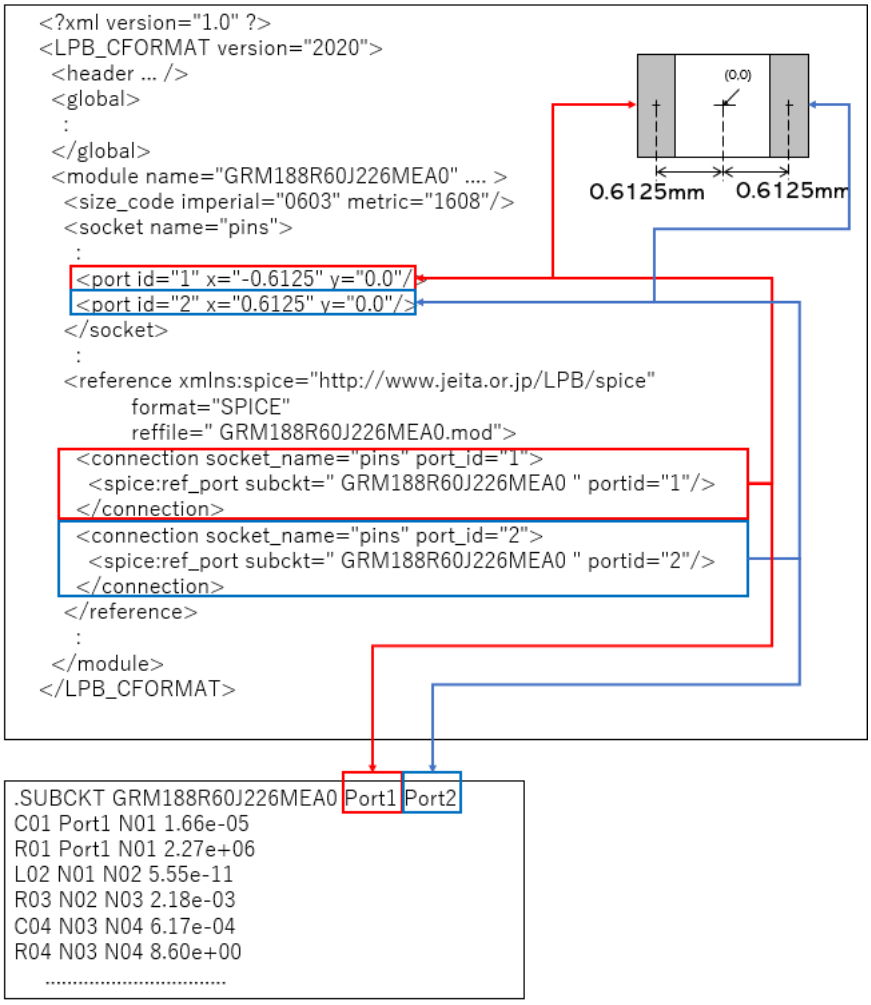


図 4.7　SIPCE モデル用の<reference>要素の例

S パラ用の<reference>要素

以下に S パラメータファイルを参照する例を示します。S パラメータの場合の **xmlns** と **format** は下記の例のように記述します。**reffile** は参照する S パラメータのファイル名です。

```
<reference xmlns:touchstone="http://www.jeita.or.jp/LPB/touchstone"
           format="TOUCHSTONE"
           reffile="GRM188R60J226MEA0.s2p">
  <connection socket_name="pins" port_id="1">
    <touchstone:ref_port portid="1"/>
  </connection>
  <connection socket_name="pins" port_id="2">
    <touchstone:ref_port portid="2"/>
  </connection>
</reference>
```

<connection>要素は、S パラメターの I/O ノードと端子との対応（クロスリファレンス）を定義します。S パラメータを参照する場合の、子要素は<touchstone:ref_port>です。下記に、その例を示します。

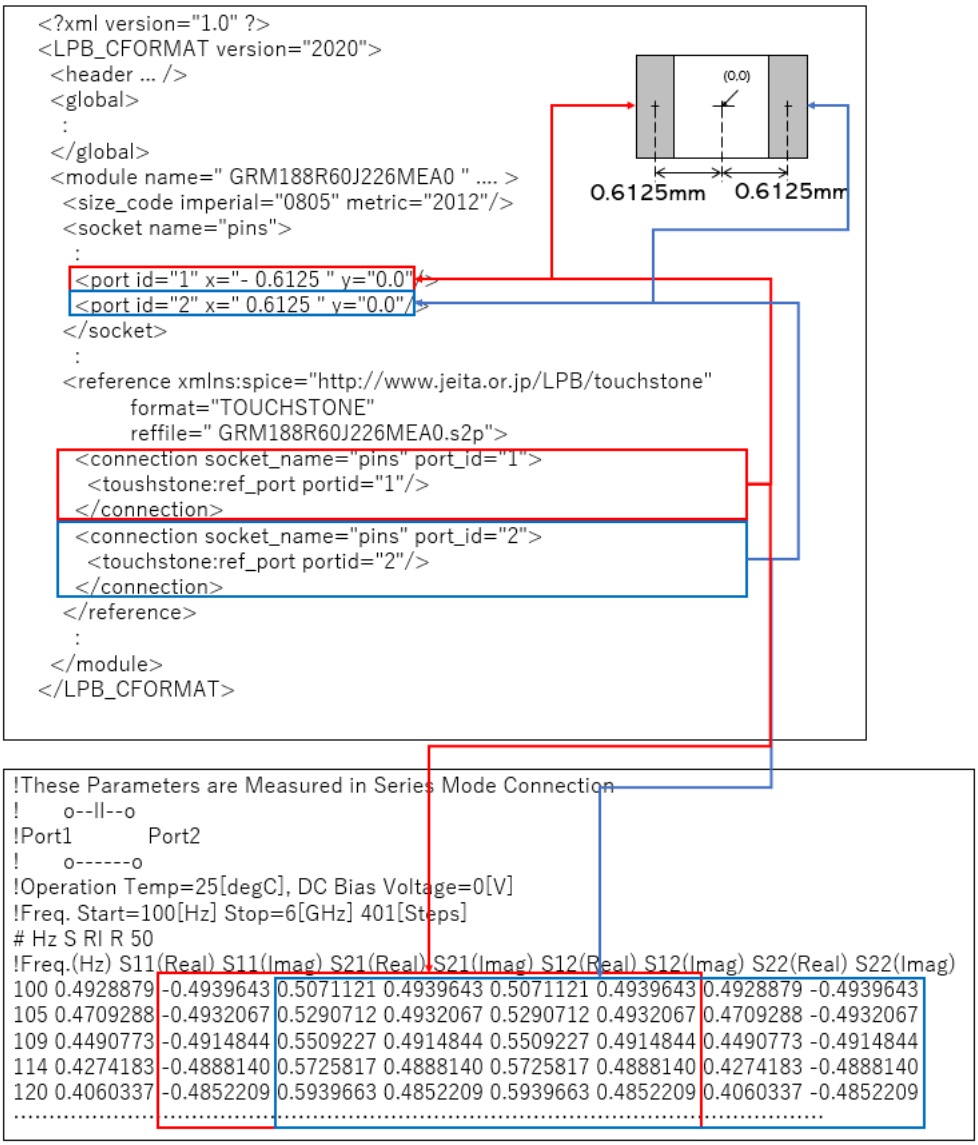
```
<connection socket_name="pins" port_id="1">
  <touchstone:ref_port portid="1"/>
</connection>
```

portid

S パラメータのポート番号です。上記例では portid="1" と記載されているので、S パラメータの第一ポートを意味します。

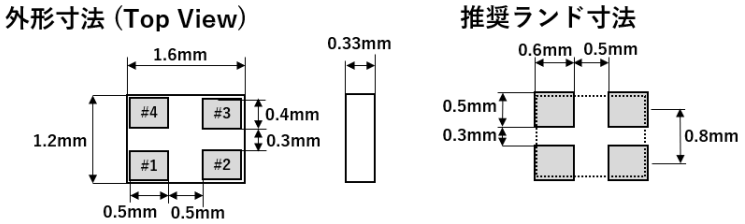
上記の<connection>要素の例では、S パラメータの第一ポートは CFormat 中の pins という名前の socket に含まれる id が 1 の port とリンクします。

`Spara_reference` は CFormat の port と S パラメータの I/O ノードの関係を示しています。GRM188R60J226MEA0 の id="1"の端子は S パラメータの第一ポートに、id="2"の端子は第 2 ポートにリンクしています。


<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
<header ... />
<global>
:
</global>
<module name=" GRM188R60J226MEA0 " .... >
<size_code imperial="0805" metric="2012"/>
<socket name="pins">
:
<port id="1" x="- 0.6125 " y="0.0"/>
<port id="2" x=" 0.6125 " y="0.0"/>
</socket>
:
<reference xmlns:spice="http://www.jeita.or.jp/LPB/touchstone"
format="TOUCHSTONE"
reffile=" GRM188R60J226MEA0.s2p">
<connection socket_name="pins" port_id="1">
<toushstone:ref_port portid="1"/>
</connection>
<connection socket_name="pins" port_id="2">
<touchstone:ref_port portid="2"/>
</connection>
</reference>
:
</module>
</LPB_CFORMAT>
(0,0)
0.6125mm
0.6125mm
!These Parameters are Measured in Series Mode Connection
! o--II--o
!Port1 Port2
! o------o
!Operation Temp=25[degC], DC Bias Voltage=0[V]
!Freq. Start=100[Hz] Stop=6[GHz] 401[Steps]
Hz S RI R 50
!Freq.(Hz) S11(Real) S11(Imag) S21(Real) S21(Imag) S12(Real) S12(Imag) S22(Real) S22(Imag)
100 0.4928879 -0.4939643 0.5071121 0.4939643 0.5071121 0.4939643 0.4928879 -0.4939643
105 0.4709288 -0.4932067 0.5290712 0.4932067 0.5290712 0.4932067 0.4709288 -0.4932067
109 0.4490773 -0.4914844 0.5509227 0.4914844 0.5509227 0.4914844 0.4490773 -0.4914844
114 0.4274183 -0.4888140 0.5725817 0.4888140 0.5725817 0.4888140 0.4274183 -0.4888140
120 0.4060337 -0.4852209 0.5939663 0.4852209 0.5939663 0.4852209 0.4060337 -0.4852209

4.2 水晶振動子

本節ではリバーエレテックの水晶振動子（FCX-07L）を例にして C-Format を見ていきます。前節のチップコンデンサと異なりこの水晶振動子は 4 つの端子を持ちます。また 4 つの端子のうち 2 つだけが水晶と接続し、他はグランドと no-connection です。シミュレーションモデルの参照も異なってきます。



内部接続

Pin	Connection
#1	Crystal
#2	Ground
#3	Crystal
#4	No Connection

図 4.8　FCX-07L

下に FCX-07L の C-Format の例を示します。以後、この C-Format の詳細を見ていきます。

前節の 2 端子表面実装部品と同様に、この C-Format もトップ要素の下に<header>, <global>, <module>の 3 つの要素を配置しています。

```
<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
  <header company="RIVERELETEC" date="Monday Sep. 16 2018"
          design_revision="1.0" project="FCX"/>
  <global>
    <unit>
      <distance unit="mm"/>
    </unit>
  <shape>
    <rectangle id="1" height="0.5" width="0.6"/>
    <rectangle id="2" height="1.2" width="1.6"/>
  </shape>
  <padstack_def>
    <padstack id="1">
      <ref_shape shape_id="1" x="0" y="0" pad_layer="BOTTOM"/>
    </padstack>
  </padstack_def>
</global>
  <module name="FCX07L" type="OTHER"
          shape_id="2" thickness="0.33" x="0" y="0">
```

(次のページに続く)

(前のページからの続き)

```
    <socket name="socket">
      <default>
        <port_shape padstack_id="1"/>
      </default>
      <port id="1" x="-0.55" y="-0.4"/>
      <port id="2" x="0.55" y="-0.4" type="ground"/>
      <port id="3" x="0.55" y="0.4"/>
      <port id="4" x="-0.55" y="0.4" type="dontcare"/>
    </socket>
  <reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
             reffile = "FCX07L.mod"
             format="SPICE" >
    <connection socket_name="socket" port_id="1">
      <spice:ref_port subckt="FCX07L" portid="1"/>
    </connection>
    <connection socket_name="socket" port_id="3">
        <spice:ref_port subckt="FCX07L" portid="2"/>
    </connection>
  </reference>
  </module>
</LPB_CFORMAT>
```

4.2.1 <header>要素

<header>要素には前節の 2 端子表面実装部品と同様に **project** 、**design_revision** 、**company** 、**date** の 4 つの属性を定義しています。**design_revision** と **project** 以外はオプション属性なので未定義でも構いませんが C-Format の管理上、定義した方が良いでしょう。

```
<header company="RIVERELETEC" date="Monday Sep. 16 2018"
        design_revision="1.0" project="FCX"/>
```

