



Pulsonix Design System

Library Integration Toolkit

Copyright Notice

Copyright © WestDev Ltd. 2000-2016

Pulsonix is a Trademark of WestDev Ltd. All rights reserved. E&OE

Copyright in the whole and every part of this software and manual belongs to WestDev Ltd. and may not be used, sold, transferred, copied or reproduced in whole or in part in any manner or in any media to any person, without the prior written consent of WestDev Ltd. If you use this manual you do so at your own risk and on the understanding that neither WestDev Ltd. nor associated companies shall be liable for any loss or damage of any kind.

WestDev Ltd. does not warrant that the software package will function properly in every hardware software environment.

Although WestDev Ltd. has tested the software and reviewed the documentation, WestDev Ltd. makes no warranty or representation, either express or implied, with respect to this software or documentation, their quality, performance, merchantability, or fitness for a particular purpose. This software and documentation are licensed 'as is', and you the licensee, by making use thereof, are assuming the entire risk as to their quality and performance.

In no event will WestDev Ltd. be liable for direct, indirect, special, incidental, or consequential damage arising out of the use or inability to use the software or documentation, even if advised of the possibility of such damages.

WestDev Ltd. reserves the right to alter, modify, correct and upgrade our software programs and publications without notice and without incurring liability.

Microsoft, Windows, Windows NT and Intellimouse are either registered trademarks or trademarks of Microsoft Corporation.

All other trademarks are acknowledged to their respective owners.

Pulsonix, a division of WestDev Ltd.

Printed in the UK.

Issue date: 16/02/16

Pulsonix

20 Miller Court
Severn Drive
Tewkesbury
Tewkesbury Business Park
Glos, GL20 8DN
United Kingdom

Phone +44 (0)1684 296 570

Fax +44 (0)1684 296 515

Email info@pulsonix.com

Web www.pulsonix.com

Contents

| | |
|---|-----------|
| CONTENTS..... | 5 |
| CHAPTER 1. GETTING STARTED | 7 |
| Introduction..... | 7 |
| The contents of the Toolkit..... | 7 |
| Licensing the Toolkit..... | 7 |
| Creating library Integration files | 7 |
| Using library integration files | 8 |
| CHAPTER 2. SYMBOL FILES | 9 |
| General commands..... | 9 |
| Header section commands | 9 |
| Symbol section commands | 9 |
| Generic commands..... | 10 |
| Segmented Rectangle symbols | 12 |
| CHAPTER 3. FOOTPRINT FILES | 13 |
| General commands..... | 13 |
| Header section commands | 13 |
| Footprint section commands | 13 |
| Generic commands..... | 14 |
| Axial footprints..... | 18 |
| PGA footprints..... | 18 |
| QUAD footprints | 18 |
| RADIAL footprints..... | 19 |
| Standard parameter types..... | 19 |
| CHAPTER 4. PART FILES..... | 21 |
| General commands..... | 21 |
| Header section commands | 21 |
| Part section commands..... | 21 |
| Finding or creating Parts | 21 |
| Other commands..... | 22 |
| APPENDIX A. SAMPLE FILES | 28 |
| Symbol Files | 28 |
| Footprint Files..... | 28 |
| Through-hole DIP..... | 28 |
| Surface-mount PLCC..... | 29 |
| Use of ADDPAD command | 30 |
| Parts Files | 30 |
| Normal use to create new parts..... | 30 |
| Use of ADDPINATTR, ADDUNGATEDPINS and ++ commands | 31 |
| Use of ADDPART command..... | 31 |
| Adding Alphanumeric Pins to Parts..... | 32 |

| | |
|---|-----------|
| APPENDIX B. CHANGE HISTORY | 33 |
| Issue 27-Aug-14 [Pulsonix 8.0]..... | 33 |
| Issue 03-May-13 [Pulsonix 8.0] | 33 |
| Issue 03-Sept-12 [Pulsonix 7.5] | 33 |
| Issue 20-Oct-10 [Pulsonix 6.1]..... | 33 |
| Issue 30-Apr-09 [Pulsonix 6.0] | 33 |
| Issue 20-Oct-08 [Pulsonix 5.1]..... | 34 |
| Issue 08-May-08 [Pulsonix 5.0]..... | 34 |
| Issue 27-Mar-08 | 34 |
| Issue 18-Feb-04 | 34 |
| Issue 01-Sep-03..... | 34 |
| Issue 02-June-03..... | 34 |
| Issue 26-Feb-03..... | 34 |
| Issue 09-Sep-02..... | 34 |
| Issue 21-Aug-02..... | 35 |
| Issue 06-Aug-02..... | 35 |
| INDEX | 37 |

Chapter 1. Getting Started

Introduction

The **Library Integration toolkit** provides you with the ability to create Pulsonix library (Schematic Symbol, PCB Footprint and Part) items from your own data sources. This might include, for example, importing data from another CAD system, or generating footprints from information retrieved from the Internet.