4.2.2 <global>要素

<global>要素には前節の 2 端子表面実装部品と同様に、<unit>要素、<shape>要素、<padstack_def>要素が定義されています。

<unit>要素

このサンプルの C-Format では長さ・座標以外の数値は使われていないため、長さ・座標の単位 (ミリメートル) だけが定義されています。

```
<unit>
  <distance unit="mm"/>
</unit>
```

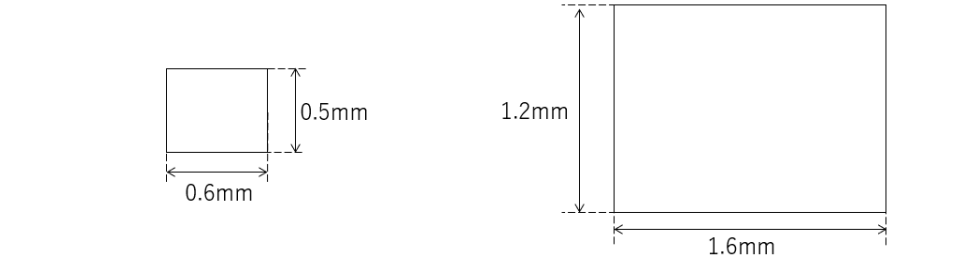
<unit>要素には C-Format 内で使われている単位系を定義します。ここで取り上げている例では長さ単位 (<distance>) だけを定義していますが、下記の単位が定義可能です。

```
<unit>
  <distance unit="um" />
  <angle unit="degree" />
  <area unit="um2" />
  <time unit="ps" />
  <resistance unit="mohm" />
  <capacitance unit="pF" />
  <resistivity unit="ohmm"/>
  <temperature unit="C" />
  <voltage unit="V" />
  <current unit="mA" />
  <power unit="mW" />
  <inductance unit="nH" />
  <frequency unit="MHz" />
  <impedance unit="ohm" />
  <thermal_conductivity unit="W/(m*K)" />
  <specific_heat_capacity unit="J/(kg*K)" />
  <density unit="kg/m3" />
  <thermal_diffusivity unit="m2/s" />
  <coefficient_of_thermal_expansion unit="1/K" />
  <dynamic_viscosity unit="Pas" />
</unit>
```

<shape>要素

水晶振動子の外形と電極（端子）の形状として 2 つの矩形 (rectangle) を定義しています。

```
<shape>
  <rectangle id="1" height="0.5" width="0.6"/>
  <rectangle id="2" height="1.2" width="1.6"/>
</shape>
```



<rectangle id="1" height="0.5" width="0.6"/> <rectangle id="2" height="1.2" width="1.6"/>

図 4.9 <rectangle>

<padstack_def>要素

<padstack_def>要素には前節の 2 端子表面実装部品と同様に部品の電極（端子）を定義しています。FCX-07L は電極を 4 つ持ちますが形状は一種類だけなのでパッドスタックの定義も一つだけです。

```
<padstack_def>
    <padstack id="1">
      <ref_shape shape_id="1" x="0" y="0" pad_layer="BOTTOM"/>
    </padstack>
</padstack_def>
```

4.2.3 <module>要素

<module>要素には<socket>要素と<referecen>要素を定義しています。前節の 2 端子表面実装部品と異なり FCX-07L は汎用的なサイズコードを持たなため、<size_code>は定義していません。また<specification>も定義していません。

```
<module name="FCX07L" type="OTHER"
            shape_id="2" thickness="0.33" x="0" y="0">
  <socket>要素
  <reference>要素
</module>
```

shape_id

識別子 2 の shape を参照しています。即ち幅 1.6mm、高さ 1.2 の矩形が FCX-07L の外郭形状です。

thickness

部品の高さとして 0.33mm を定義しています。

type

水晶振動子は C-Format で定義した部品タイプには含まれていなので、**OTHER** (その他のタイプ）を定義しています。**OTHER** の場合は **type** 属性は省略可能です。

<socket>要素

FCX-07L は電極（端子）を 4 つ持つので<socket>要素には 4 つの<port>要素を定義しています。

```
<socket name="socket">
  <default>
    <port_shape padstack_id="1"/>
  </default>
  <port id="1" x="-0.55" y="-0.4"/>
  <port id="2" x="0.55" y="-0.4" type="ground"/>
  <port id="3" x="0.55" y="0.4"/>
  <port id="4" x="-0.55" y="0.4" type="dontcare"/>
</socket>
```

<default>要素

全ての電極（端子）の形状として id="1"の padstack を定義しています。

```
<default>
  <port_shape padstack_id="1"/>
</default>
```

id="1"の padstack は以下のように定義されています。

```
<padstack id="1">
  <ref_shape shape_id="1" x="0" y="0" pad_layer="BOTTOM"/>
</padstack>
```

この padstack は下記の id="1"の矩形を参照しています。従って端子形状は **BOTTOM** に配置された幅 0.6mm、高さ 0.5mm の矩形となります。

<port>要素

このサンプルでは 4 つの端子を定義しています。

```
<port id="1" x="-0.55" y="-0.4"/>
<port id="2" x="0.55" y="-0.4" type="ground"/>
<port id="3" x="0.55" y="0.4"/>
<port id="4" x="-0.55" y="0.4" type="dontcare"/>
```

それぞれの端子は (-0.55mm, -0.4mm)、(0.55mm, -0.4mm)、(0.55mm, 0.4mm)、(-0.55mm, 0.4mm) の座標に配置されています。それぞれの端子の形状は<default>要素で定義されている幅 0.6mm、高さ 0.5mm の矩形です (図 4.10)。

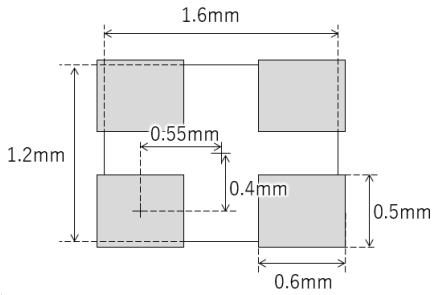


図 4.10　FCX-07L の C-Format の形状

図 4.10 を見るとわかる通り FCX-07L の C-Format は推奨ランドの形状で定義されています。

type

type 属性で、その端子に入出力する信号種別を定義します。オプション属性です。この例では端子 2 に **ground** 、端子 4 に **dontcare** を設定しています。以下の 8 種類のタイプがあります。

power　電源

ground　グランド

signal　一般信号

floating　如何なるネットにも接続してはならない端子です。この端子は未接続のままにしておくことを強制します。

dontcare　サーマルボールや non-connection のように論理的には意味を持たない端子

through　いわゆるフィードスルー端子と呼ばれる端子。フィードスルー端子は如何なる内部回路とも接続せず単にパッケージを貫通しているだけです

thermal　熱端子です。VHDL-AMS や SPICE などのシミュレーションモデルの熱端子と接続する場合に使用します。この端子に接続する熱（温度）の単位は K(Kelvin) です

thermal_c　thermal と同様な熱端子です。ただし、この端子に接続する熱（温度）の単位は ℃ (Celsius) です

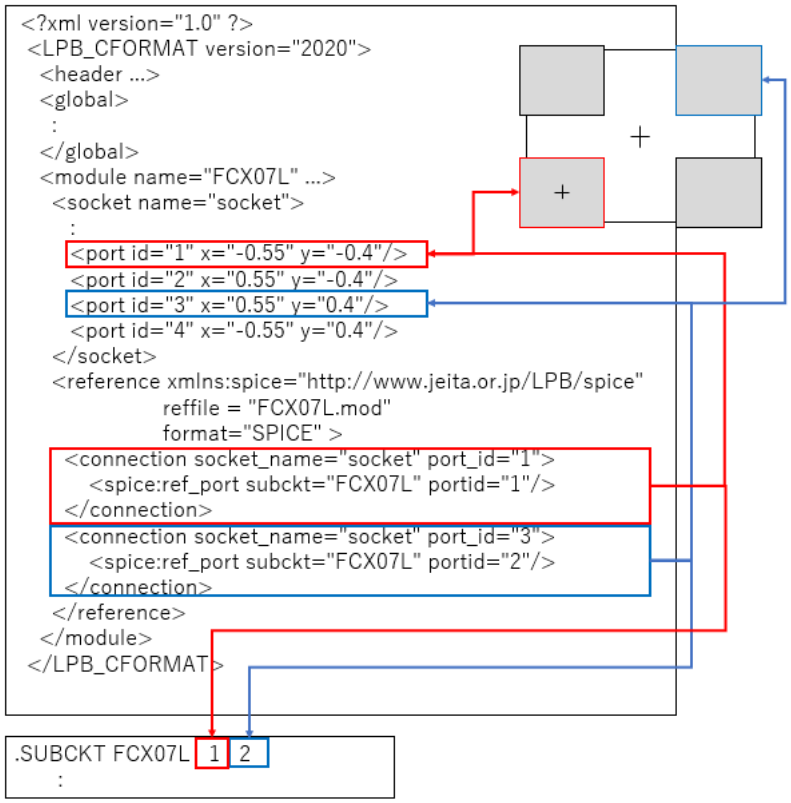
<reference>要素

サンプルでは SPICE モデルを参照する<reference>要素を定義しています。

FCX-07L は 4 つの電極（端子）を持ちますが、id="1"と id="3"の 2 つの端子だけが水晶とつながっています。SPICE モデルにつながる端子もこの 2 つだけです。従って、下記に示すように 2 つの<connection>要素で SPICE モデルの関連を定義しています。

```
<reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
           reffile = "FCX07L.mod"
           format="SPICE" >
  <connection socket_name="socket" port_id="1">
    <spice:ref_port subckt="FCX07L" portid="1"/>
  </connection>
  <connection socket_name="socket" port_id="3">
    <spice:ref_port subckt="FCX07L" portid="2"/>
  </connection>
</reference>
```

FCX07L_spice_reference は C-Format の port と SPICE モデルの I/O ノードの関係を示しています。



4.3 MOSFET

本節では東芝の MOS-FET（TPHR7904PB）を例にして C-Format を見ていきます。前節までのコンデンサや水晶振動子とことなり、複雑な形状の端子を持っています。この部品のパッケージ種別は SOP となります。その形状を下図に示します。

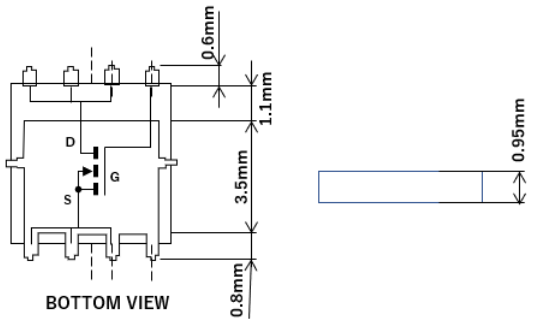


図 4.11　TPHR7904PB

下に TPHR7904PB の C-Format の例を示します。以後、この C-Format の詳細を見ていきます。

他の例と同様に、この C-Format もトップ要素の下に<header>, <global>, <module>の 3 つの要素を配置しています。

```
<?xml version="1.0"?>
<LPB_CFORMAT version="2020" >
<header company="TOSHIBA" project="Power MOSFET"
         date="20190621" design_revision ="1.0"/>
<global>
  <unit>
    <distance unit="um" />
    <angle    unit="degree" />
  </unit>
  <shape>
    <polygon id="COMP_SHAPE_SOP_Advance_WF"
              points="-2305,-3375,-2305,-2600,-2600,-2600,-2600,-250,-2750,-250,
                      -2750,250,-2600,250,-2600,2600,-2305,2600,-2305,3375,
                      2305,3375,2305,2600,2600,2600,2600,250,2750,250,2750,
                      -250,2600,-250,2600,-2600,2305,-2600,2305,-3375,-2305,-3375" />
    <polygon id="PAD_SHAPE_r80000x125000"
             points="325,625,362,615,390,587,400,550,400,-550,390,-588,362,
                     -615,325,-625,-325,-625,-363,-615, -390,-588,-400,-550,
                     -400,550,-390,587,-363,615,-325,625,325,625" />
    <polygon id="PAD_SHAPE_r_sop_advance_wf"
             points="-2150,-1400,-2187,-1390,-2215,-1362,-2225,-1325,-2225,2225,
                     -2235,2263,-2262,2290,-2300,2300,-2230,2300,-2267,2310,-2295,
                     2338,-2305,2375,-2305,3300,-2295,3338,-2267,3365,-2230,3375,
```

(次のページに続く)

(前のページからの続き)

```
                    -1580,3375,-1542,3365,-1515,3338,-1505,3300,-1505,2375,-1495,
                    2338,-1467,2310,-1430,2300,-1110,2300,-1072,2310,-1045,2338,
                    -1035,2375,-1035,3300,-1025,3338,-997,3365,-960,3375,-310,3375,
                    -272,3365,-245,3338,-235,3300,-235,2375,-225,2338,-197,2310,
                    -160,2300,160,2300,198,2310,225,2338,235,2375,235,3300,245,
                    3338,273,3365,310,3375,960,3375,998,3365,1025,3338,1035,3300,
                    1035,2375,1045,2338,1073,2310,1110,2300,1430,2300,1468,2310,
                    1495,2338,1505,2375,1505,3300,1515,3338,1543,3365,1580,3375,
                    2230,3375,2268,3365,2295,3338,2305,3300,2305,2375,2295,2338,
                    2268,2310,2230,2300,2300,2300,2263,2290,2235,2263,2225,2225,2225,
                    -1325,2215,-1362,2188,-1390,2150,-1400,-2150,-1400" />
  </shape>
  <padstack_def>
    <padstack id="R80000X125000F">
      <ref_shape shape_id="PAD_SHAPE_r80000x125000" type="Land" x="0" y="0"
                 angle="0" pad_layer="BOTTOM" />
    </padstack>
    <padstack id="R_SOP_ADVANCE_WF">
      <ref_shape shape_id="PAD_SHAPE_r_sop_advance_wf" type="Land" x="0" y="0"
                 angle="0" pad_layer="BOTTOM" />
    </padstack>
  </padstack_def>
</global>
<module name="TPHR7904PB" type="PKG" shape_id="COMP_SHAPE_SOP_Advance_WF"
        x="0" y="0" angle="0" thickness="0.95" >
  <socket  name="SOP_Advance_WF" >
    <port id="1" padstack_id="R80000X125000F" x="-1905" y="-2750"  angle="180"
          name="D" />
    <port id="2" padstack_id="R80000X125000F" x="-635" y="-2750"  angle="180"
          name="D" />
    <port id="3" padstack_id="R80000X125000F" x="635" y="-2750"  angle="0"
          name="D" />
    <port id="4" padstack_id="R80000X125000F" x="1905" y="-2750"  angle="0"
          name="G"/>
    <port id="5" padstack_id="R_SOP_ADVANCE_WF" x="0" y="0"  angle="0"
          name="S" />
    <portgroup name="drain">
      <mustjoin/>
      <ref_port id="1"/>
      <ref_port id="2"/>
      <ref_port id="3"/>
    </portgroup>
  </socket>
  <reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
             reffile = "TPHR7904PB.lib" format="SPICE" >
    <connection socket_name="SOP_Advance_WF" port_id="1">
          <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
    </connection>
```

(次のページに続く)

(前のページからの続き)

```
    <connection socket_name="SOP_Advance_WF" port_id="2">
          <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
    </connection>
    <connection socket_name="SOP_Advance_WF" port_id="3">
          <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
    </connection>
    <connection socket_name="SOP_Advance_WF" port_id="4">
          <spice:ref_port subckt="NMOS_TPHR7904PB" portid="2"/>
    </connection>
    <connection socket_name="SOP_Advance_WF" port_id="5">
          <spice:ref_port subckt="NMOS_TPHR7904PB" portid="5"/>
    </connection>
  </reference>
</module>
</LPB_CFORMAT>
```

4.3.1 C-Format の構成

C-Format のトップ要素は<LPB_CFORMAT>です。トップ要素の下に<header>, <global>, <module>の 3 つの要素を配置します。

```
<?xml version="1.0" ?>
<LPB_CFORMAT version="2020">
  <header>要素
  <global>要素
  <module>要素
</LPB_CFORMAT>
```

4.3.2 <header>要素

<header>要素には **project** 、**design_revision** 、**company** 、**date** の 4 つの属性を定義しています。

```
<header company="TOSHIBA" project="Power MOSFET"
        date="20190621" design_revision ="1.0"/>
```

4.3.3 <global>要素

<global>要素には他のサンプルと同様に、<unit>要素、<shape>要素、<padstack_def>要素が定義されています。

<unit>要素

このサンプルの C-Format では長さ・座標の単位 (ミリメートル) と角度の単位 (degree、度) が定義されています。

```
<unit>
    <distance unit="um" />
    <angle    unit="degree" />
</unit>
```

<angle>要素

<angle>は、CFormat 内で使用する角度の単位を定義します。以下の 2 つを使用することができます。

degree　度

radian　ラジアン

このサンプルでの角度は「度 (degree)」で表現されまています

<shape>要素

このサンプルでは、これまでに示したサンプルと違ってポリゴン (polygon) を使って形状を定義してます。また id も任意文字列で定義しています。

```
<shape>
    <polygon id="COMP_SHAPE_SOP_Advance_WF"
             points="-2305,-3375,-2305,-2600,-2600,-2600,-2600,-250,-2750,-250,
                     -2750,250,-2600,250,-2600,2600,-2305,2600,-2305,3375,
                     2305,3375,2305,2600,2600,2600,2600,250,2750,250,2750,
                     -250,2600,-250,2600,-2600,2305,-2600,2305,-3375,-2305,-3375" />
    <polygon id="PAD_SHAPE_r80000x125000"
             points="325,625,362,615,390,587,400,550,400,-550,390,-588,362,
                     -615,325,-625,-325,-625,-363,-615, -390,-588,-400,-550,
                     -400,550,-390,587,-363,615,-325,625,325,625" />
    <polygon id="PAD_SHAPE_r_sop_advance_wf"
             points="-2150,-1400,-2187,-1390,-2215,-1362,-2225,-1325,-2225,2225,
                     -2235,2263,-2262,2290,-2300,2300,-2230,2300,-2267,2310,-2295,
                     2338,-2305,2375,-2305,3300,-2295,3338,-2267,3365,-2230,3375,
                     -1580,3375,-1542,3365,-1515,3338,-1505,3300,-1505,2375,-1495,
                     2338,-1467,2310,-1430,2300,-1110,2300,-1072,2310,-1045,2338,
                     -1035,2375,-1035,3300,-1025,3338,-997,3365,-960,3375,-310,3375,
                     -272,3365,-245,3338,-235,3300,-235,2375,-225,2338,-197,2310,
                     -160,2300,160,2300,198,2310,225,2338,235,2375,235,3300,245,
                     3338,273,3365,310,3375,960,3375,998,3365,1025,3338,1035,3300,
                     1035,2375,1045,2338,1073,2310,1110,2300,1430,2300,1468,2310,
                     1495,2338,1505,2375,1505,3300,1515,3338,1543,3365,1580,3375,
                     2230,3375,2268,3365,2295,3338,2305,3300,2305,2375,2295,2338,
                     2268,2310,2230,2300,2300,2300,2263,2290,2235,2263,2225,2225,2225,
```

(次のページに続く)

(前のページからの続き)

```
                -1325,2215,-1362,2188,-1390,2150,-1400,-2150,-1400" />
</shape>
```

<polygon> は閉じたポリゴン図形を定義する要素です。以下の必須属性を持ちます。

id

この shape の識別子です。識別子はユニークである必要があります。

points

ポリゴン図形の頂点座標を定義します。"x 座標, y 座標, x 座標, y 座標, ..."と、座標を定義します。始点と終点の座標を同じとしてポリゴンを閉じます。

例えば以下の例は左下点座標を (-100,-200)、右上点座標を (200,300) とする矩形となります。またポリゴン図形の原点は (0, 0) となります (`polygon`)。

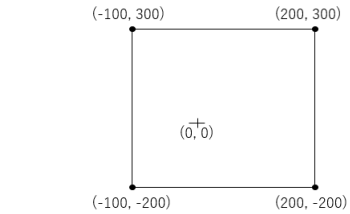


図 4.12　<polygon>

このサンプルの shape で定義されているポリゴン図形は `TPHR7904PB_polygon` の形状となります。

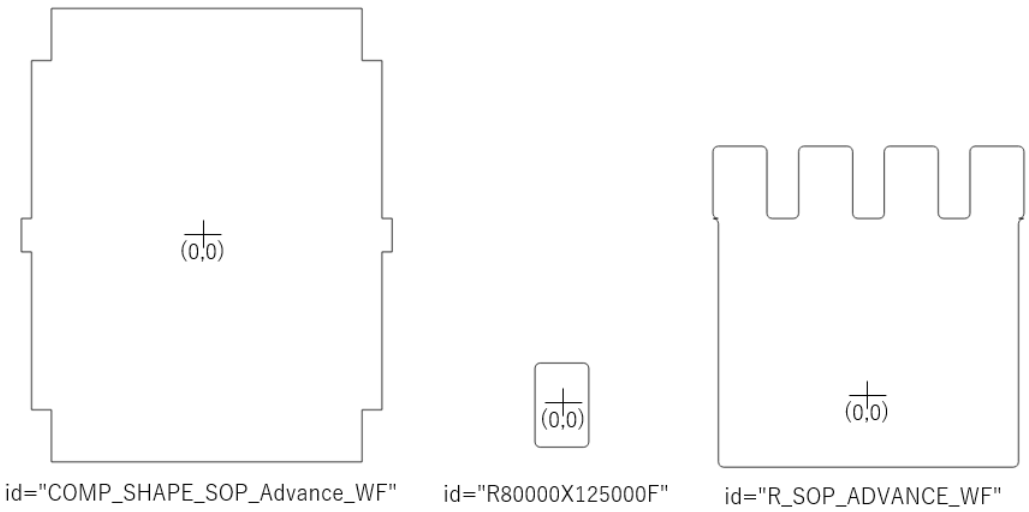


図 4.13　TPHR7904PB でのポリゴン図形

<padstack_def>要素

<padstack_def>要素では部品の電極（端子）を定義しています。

```
<padstack_def>
    <padstack id="R80000X125000F">
      <ref_shape shape_id="PAD_SHAPE_r80000x125000" type="Land"
                 x="0" y="0" angle="0" pad_layer="BOTTOM" />
    </padstack>
    <padstack id="R_SOP_ADVANCE_WF">
      <ref_shape shape_id="PAD_SHAPE_r_sop_advance_wf" type="Land"
                 x="0" y="0" angle="0" pad_layer="BOTTOM" />
    </padstack>
</padstack_def>
```

このサンプルでは、これまでの例と異なり<ref_shape>要素に **type** と **angle** の新しい 2 つの属性を使って padstack を定義しています。

```
<padstack id="R80000X125000F">
      <ref_shape shape_id="PAD_SHAPE_r80000x125000" type="Land"
                 x="0" y="0" angle="0" pad_layer="BOTTOM" />
</padstack>
```

type

padstack は端子（電極）以外にも VIA の形状を定義するためにも使用します。VIA を定義する場合、金属箔の形状以外にドリル穴やクリアランス (antipad) の形を定義する必要があります。それらを区別するために **type** 属性に以下の値を設定します。

Land　ランド形状。いわゆる金属箔の形状です。(デフォルト値）

NonConnection　ネットとの接続をしないランド形状。

Antipad　ランドと異なるネットの導体パターンとのクリアランス用の形状です

Drill　VIA のドリル穴の形状です。

Hole　穴の形状。Drill との違いは `via` を参照ください。

SolderMask　ソルダマスクの形状です。

Resist　ソルダレジスト（メタルマスク）の形状です。いわゆるペーストはんだが塗布された形状です。

このサンプルの例では金属箔の形状を表す **Land** を明示的に設定しますが、省略しても同じ意味になります。

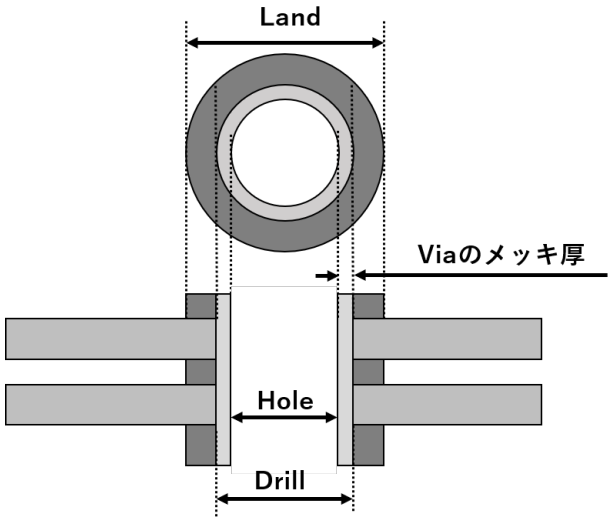
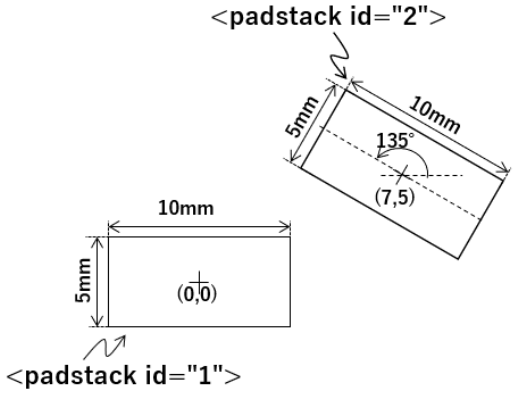