The files are of a fixed ASCII format as defined in this specification and are imported using a licensed interface mechanism.

The contents of the Toolkit

The Toolkit consists of:

- This documentation (the import file specification).
- Importers to read library integration files.

Licensing the Toolkit

The license for this option is shown in the **License Manager** dialog as **Library Integration Toolkit**. There is no charge for this option.

In addition, creation of **bond pads** and **wires** in footprints is controlled by the availability of another license, which appears in the **License Manager** dialog as **Chip Packaging Toolkit**. Without this license, bond pad and wire commands will be ignored.

Creating library Integration files

The commands available for use in a library integration file generally follow the 'controls' presented on the pages of the corresponding **Pulsonix Library Wizard**, and the resulting library items should correspond to those that can be generated using the same values in the wizard.

Integration files are processed sequentially (from start to finish), and – except for the **FORMAT command** - any of the commands listed in the reference sections may appear anywhere in the file. However, for ease of understanding the commands are separated into those that generally only appear at the start of the file and those that are used to generate the actual footprints.

The positioning and ordering of commands in the file is not generally critical. Any side-effects for particular commands are mentioned in the description for those commands.

Each line of a integration file contains a keyword, followed by one or more parameters (depending on the keyword), separated from each other by commas. Blank lines, and

any lines starting with semi-colon, are ignored. Text (e.g. name) fields may be enclosed in double quotes if desired.

Some commands are applicable only in certain contexts, for example only for certain footprint types. The toolkit does not context-check the commands, so specifying commands that are not applicable may either have no effect, or cause unpredictable results. If commands are specified as they would be entered interactively through the corresponding Wizard, then this should give the desired result. The same applies to range checking, for example on the number of pads on a footprint.

Using library integration files

Library integration files are loaded into Pulsonix through the **Import** option on the relevant page of the **Libraries** dialog.

For example, to import a footprint integration file, open the **Libraries** dialog and go to the **Footprints** tab. Choose the library to which the generated footprints should be written. Click the **Import** button, browse to the desired integration file, and click on **Open**. The file will be loaded and processed to generate the specified footprints.

Chapter 2. Symbol Files

General commands

Text_here

The hash (#) character can be used at the start of any new line to write a comment. It should be added to every line which needs a comment. It can be used at any point in the SGF file. Comments can also be made using the semi-colon character ; at the start of a line.

EXIT

The EXIT command can be placed anywhere in the format file to stop the import of the file. You may wish to use the command during a multi-item import to debug the items above it or to restrict import to a particular number of items.

Header section commands

FORMAT,SGF,1

The first line of the file should contain this command, which specifies that this is a symbol library toolkit file, and also gives the format version number (currently version 1).

FILE,filename

Include another integration file. Each line in the specified file is processed, then processing of the current file resumes.

TYPE,symbol_type

Specify the type of symbol to create. These values correspond to those available interactively through the Symbol Wizard.

symbol_type := RECT | SEG | TRI

UNITS,unit,factor,precision

Co-ordinate units.

| | | |
|---------|---|------------|
| unit := | 0 | inch |
| | 1 | thou |
| | 2 | metre |
| | 3 | millimetre |
| | 4 | micron |
| | 5 | centimetre |

factor := units per base units, generally always 1

precision := number of decimal places

Symbol section commands

The commands listed here are divided into several sections. Many commands (listed under ‘Generic commands’) are applicable to many or all symbol types, whilst others only really apply to particular types.

Generic commands

- NAME**,symbol_name
The name by which the symbol is to be known. This command is the one that triggers generation of the symbol and writing it to the library, so positioning of this command in the file is critical to achieving the desired symbols.
- BOTTOMPINDIRLEFT**,bool
True if the pins on the bottom side are numbered leftwards.
- BOTTOMPINORDER**,int
The ‘side order’ for numbering pins on the bottom side of the symbol.
- BOTTOMPINS**,count
The number of pins on the bottom side.
- CLKPIN**,size
The size of the clock pin marker.
- INVPIN**,size
The size of the inverted (not) pin marker.
- LEFTPINDIRUP**,bool
True if the pins on the left side are numbered upwards.
- LEFTPINORDER**,int
The ‘side order’ for numbering pins on the left side of the symbol.
- LEFTPINS**,count
The number of pins on the left side.
- LEGLEN**,size
The length of the terminal “legs”.
- MARKPIN**,pin_number[,type][,CLK]
Adds ‘special’ marker(s) to specified pin. The available types correspond to those available interactively in the Symbol Wizard. Including the optional field “CLK” will also include a Clock pin marker inside the symbol outline.
- | | |
|------------|-----------------------|
| pin_number | pin (terminal) number |
| type | POLIN POLOUT INV |
- NAMEPOS**,name_pos
Specify the position for the component name.
- | | | |
|----------|---|--------------|
| name_pos | 0 | bottom left |
| | 1 | bottom right |
| | 2 | top right |
| | 3 | top left |
| | 4 | centre |
- NUMBERSOUTSIDE**,bool
Whether or not pin numbers should be placed ‘outside’ the pins, away from the body of the symbol.
- ORIGIN**,origin_pos
The position of the footprint (symbol) origin.

| | | |
|------------|---|--------|
| origin_pos | 0 | centre |
| | 1 | pin 1 |

SPICENAMEPOS,name_pos

Specify the position for the Spice model name. Values are the same as for the NAMEPOS command.