図 4.14　VIA 形状

angle

shape_id で参照した shape の回転角度を設定します。デフォルト値は 0 です。サンプルでの回転角度は 0 度なので、 **angle** を省略しても同じ意味になります。**angle** が指定された場合、shape を shape の原点を中心として半時計回りに回転させます。

下に **angle** を使った少し複雑な padstack の例を示します。この padstack は 2 つの shape によって構成されています。id="1"の shape は原点 (0,0) に配置され、id="2"の shape は座標 (7, 5) に 135°回転して配置されています。(`twoshape_padstack`)。




```
<unit>
    <distance unit="um" />
    <angle    unit="degree" />
</unit>
<shape>
    <rectangle id="1" width="10" height="5"/>
</shape>
<padstack_def>
 <padstack id="1">
     <ref_shape shape_id="1" x="0" y="0" angle="0"/>
 </padstack>
 <padstack id="2">
     <ref_shape shape_id="1" x="7" y="5"
angle="135"/>
 </padstack>
</padstack_def>
```

図 4.15　2 つの shape で構成される padstack の例

4.3.4 <module>要素

<module>要素には<socket>要素と<reference>要素を定義しています。

```
<module name="TPHR7904PB" type="PKG" shape_id="COMP_SHAPE_SOP_Advance_WF"
        x="0" y="0" angle="0" thickness="0.95" >
  <socket>要素
  <reference>要素
</module>
```

shape_id

id が COMP_SHAPE_SOP_Advance_WF の shape を参照しています。すなわち TPHR7904PB の外郭形状は `TPHR7904PB_bbox` となります。

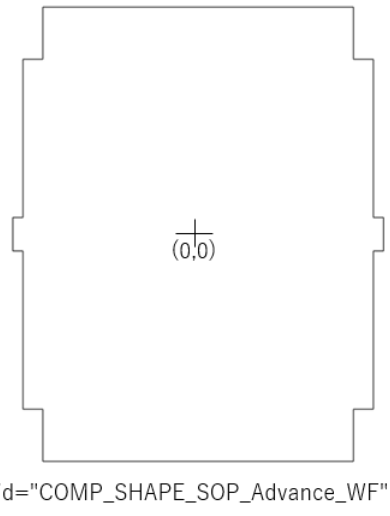


図 4.16　TPHR7904PB の外郭形状 (TOP VIEW)

angle

angle は 0 を定義しています。すなわち、このオプションは省略しても同じ意味となります。

thickness

部品の高さとして 0.95mm を定義しています。

type

PKG (半導体パッケージ）を定義しています。

<socket>要素

TPHR7904PB の<socket>要素には、<port>と<portgroup>の 2 つの要素を定義しています。

```
<socket  name="SOP_Advance_WF" >
    <port>要素
    <portgroup>要素
</socket>
```

これまでの例と異なり、TPHR7904PB の<socket>要素には<default>要素を定義していないことに注意してください。また、新たに<portgroup>要素も使用されています。

<port>要素

TPHR7904PB は電極（端子）を 5 つ持っています。<socket>要素には 5 つの<port>要素を定義しています。

また端子の形状の定義に<default>要素を使用していません。個々の<port>要素の **shape_id** 属性で端子の形状を定義しています。id が 1 から 4 の端子の形状は id が R80000X125000F の padstack、id が 5 の端子の形状は id が R_SOP_ADVANCE_WF の padstack となります。

```
<port id="1" padstack_id="R80000X125000F" x="-1905" y="-2750"  angle="180" name="D" />
<port id="2" padstack_id="R80000X125000F" x="-635" y="-2750"  angle="180" name="D" />
<port id="3" padstack_id="R80000X125000F" x="635" y="-2750"  angle="0" name="D" />
<port id="4" padstack_id="R80000X125000F" x="1905" y="-2750"  angle="0" name="G"/>
<port id="5" padstack_id="R_SOP_ADVANCE_WF" x="0" y="0"  angle="0" name="S" />
```

padstack_id

この端子の形状を定義している padstack の識別子を記載します。

<default>要素で端子形状が定義され、なおかつ<port>要素の padstack_id 属性でも端子形状が定義さえた場合は、padstack_id 属性を優先します。すなわち下記の 2 つの<socket>要素は同じ意味となります。

```
<socket  name="SOP_Advance_WF" >
    <port id="1" padstack_id="R80000X125000F" x="-1905" y="-2750"  angle="180"␣
↪name="D" />
    <port id="2" padstack_id="R80000X125000F" x="-635" y="-2750"  angle="180"␣
↪name="D" />
    <port id="3" padstack_id="R80000X125000F" x="635" y="-2750"  angle="0" name="D
↪" />
    <port id="4" padstack_id="R80000X125000F" x="1905" y="-2750"  angle="0" name=
↪"G"/>
    <port id="5" padstack_id="R_SOP_ADVANCE_WF" x="0" y="0"  angle="0" name="S" />
</socket>
```

```
<socket  name="SOP_Advance_WF" >
    <default>
        <port_shape padstack_id="R80000X125000F"/>
    </default>
    <port id="1" x="-1905" y="-2750"  angle="180" name="D" />
```

(次のページに続く)

(前のページからの続き)

```
    <port id="2" x="-635" y="-2750"  angle="180" name="D" />
    <port id="3" x="635" y="-2750"  angle="0" name="D" />
    <port id="4" x="1905" y="-2750"  angle="0" name="G"/>
    <port id="5" padstack_id="R_SOP_ADVANCE_WF" x="0" y="0"  angle="0" name="S" />
</socket>
```

angle

端子形状である padstack の回転角度を設定します。デフォルト値は 0 です。**x y** 属性で指定された座標に padstack を配置し、反時計回りに回転させます。

name

端子の信号名を定義します。TPHR7904PB の例では id が 1 と 2 と 3 の端子の信号名が D、id が 4 の信号名が G、id が 5 の端子が S となっています。TPHR7904PB は MOS-FET なので、それぞれドレイン (D)、ゲート (G)、ソース (S) となります。識別子である id は同一 socket 内でユニークである必要がりましたが、信号名である name はユニークの必要はありません (`FET`)。

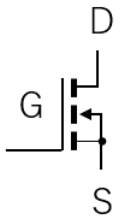


図 4.17　FET

TPHR7904PB の端子形状は `TPHR7904PB_ports` のようになります。

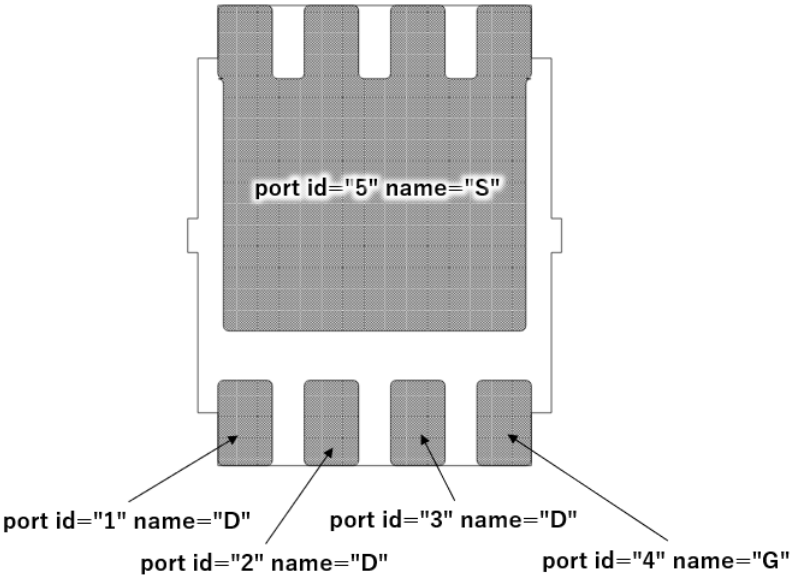


図 4.18　TPHR7904PB の端子

<portgroup>要素

<porggroup>要素は端子 (port) をグルーピングして制約を与えます。様々な制約がありますが、このサンプルでは **mustjoin** 制約を与えています。

```
<portgroup name="drain>
    <mustjoin/>
    <ref_port id="1"/>
    <ref_port id="2"/>
    <ref_port id="3"/>
</portgroup>
```

name

ポートグループにユニークな名前を定義します。ここで定義したポートグループ名は、他の要素からポートグループを参照する際に使用されます。

<ref_port>要素

グルーピングする端子 (port) を定義します。このサンプルでは id が 1 と 2 と 3 の端子をグルーピングしています。

<mustjoin/>要素

<mustjoin>要素が定義されたグループは、グループに属する端子 (port) はレイアウト設計時に必ず接続しなければならないことを意味します。このサンプルでは id が 1 と 2 と 3 の端子、すなわち全てのドレイン端子はレイアウト設計時に必ず接続しなければなりません。

<reference>要素

<reference>要素で、この部品のシミュレーションモデルと端子との参照関係を定義します。サンプルは SPICE モデルの定義用と S パラモデルの定義用の 2 つの<reference>要素を含んでいます。

<reference>要素の構文規則

サンプルでは SPICE モデルを参照する<reference>要素を定義しています。

TPHR7904PB は 5 つの端子を持ちますが FET であるため、実際の信号はソース (S)、ドレイン (D)、ゲート (G) の 3 つです。id が 1 から 3 の端子はドレイン (D) であるため、これらの端子は SPICE の同一の I/O ノードに接続されます。

```
<reference xmlns:spice="http://www.jeita.or.jp/LPB/spice"
           reffile = "TPHR7904PB.lib" format="SPICE" >
  <connection socket_name="SOP_Advance_WF" port_id="1">
        <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
  </connection>
  <connection socket_name="SOP_Advance_WF" port_id="2">
        <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
  </connection>
  <connection socket_name="SOP_Advance_WF" port_id="3">
        <spice:ref_port subckt="NMOS_TPHR7904PB" portid="1"/>
  </connection>
  <connection socket_name="SOP_Advance_WF" port_id="4">
        <spice:ref_port subckt="NMOS_TPHR7904PB" portid="2"/>
  </connection>
  <connection socket_name="SOP_Advance_WF" port_id="5">
        <spice:ref_port subckt="NMOS_TPHR7904PB" portid="5"/>
  </connection>
</reference>
```

`TPHR7904PB_spice_reference` は C-Format の port と SPICE モデルの I/O ノードの関係を示しています。

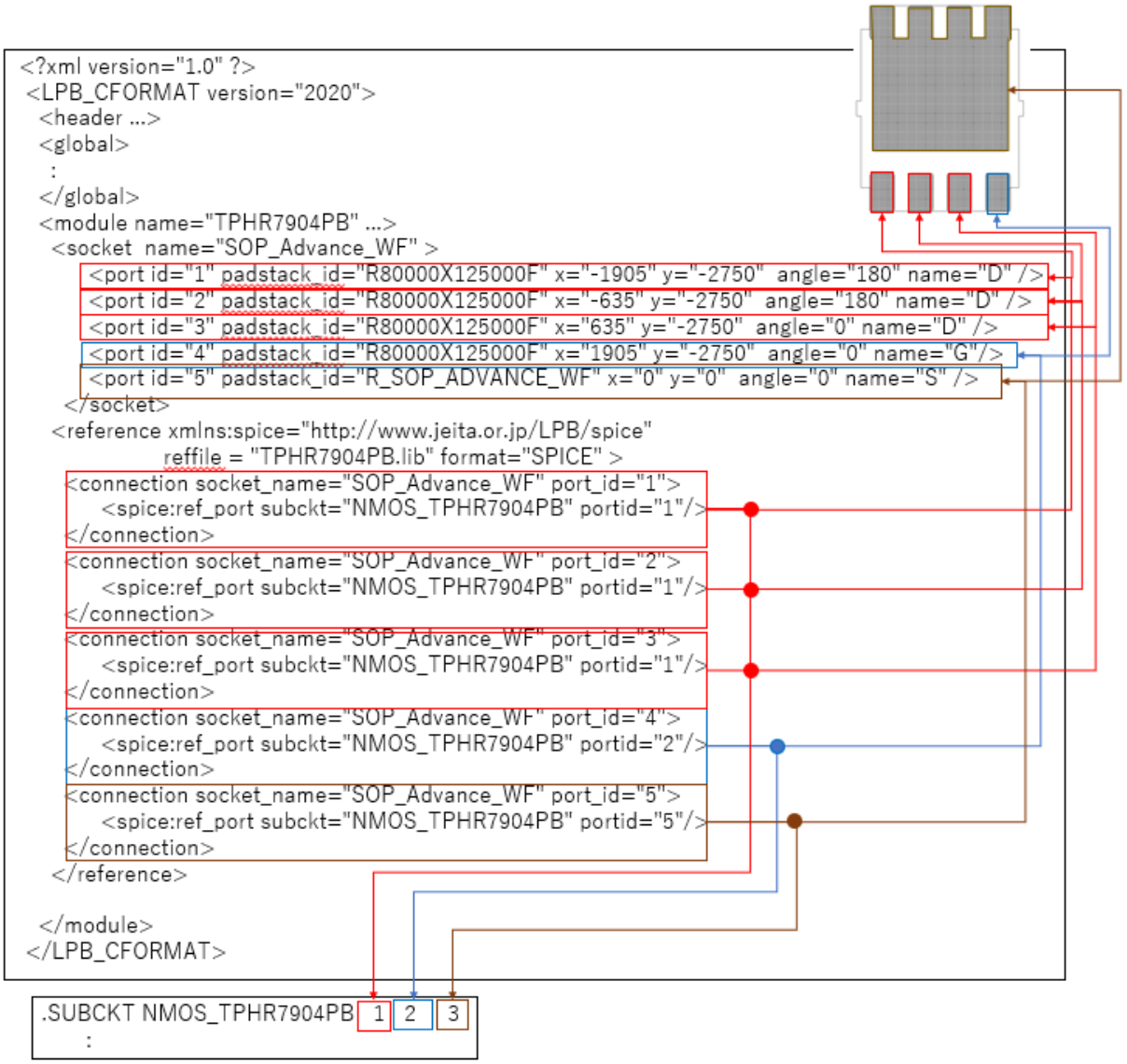


図 4.19　TPHR7904PB の SIPCE モデル用の<reference>要素

4.4 DDR3

本章では DDR3 を例として、より複雑な C-Format について学習します。C-Format の全ての機能を説明することはせず、基本的な概念について例を示しながら説明します。

4.4.1 パッケージ形状の定義

まず C-Format の基本であるパッケージの構造を定義します。図 4.20 に DDR3 のパッケージレイアウトを、図 4.21 に端子レイアウトを示します。以後このパッケージを例に C-Format の説明を行います。

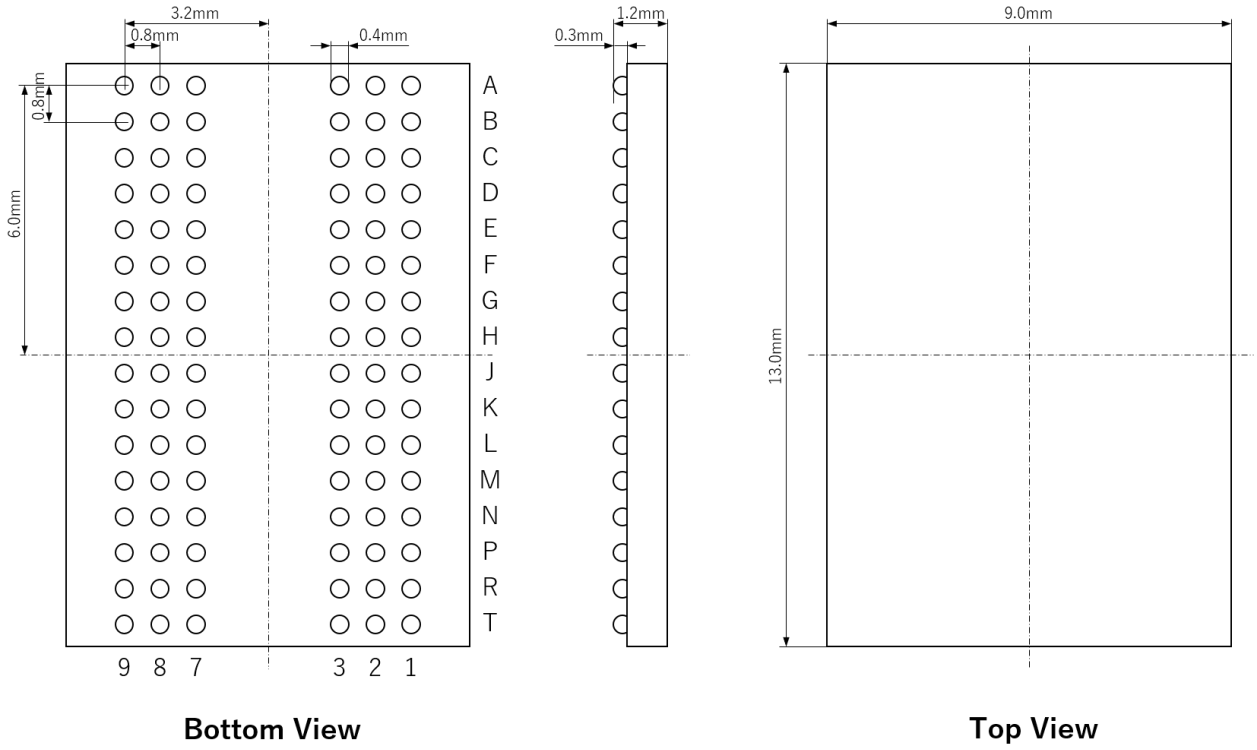
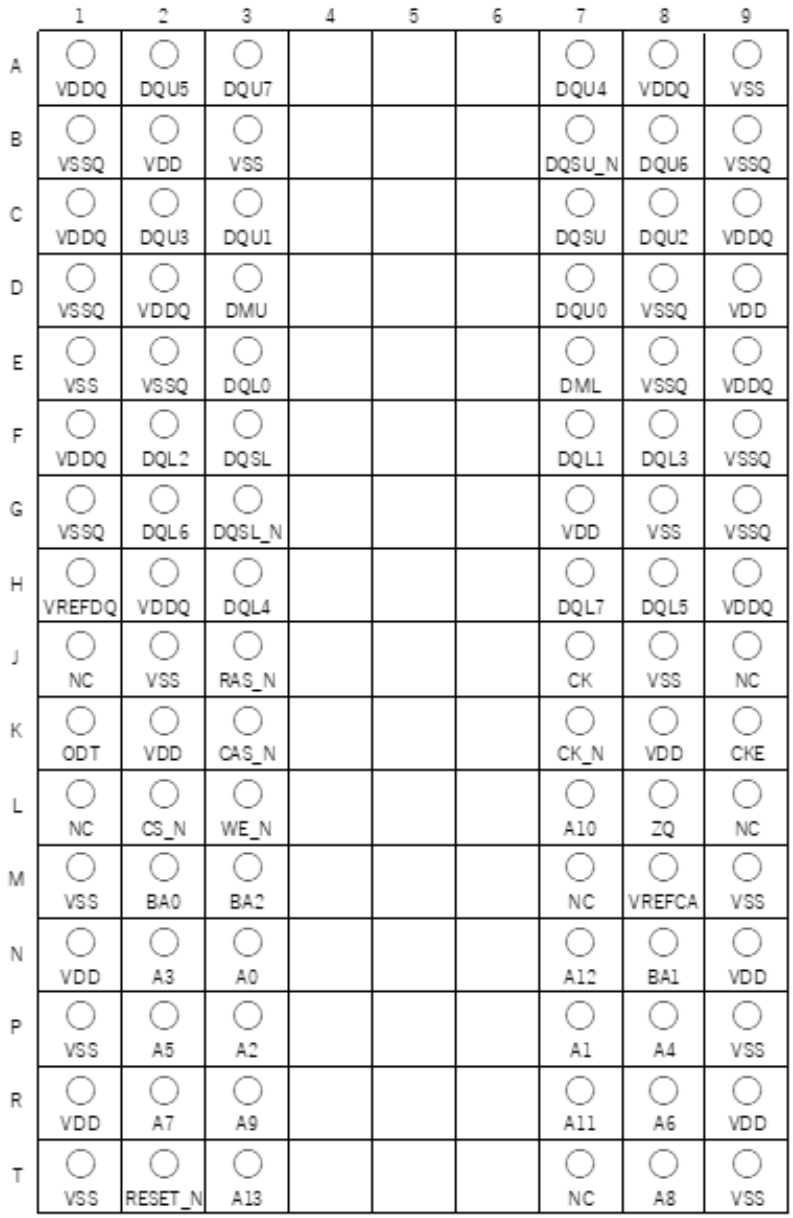


図 4.20　DDR3 package layout



	1	2	3	4	5	6	7	8	9
A	VDDQ	DQU5	DQU7				DQU4	VDDQ	VSS
B	VSSQ	VDD	VSS				DQSU_N	DQU6	VSSQ
C	VDDQ	DQU3	DQU1				DQSU	DQU2	VDDQ
D	VSSQ	VDDQ	DMU				DQU0	VSSQ	VDD
E	VSS	VSSQ	DQL0				DML	VSSQ	VDDQ
F	VDDQ	DQL2	DQSL				DQL1	DQL3	VSSQ
G	VSSQ	DQL6	DQSL_N				VDD	VSS	VSSQ
H	VREFDQ	VDDQ	DQL4				DQL7	DQL5	VDDQ
J	NC	VSS	RAS_N				CK	VSS	NC
K	ODT	VDD	CAS_N				CK_N	VDD	CKE
L	NC	CS_N	WE_N				A10	ZQ	NC
M	VSS	BA0	BA2				NC	VREFCA	VSS
N	VDD	A3	A0				A12	BA1	VDD
P	VSS	A5	A2				A1	A4	VSS
R	VDD	A7	A9				A11	A6	VDD
T	VSS	RESET_N	A13				NC	A8	VSS

図 4.21　DDR3 port map

基本図形の定義

最初にパッケージの外形形状と端子の形状を定義します。既に前章までで形状の定義方法に関しては述べています。ここでは復習を兼ねて簡単に説明するにとどめます。

C-Format 内で使用される形状は<global>要素内の<unit>, <shape>, <padstack_def>の子要素に定義します。以下に、その例を示します。

```
<global>
  <unit>
```

(次のページに続く)

(前のページからの続き)

```
    <distance unit="um"/>
  </unit>
  <shape>
    <rectangle id="boundary" width="9000" height="13000"/>
    <circle id="ball" diameter="400"/>
  </shape>
  <padstack_def>
    <padstack id="port">
      <ref_shape shape_id="ball" x="0" y="0" pad_layer="BOTTOM"/>
    </padstack>
  </padstack_def>
</global>
```

<unit>要素

図形を定義する際に使用する長さの単位は<unit>要素内の<distance>子要素で定義します。この例では長さの単位としてマイクロンメートルを使用します。

```
<unit>
  <distance unit="um"/>
</unit>
```

<shape>要素

<shape>要素で基本図形を定義します。図 4.20 に示すように、このパッケージの外形は 9000um × 13000um(9mm × 13mm) の長方形、端子は直径 400um(0.4mm) の円形です。長方形は<rectangle>要素、円形は<circle>要素で定義します。

```
<shape>
  <rectangle id="boundary" width="9000" height="13000"/>
  <circle id="ball" diameter="400"/>
</shape>
```

id 属性は、ここの図形を参照するための識別子です。C-Format ファイル内でユニークである必要があります。

<padstack_def>要素

端子などの層構造を持つ図形（パッドスタック）は<padstack_def>要素で定義します。このパッケージ例では、パッケージの下部に円形の端子を定義します。

```
<padstack_def>
  <padstack id="port">
```

(次のページに続く)

(前のページからの続き)

```
    <ref_shape shape_id="ball" x="0" y="0" pad_layer="BOTTOM"/>
  </padstack>
</padstack_def>
```

shape_id 属性に参照する図形の識別子 **ball** を定義します。端子をパッケージ下部に定義するので **pad_layer** 属性に **BOTTOM** を定義します。

パッケージ外形と端子の定義

下にパッケージ外形と端子を定義した C-Format の例を示します。このパッケージには 96 個の端子がありますが、例では 5 個の端子だけを記載してます。完全な例は **DDR-example.xml** を参照してください。

```
<module name="DDR3"
        shape_id="boundary" x="0" y="0" thickness="1200" type="PKG">
  <socket name="ddrports">
    <default>
      <port_shape padstack_id="port"/>
    </default>
    <port id="A1" x="-3200" y="6000" name="VDDQ" direction="inout" type="power"/>
    <port id="A2" x="-2400" y="6000" name="DQU5" direction="inout" type="signal"/>
    <port id="A3" x="-1600" y="6000" name="DQU7" direction="inout" type="signal"/>
    <port id="A7" x="1600"  y="6000" name="DQU4" direction="inout" type="signal"/>
    <port id="A8" x="2400"  y="6000" name="VDDQ" direction="inout" type="power"/>
          :
  </socket>
</module>
```

<module>要素

<module>要素の属性に部品の名称、外形および厚みを定義します。

```
<module name="DDR3"
        shape_id="boundary" x="0" y="0" thickness="900" type="PKG">
```

name 属性に部品名称、**shape_id** 属性にパッケージ外形の図形の識別子、**thickness** にパッケージの厚みを定義します。このパッケージの例では厚みは 900um(0.9mm) です。**type** にはモジュールのタイプを設定します。この例は IC パッケージなので **type** 属性は **PKG** となります。

<socket>要素

パッケージの端子は<socket>要素の子要素で定義します。**name** 属性でソケットの名称を定義します。これはシミュレーションモデルと端子とのクロスリファレンスを定義するときに使用します。

<default>要素

<default>要素で端子の形状を定義します。子要素の<port_shape>要素で端子の形状を定義した padstack を参照します。この例では端子の形状は **port** と名付けられたパッドスタックで定義されています。

```
<socket name="ddrports">
  <default>
    <port_shape padstack_id="port"/>
  </default>
```

以上で前章までの復習は終わります。以後、初めての内容も含みますので詳しく解説していきます。

<port>要素

<port>要素でパッケージの端子を定義します。それぞれの端子の形状は<default>要素で定義されています。このパッケージには 96 個の端子がありますが、以下の例では 5 個の端子だけを記載してます。

```
<port id="A1" x="-3200" y="6000" name="VDDQ" direction="inout" type="power"/>
<port id="A2" x="-2400" y="6000" name="DQU5" direction="inout" type="signal"/>
<port id="A3" x="-1600" y="6000" name="DQU7" direction="inout" type="signal"/>
<port id="A7" x="1600"  y="6000" name="DQU4" direction="inout" type="signal"/>
<port id="A8" x="2400"  y="6000" name="VDDQ" direction="inout" type="power"/>
```

id

端子の識別子（端子番号）です。同一 socket 内で識別子はユニークである必要があります。C-Format の規定では **id** の付け方に定めはありませんが、広く一般的に使用されているルールに従うことを推奨します。この例では JEDEC ルールに従って **id** を与えています。

x y

端子の配置座標です。端子形状の padstack の原点を、ここで指定された座標に配置します。

name

端子名を設定します。端子名は **id** とは異なりユニークである必要はありません。この資料で使用している例では電源・グランド端子の端子名 **VDD** 、 **VDDQ** 、 **VSS** 、 **VSSQ** はユニークではありません。

direction

信号の入出力方向を設定します。以下のいずれかを設定します。省略時は **inout** (双方向) となります。

inout　入力

output　出力

inout　双方向

電源・グランド端子には信号の方向はありませんので、**inout** を設定します。

type

端子に入出力する信号のタイプを設定します。例示しているパッケージでは **signal** 、 **power** 、 **ground** の 3 種類だけが使われていますが、以下の 8 種類のタイプがあります。いずれかの値を設定します。

power　電源

ground　グランド

signal　一般信号

floating　如何なるネットにも接続してはならない端子です。この端子は未接続のままにしておくことを強制します。

dontcare　サーマルボールや non-connection のように論理的には意味を持たない端子

through　いわゆるフィードスルー端子と呼ばれる端子。フィードスルー端子は如何なる内部回路とも接続せず単にパッケージを貫通しているだけです

thermal　熱端子です。VHDL-AMS や SPICE などのシミュレーションモデルの熱端子と接続する場合に使用します。この端子に接続する熱（温度）の単位は K(Kelvin) です

thermal_c　thermal と同様な熱端子です。ただし、この端子に接続する熱（温度）の単位は ℃ (Celsius) です

4.4.2 制約の定義

既に<socket>要素の子要素の<default>と<port>は説明しました。<socket>要素は、これら以外に設計制約を定義するための子要素を持ちます。本章では、設計制約を定義するための子要素について説明します。以下は、<socket>要素の構造です。

```
<socket name="ddrports">
  <default>要素
  <port>要素
  <portgroup>要素
  <constraint>要素
  <swappable_port>要素
</socket>
```

単位系の定義

形状の定義では使用する単位系は長さだけでした。ここでは、それに追加して制約で使用するインピーダンスと時間の単位を定義します。以下の例は、時間単位としてピコセカンド、インピーダンス単位としてオームを定義した例です。

```
<unit>
  <impedance unit="ohm"/>
  <time unit="ps"/>
  <distance unit="um"/>
  <voltage unit="V"/>
</unit>
```

<time>

unit 属性で時間単位を設定します。以下のいずれかの値を設定します。もし時間単位の設定が省略された場合は、時間の単位は **s** (秒）となります。

ps　ピコ秒

ns　ナノ秒

us　マイクロ秒

ms　ミリ秒

s　秒

<impedance>

unit 属性でインピーダンスの単位を設定します。以下のいずれかの値を設定します。もし設定が省略された場合は、インピーダンスの単位は **ohm** (Ω）となります。

fohm　フェムトΩ

pohm　ピコΩ

nohm　ナノΩ

uohm　マイクロΩ

mohm　ミリΩ

ohm　Ω

kohm　キロΩ

Mohm　メガΩ

<voltage>

unit 属性で電圧の単位を設定します。以下のいずれかの値を設定します。もし設定が省略された場合は、電圧の単位は **V** となります。

pV　ピコボルト

nV　ナノボルト

uV　マイクロボルト

mV　ミリボルト

V　ボルト

kV　キロボルト

<portgroup>要素

<portgroup>要素（端子グループ）は、ある特定の意味を持つ端子の集まりで、これに対し様々な制約を与えることができます。

先ず、基本的な端子グループの作り方を見ていきましょう。下記記述は、DDR3 の Low チャネルのデータストローブの差動信号の端子 (DQSL, DQSL_N) をグルーピングして **DATASTROB_L** というグループ名を与えています。

```
<portgroup name="DATASTROB_L">
   <ref_port name="DQSL"/>
   <ref_port name="DQSL_N"/>
</portgroup>
```

また以下は Low チャネル側のデータ端子（DQL0～ DQL7) をグルーピングして **DATA_L** というグループ名を与えた例です。

```
<portgroup name="DATA_L">
  <ref_port name="DQL0"/>
  <ref_port name="DQL1"/>
  <ref_port name="DQL2"/>
  <ref_port name="DQL3"/>
  <ref_port name="DQL4"/>
  <ref_port name="DQL5"/>
  <ref_port name="DQL6"/>
  <ref_port name="DQL7"/>
</portgroup>
```

<portgroup>要素の **name** 属性　グループ名です。グループ名は<socket>要素内でユニークである必要が有ります。

<ref_port>要素でグルーピングする端子を定義します。例では端子名 (**name**) で端子を参照していますが、端子番号 (**id**) で参照することもできます。以下は端子番号で DQSL, DQSL_N をグルーピングした例です。

```
<portgroup name="DATASTROB_L">
   <ref_port id="F3"/>
   <ref_port id="G3"/>
</portgroup>
```

ここで、端子名でのグルーピングと端子番号でのグルーピングの記述方法の違いを説明するために別の例を示しま

す。この DDR の例は VSS と VSSQ の 2 つのグランドを持ちます。このグランド端子をグルーピングする例を考えます。端子名で記載する場合は以下のようになります。すなわち端子名が VSS もしくは VSSQ である 21 個の端子がグループとなります。

```
<portgroup name="GND">
   <ref_port name="VSS"/>
   <ref_port name="VSSQ"/>
</portgroup>
```

これに対し端子番号で参照する場合は、一つ一つの端子を個別に設定します。以下のような記述になります。

```
<portgroup name="GND">
  <ref_port id="A9">
  <ref_port id="B1">
  <ref_port id="B3">
  <ref_port id="B9">
  <ref_port id="D1">
  <ref_port id="D8">
  <ref_port id="E1">
  <ref_port id="E2">
  <ref_port id="E8">
  <ref_port id="F9">
  <ref_port id="G1">
  <ref_port id="G8">
  <ref_port id="G9">
  <ref_port id="J2">
  <ref_port id="J8">
  <ref_port id="M1">
  <ref_port id="M9">
  <ref_port id="P1">
  <ref_port id="P9">
  <ref_port id="T1">
  <ref_port id="T9">
</portgroup>
```

以上 2 つの記載方法は、いずれも同じ意味となります。状況によって使い分けてください。

差動信号

ストローブ信号の例に戻ります。グループした端子が差動信号であることを明示したいときは、以下のように端子グループに **<differential/>** を追加します。また正信号/負信号を表す **polarity** 属性を追加します。

```
<portgroup name="DATASTROB_L">
  <differential/>
  <ref_port name="DQSL" polarity="POSITIVE"/>
  <ref_port name="DQSL_N" polarity="NEGATIVE"/>
</portgroup>
```

polarity

作動信号の正信号および負信号を表す属性です。以下のいずれかを設定します。

POSITIVE　正信号

NEGATIVE　負信号

では、この信号にインピーダンスとスキューの制約を追加してみます。これらの制約は<constraint>要素の子要素として定義します。

インピーダンス制約

インピーダンス制約は<impedance>要素で定義します。下はデータ信号 (DQL0～DQL7) に対し 50 Ω ± 10% のインピーダンス制約を与える例です。

```
<constraint>
  <impedance port_name="DQL0" min="45" typ="50" max="55"/>
  <impedance port_name="DQL1" min="45" typ="50" max="55"/>
  <impedance port_name="DQL2" min="45" typ="50" max="55"/>
  <impedance port_name="DQL3" min="45" typ="50" max="55"/>
  <impedance port_name="DQL4" min="45" typ="50" max="55"/>
  <impedance port_name="DQL5" min="45" typ="50" max="55"/>
  <impedance port_name="DQL6" min="45" typ="50" max="55"/>
  <impedance port_name="DQL7" min="45" typ="50" max="55"/>
</constraint>
```

port_name

インピーダンス制約を与える端子名です。

typ min max

インピーダンス値を設定します。公差を設定しない場合は **min** 、 **max** は不要です（下例）。

```
<constraint>
  <impedance port_name="DQL0" typ="50"/>
  <impedance port_name="DQL1" typ="50"/>
  <impedance port_name="DQL2" typ="50"/>
  <impedance port_name="DQL3" typ="50"/>
  <impedance port_name="DQL4" typ="50"/>
  <impedance port_name="DQL5" typ="50"/>
  <impedance port_name="DQL6" typ="50"/>
  <impedance port_name="DQL7" typ="50"/>
<constraint>
```

上記の例では端子毎に制約を与えていますが、端子グループを使って一括で制約を与えることもできます。下の例はデータ信号 (DQL0～DQL7) をグループ化し、グループ全体に 50 Ω ± 10% のインピーダンス制約を与えてい

ます。

```
<portgroup name="DATA_L">
  <ref_port name="DQL0"/>
  <ref_port name="DQL1"/>
  <ref_port name="DQL2"/>
  <ref_port name="DQL3"/>
  <ref_port name="DQL4"/>
  <ref_port name="DQL5"/>
  <ref_port name="DQL6"/>
  <ref_port name="DQL7"/>
</portgroup>
<constraint>
  <impedance group_name="DATA_L" min="45" typ="50" max="55"/>
</constraint>
```

group_name

インピーダンス制約を与える端子グループの名称です。

差動信号に作動インピーダンスを定義する場合は、作動信号を定義した端子グループに対して制約を与えます。下の例は、データストローブ信号に 100 Ω ± 10% の差動インピーダンス制約を設定した例です。

```
<portgroup name="DATASTROB_L">
  <differential/>
  <ref_port name="DQSL" polarity="POSITIVE"/>
  <ref_port name="DQSL_N" polarity="NEGATIVE"/>
</portgroup>
<constraint>
  <impedance group_name="DATASTROB_L" type="differential" min="90" typ="100" max="110"/
↪>
</constraint>
```

type

インピーダンスのタイプを設定します。以下の 3 つの何れかを設定します。省略時はシングルエンド・インピーダンスを意味します。

single　シングルエンド・インピーダンス。デフォルト値

differential　差動インピーダンス

common　コモンモード・インピーダンス

スキュー制約

スキューの制約は<skew>要素で定義します。例えばデータストローブ信号の差動信号間の遅延差 (skew) を 5ps に設定する場合は、以下のように記述します。

```
<constraint>
  <skew group_name="DATASTROB_L" max="5"/>
</constraint>
```

また、データ信号の信号間の遅延 (skew) を 5ps に設定する場合は、以下のように記述します。

```
<constraint>
  <skew group_name="DATA_L" max="5"/>
</constraint>
```

group_name

スキュー制約を与える信号のグループ名を設定します。ここで参照しているグループに属する信号の遅延差を **max** 属性で与えた値以下に抑えることを要求します。

max

許容される最大スキュー値を設定します。

スキュー制約には、スキューの基準信号 (reference_port_name) を設定することができます。この記述方法は、例えば 図 4.22 のタイミングチャートようなデータストローブとデータ信号の関係の制約を設定する際に使用します。

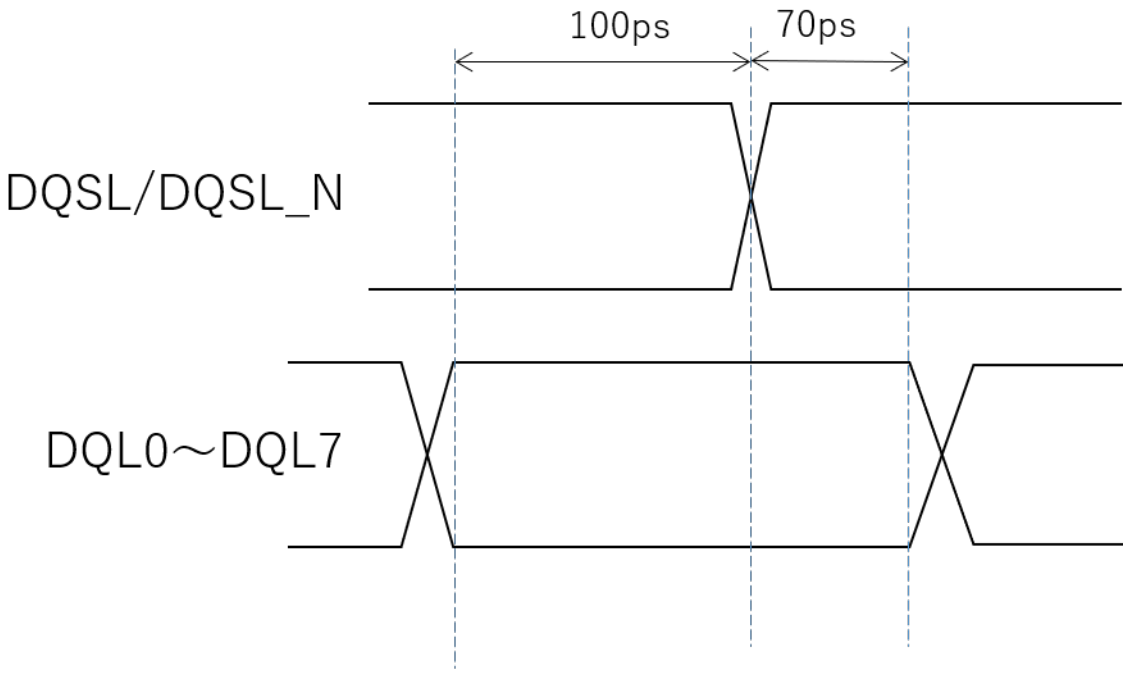


図 4.22　DQS vs DQ Timing Chart

```
<portgroup name="DATA_L">
  <ref_port name="DQL0"/>
```

(次のページに続く)

(前のページからの続き)

```
  <ref_port name="DQL1"/>
  <ref_port name="DQL2"/>
  <ref_port name="DQL3"/>
  <ref_port name="DQL4"/>
  <ref_port name="DQL5"/>
  <ref_port name="DQL6"/>
  <ref_port name="DQL7"/>
</portgroup>
<constraint>
  <skew group_name="DATA_L" reference_port_name="DQSL" max="70" min="-100"/>
</constraint>
```

reference_port_name

スキューの基準信号となる端子名

4.4.3 電源設定

本稿で使用している DDR の例には VDDQ/VSSQ（DQ 用の電源・グランド）と VDD/VSS（DQ 以外の電源・グランド）の 2 つの電源系統があります。C-Format には、これらの電源系統は<powerdomain_group>要素で定義します。電源定義はデジタル・アナログが混載した IC で、それぞれの専用電源とそれに対応する信号を定義する場合などに使用します。

まず、それぞれの電源系統の電圧を定義しましょう。1.5V±5% とすると以下のような記述になります。

```
<powerdomain_group
  pwr_port_name="VDDQ" pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575"
  gnd_port_name="VSSQ" gnd_typ="0"/>
<powerdomain_group
  pwr_port_name="VDD" pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575"
  gnd_port_name="VSS" gnd_typ="0"/>
```

pwr_port_name gnd_port_name

pwr_port_name に電源端子名を、 **gnd_port_name** に、それと対となるグランド端子名を定義します。

pwr_min pwr_typ pwr_max gnd_min gnd_typ gnd_max

pwr_typ で電源電圧を定義します。公差を定義する場合は、 **pwr_min** 、 **pwr_max** に最小許容電圧値、最大許容電圧値を定義します。このサンプルの例では電源電圧 1.5V、± 5% の公差なので pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575" と定義しています。

同様に **gnd_typ** でグランド電圧、 **gnd_min** 、 **gnd_max** 最小許容電圧値、最大許容電圧値を定義します。

以上の例ではグランドを VSSQ と VSS の 2 種類に分割していますが、単一グランドとして定義する場合は、端子グループを使用して以下のように記述します。

```
<portgroup name="GND">
  <ref_port name="VSS"/>
  <ref_port name="VSSQ"/>
</portgroup>
<powerdomain_group
  pwr_port_name="VDDQ" pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575"
  gnd_group_name="GND" gnd_typ="0"/>
<powerdomain_group
  pwr_port_name="VDD" pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575"
  gnd_group_name="GND" gnd_typ="0"/>
```

pwr_group_name gnd_group_name

> **pwr_group_name** で電源端子の端子グループ、 **gnd_group_name** でグランド端子の端子グループを定義します。

以上に加え各電源系統に、それに対応する信号を割り当てることができます。例えば DDR では VDDQ はデータ信号 (DQ) の専用電源となりますが、これを明示的に定義する場合は、<powerdomain>要素の子要素に電源系統に割り当てる信号（端子）を定義します。下記は VDDQ/VSSQ の電源系統と DQL/DQU のデータ信号の関係を明記した例です。

```
<portgroup name="DATA_L">
  <ref_port name="DQL0"/>
  <ref_port name="DQL1"/>
  <ref_port name="DQL2"/>
  <ref_port name="DQL3"/>
  <ref_port name="DQL4"/>
  <ref_port name="DQL5"/>
  <ref_port name="DQL6"/>
  <ref_port name="DQL7"/>
</portgroup>
<portgroup name="DATA_U">
  <ref_port name="DQU0"/>
  <ref_port name="DQU1"/>
  <ref_port name="DQU2"/>
  <ref_port name="DQU3"/>
  <ref_port name="DQU4"/>
  <ref_port name="DQU5"/>
  <ref_port name="DQU6"/>
  <ref_port name="DQU7"/>
</portgroup>
<powerdomain_group
   pwr_port_name="VDDQ" pwr_min="1.425" pwr_typ="1.5" pwr_max="1.575"
   gnd_port_name="VSSQ" gnd_typ="0">
    <ref_portgroup name="DATA_L"/>
    <ref_portgroup name="DATA_U"/>
</powerdomain_group>
```

4.4.4 端子の交換

DDR では同じバイトレーン内のデータ信号は交換可能です。すなわち DQL0～DQL7 は互いに交換可能です。同様に DQU0～DQU7 も交換可能です。C-Format で、これを明示的に定義するには<socket>要素の下の、**<swappable_port>** を使用します。 以下に、その例を示します。<ref_port>要素の **name** 属性 に端子名を設定します。<swappable_port>内で参照された端子は、互いに交換可能であることを意味します。

```
<swappable_port>
  <ref_port name="DQL0"/>
  <ref_port name="DQL1"/>
  <ref_port name="DQL2"/>
  <ref_port name="DQL3"/>
  <ref_port name="DQL4"/>
  <ref_port name="DQL5"/>
  <ref_port name="DQL6"/>
  <ref_port name="DQL7"/>
</swappable_port>
<swappable_port>
  <ref_port name="DQU0"/>
  <ref_port name="DQU1"/>
  <ref_port name="DQU2"/>
  <ref_port name="DQU3"/>
  <ref_port name="DQU4"/>
  <ref_port name="DQU5"/>
  <ref_port name="DQU6"/>
  <ref_port name="DQU7"/>
</swappable_port>
```

上の例では端子名で交換可能な端子を参照していましたが、端子番号 (**id**) で参照することも可能です。以下に、その例を示します。<ref_port>要素の **id** 属性に端子番号を設定します。

```
<swappable_port>
  <ref_port id="A2"/>
  <ref_port id="A3"/>
  <ref_port id="A7"/>
  <ref_port id="B8"/>
  <ref_port id="C2"/>
  <ref_port id="C3"/>
  <ref_port id="C8"/>
  <ref_port id="D7"/>
</swappable_port>
<swappable_port>
  <ref_port id="E3"/>
  <ref_port id="F2"/>
  <ref_port id="F7"/>
  <ref_port id="F8"/>
  <ref_port id="G2"/>
```

(次のページに続く)

(前のページからの続き)

```
  <ref_port id="H3"/>
  <ref_port id="H7"/>
  <ref_port id="H8"/>
</swappable_port>
```

4.4.5 3 次元データの参照

C-Format には部品の 3 次元データ（STEP, SAT, IGES) を参照する機能があります。ここでは 3D_data に示す STEP データを参照する例を示します。

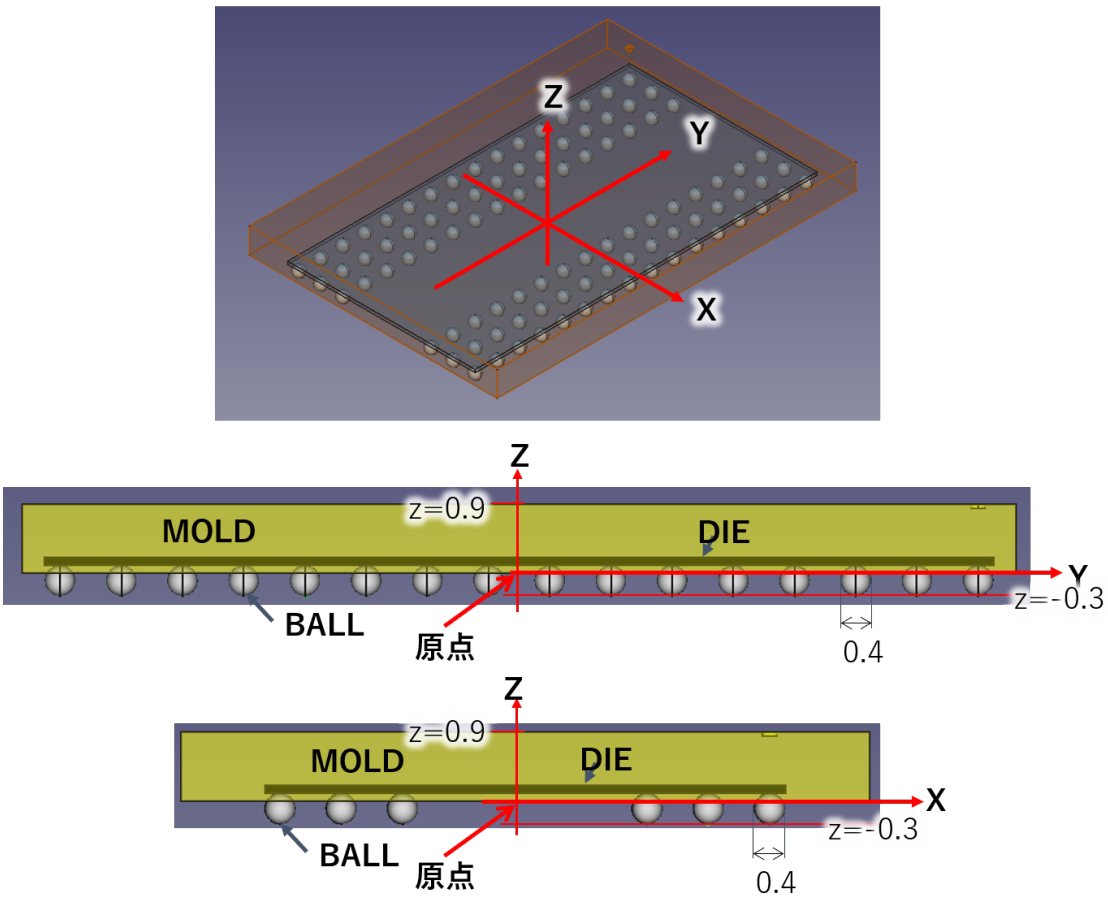


図 4.23　DDR3 の 3 次元データ

3 次元データファイルは <reference>で参照します。下に<reference>の例を示します。

```
<reference xmlns:step="http://www.jeita.or.jp/LPB/step"
           format="STEP"
           reffile="DDR3-example.step" >
    <affine_transformation
        a11="1.0" a12="0.0" a13="0.0" a14="0.0"
        a21="0.0" a22="1.0" a23="0.0" a24="0.0"
        a31="0.0" a32="0.0" a33="1.0" a34="0.3"
    />
    <material name="epoxyresin" ref_rule_name="PackageRule">
        <step:ref_product name="MOLD">
    </material>
    <material name="sillicon" ref_rule_name="PackageRule">
        <step:ref_product name="DIE">
    </material>
    .......
</reference>
```

xmlns format

参照する 3 次元データファイルのフォーマットにより **xmlns** と **format** は以下のようになります。

STEP フォーマット

```
xmlns:step="http://www.jeita.or.jp/LPB/step"
format="STEP"
```

STA フォーマット

```
xmlns:sat="http://www.jeita.or.jp/LPB/sat"
format="SAT"
```

IGES フォーマット

```
xmlns:iges="http://www.jeita.or.jp/LPB/iges"
format="IGES"
```

reffile

参照する 3D データのモデルのファイル名です。

多くの場合、3 次元データファイルで定義されている図形の原点と C-Format での原点は一致していません。本稿の例 (`3D_data`) では 3 次元データの原点はパッケージ下面の中心にあります。この原点を C-Format が配置された座標に一致させると、パッケージの BALL は基板にめり込んだ形になります (図 4.24)。これを防ぐためには 3 次元データの原点を半田ボールの高さ分だけ上方に移動させる必要があります。CFormat ではアファイン変換行列を使って原点座標を一致させます

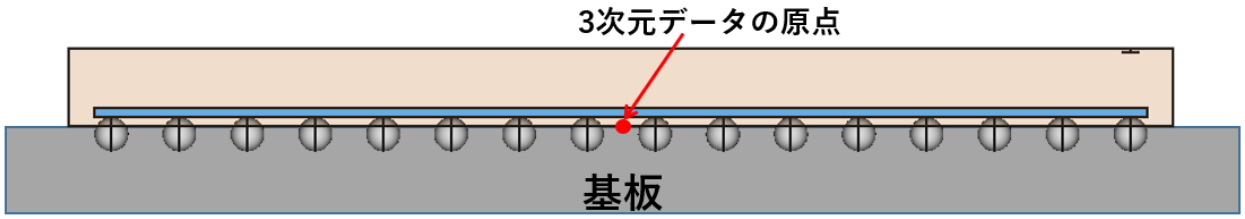


図 4.24　座標変換せずに配置した場合

図形を移動・回転させたり拡大・縮小させたりする変換をアファイン返還と呼びます。(**??**) に示す行列式で定義されます。ここで xyz は図形のオリジナル座標、$x'y'z'$ は変換後の座標です。

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.1}$$

下式のアフィン変換行列 ((**??**)) は原点座標を Z 方向に 0.3mm（半田ボール）だけシフトさせるものです。この行列式で 3 次元データの座標を変換させて配置すると BALL 下面が基板表面と一致します (`placed_3D_data`)。

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z + 0.3 \\ 1 \end{bmatrix} \tag{4.2}$$

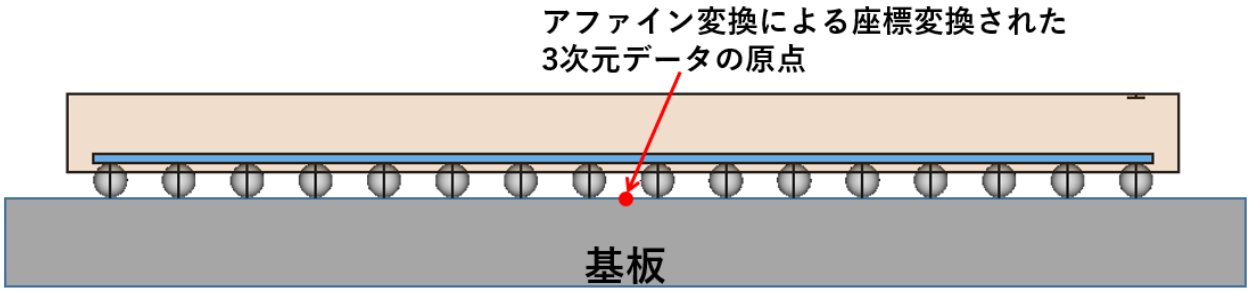


図 4.25　アファイン変換により座標して配置した場合

下に CFormat にアファイン変換行列を定義する例を示します。<affine_transformation>でアファイン変換式 ((**??**)) の要素 (a_{11} 〜 a_{34}) を定義します。下の例は Z 方向に 0.3mm 移動させる例です。

```
<affine_transformation
    a11="1.0" a12="0.0" a13="0.0" a14="0.0"
    a21="0.0" a22="1.0" a23="0.0" a24="0.0"
    a31="0.0" a32="0.0" a33="1.0" a34="0.3"
/>
```

STEP や IGES のような 3 次元データフォーマットは物体の形状は定義できますが、個々の物体の素材を定義することはできません。C-Format は<material>要素で 3 次元データに含まれる物体の素材を定義することができます。

下に、STEP データフォーマットで定義されている物体の素材を定義する例を示します。<material>要素の **name** で素材名を定義します。**ref_rule_name** はその素材が定義されているルール名で、LPB R-Format の<Physicaldesign>で定義されています。ここでは<Physicaldesign>の詳細は省略しますが、誘電率や比熱等の物性情報が定義されています。

<step:ref_product>要素の **name** で<material>で定義された素材を使用している STEP データフォーマットの物体を定義します。

```
<material name="sillicon" ref_rule_name="PackageRule">
    <step:ref_product name="DIE">
</material>
<material name="epoxyresin" ref_rule_name="PackageRule">
    <step:ref_product name="MOLD">
</material>
<material name="solder" ref_rule_name="PackageRule">
    <step:ref_product name="BGA_B7"/>
    <step:ref_product name="BGA_B9"/>
    <step:ref_product name="BGA_N9"/>
    <step:ref_product name="BGA_M1"/>
      .....
</material>
```

上記サンプルでは、DIE の素材を sillicon、MOLD の素材を epoxyresion、BALL の素材を solder と定義しています。

<head_source>要素を使って 3 次元データ中の物体 (object) に熱源を設定することができます。下記にその例を示します。この例では 3 次元データ中の DIE オブジェクトに 0.5W の熱源を設定しています。

第 5 章

LPB Format 構文チェッカー

本章では LPB Format の構文チェッカを紹介します。

構文チェッカには LPB デザインキットに含まれる Syntax Checker と、(株) ジェム・デザイン・テクノロジーズが配布している GemCheck の 2 種類があります。共に無料で使用することができます。

5.1 LPB デザインキット Syntax Checker

LPB デザインキットは JEITA 半導体＆システム設計技術委員会が LPB Format の普及を目的として無償配布するものです。ツールのソースコードも公開されているので、LPB Format を利用するデザインフローや EDA を開発するときのサンプルとしても利用可能です。

本節ではデザインキットに含まれる構文チェッカ（Syntax Checker) の使用方法をを紹介します。

5.1.1 インストール

下記の URL からデザインキットをご請求ください。折り返しダウンロード用の URL をご連絡します。

http://jeita-sdtc.com/lpb-open-source-project/lpbdesignkit/

ダウンロードした.msi ファイルをダブルクリックし、指示に従って LPB DesignKit をインストールしてください。

5.1.2 アンインストール

LPB デザインキットをアンインストールする場合は、**Windows メニューを右クリック > アプリと機能**をクリック。インストールされているアプリの一覧表が出てくるので、**LPB-DesignKit** を選択し**アンインストール**をクリックします。

5.1.3 起動

Windows メニューから **LPB DesignKit > Symtax Checker - CFormat** を選択します。

5.1.4 画面の説明

下図にウインドウレイアウトを示します。

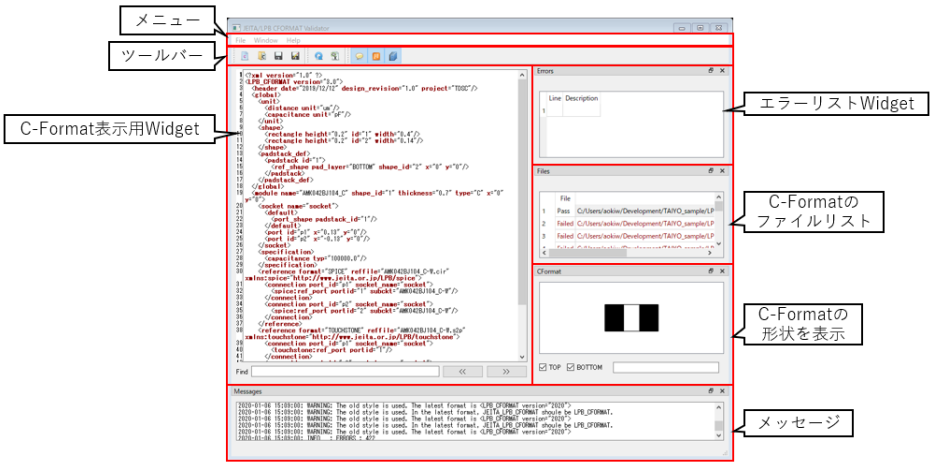


図 5.1　Window Layout

C-Format 表示用 Widget:

> チェックしている C-Format を表示します。キーワードは赤文字で表示されます。またエラーがある箇所には下線が引かれます。

エラーリスト Widget:

> エラーのある行番号とコメントが表示されます。

C-Format のファイルリスト:

> Format Validator は複数の C-Format ファイルを入力することができます。このリストで選択したファイルがチェックされます。

5.1.5 CFORMAT の構文チェック

File > Open CFORMAT... でチェックする C-Format ファイルを選択します。Shift キーで複数のファイルを選択することも可能です。

C-Format のファイルリスト に入力したファイルの一覧が表示されます。構文をチェックするファイルを選択してください。

C-Format 表示用 Widget にチェックしている C-Format の全文が表示されます。構文エラーがある場合は **エラーリスト** に、エラーのある行番号が表示されます。また **C-Format 表示用 Widget** でエラーがある箇所には下線が引かれます。

5.2 GemCheck によるフォーマットチェック

GemCheck は Gem Design Technologies (https://gemdt.com/) が無償公開している LPB Format verson3 (IEEE 2401-2019) 用のフォーマットチェッカです。Windows のコマンドプロンプトから起動して使用します。

5.2.1 機能・特徴

GemCheck は文法チェックに加え、データの参照関係のチェックまで行います。

5.2.2 ダウンロード・インストール

下記 URL からお申し込みください。ダウンロード用の URL とライセンスコードをご案内致します。

https://gemdt.com/products/freetools/

ダウンロードした zip ファイルには **gemcheck_xxx_install.exe** と、マニュアルが入っています（**xxx** は GenCheck のバージョン番号です）。**gemcheck_xxx_install.exe** をダブルクリックするとインストーラが起動します。デフォルトでは"C:\Program Files (x86)\Gem\GenCheckx.xx"にインストールされます。GemCheck をインストールしたフォルダ名は、**パスの設定**で使用しますので、メモしておいてください。

GemCheck の詳細は同封されているマニュアルを参照してください。

5.2.3 パスの設定

環境変数 Path に GemCheck がインストールされたフォルダ名を追加します。

Windows メニューを右クリック > システム > システム情報 > システムの詳細設定から設定画面を開き、**環境変数**ボタンをクリックします。システム環境変数の Path の最後に GemCheck をインストールしたフォルダ名を追加します。

5.2.4 アクティベート（ライセンス設定）

"C:\Users\uuuuu\AppData\Roaming\Gen"の下にお送りしたライセンスコードを記入した"gencheck.lic"というファイルを作ってください。このファイルが無いと GemCheck が起動しません。

5.2.5 使用方法

GemCheck はコマンドプロンプトから使用します。

Windows メニュ > Windows システムツール > コマンドプロンプト

でコマンドプロンプトを立ち上げます。

コマンド gemcheck に続けてチェックするファイル名を列挙して入力します。結果はコマンドウインドウ内に表示されます。

例えば、

gemcheck file1.xml file2.xml file3.xml file4.xml

は 4 つのファイルの構文をチェックします。

第 6 章

用語集・索引・検索

6.1 その他の用語

全ての用語については genindex から参照して下さい。

#.. glossary:

6.2 Todos

- *Todos*
- その他の用語
- genindex
- modindex
- search