PADDIM3,size

The size of the third size parameter required for some pad style shapes. This is the thickness for target or cross, the inner size for annulus, and the corner radius for rounded rectangle.

PADFILLED,bool.

Specify whether the pad style for terminals is to be 'filled'.

PADLENGTH,size

The length for some pad style shapes (rectangle, rounded rectangle, triangle, oval, bullet, pointer, diamond).

PADPITCH,size

The size of the step between adjacent pads.

PADSHAPE,pad_shape

The shape of the pads. Note that presently only some of the possible pad shapes are supported by the toolkit.

pad_shape := DIAMOND | OVAL | RECT | ROUND | ROUNDRECT | SQUARE | CROSS | CHAMFERRECT | BULLET | TRIANGLE | PLUS

PADSIZE,size

The width of pads.

PADSTYLE,style_name

The name to use for the pad style.

POLPIN,size

The size of the polarity in/out pin marker.

RIGHTPINDIRUP,bool

True if the pins on the right side are numbered upwards.

RIGHTPINORDER,int

The 'side order' for numbering pins on the right side of the symbol.

RIGHTPINS,count

The number of pins on the right side.

SHOWLOGICNAMES,bool

Whether or not to set logic names to be visible.

SHOWPINNAMES,bool

Whether or not to set pin names to be visible.

SYMWIDTH,size

The width of the symbol outline ("body").

12 Symbol Files

TOPPINDIRLEFT,bool

True if the pins on the top side are numbered leftwards.

TOPPINORDER,int

The 'side order' for numbering pins on the top side of the symbol.

TOPPINS,count

The number of pins on the top side.

Segmented Rectangle symbols

SEGMENTS,count

Number of segments.

SEGLEFTPINS,count

Number of pins on the left of each segment.

SEGRIGHTPINS,count

Number of pins on the right of each segment.

Chapter 3. Footprint Files

General commands

Text_here

The hash (#) character can be used at the start of any new line to write a comment. It should be added to every line which needs a comment. It can be used at any point in the FGF file. Comments can also be made using the semi-colon character ; at the start of a line.

EXIT

The EXIT command can be placed anywhere in the format file to stop the import of the file. You may wish to use the command during a multi-item import to debug the items above it or to restrict import to a particular number of items.

Header section commands

FORMAT,FGF,1

The first line of the file should contain this command, which specifies that this is a footprint library toolkit file, and also gives the format version number (currently version 1).

TYPE,footprint_type

Specify the type of footprint to create.

footprint_type := AXIAL | DIP | PGA | QUAD | RADIAL

UNITS,unit,factor,precision

Co-ordinate units.

| | | |
|---------|---|------------|
| unit := | 0 | inch |
| | 1 | thou |
| | 2 | metre |
| | 3 | millimetre |
| | 4 | micron |
| | 5 | centimetre |

factor := units per base units, generally always 1

precision := number of decimal places

FILE,filename

Include another footprint integration file. Each line in the specified file is processed, then processing of the current file resumes.

Footprint section commands

The commands listed here are divided into several sections. Many commands (listed under ‘Generic commands’) are applicable to many or all footprint types, whilst others only really apply to particular types.

14 Footprint Files

Generic commands

ADDBONDPAD,x,y,angle

Add a bond pad in a particular position. This command allows you to create bond pads and wires. The co-ordinates supplied are relative to the position of the symbol origin. Each bond pad created (using the bond pad style, see BONDPAD* commands) will be wired to the corresponding normal (die) pad using the specified wire style (see WIRE* commands).

Note: This list of bond pads is only applied to the current footprint; the list is deleted as soon as the current footprint has been generated.

ADDBONDPADREL,dx,dy,angle

Add a bond pad in a particular position relative to the previous bond pad added.

ADDBONDPADSET,count,sx,sy,dx,dy,angle

Add a number of bond pads in specified positions, the equivalent of repeated calls to ADDBONDPAD command.

count := how many bond pads to add
sx,sy := co-ordinates of first bond pad
dx,dy := delta “step” between each pad in x & y
angle := pad angle

ADDBONDPADSETREL,count,dx,dy,angle

Add a number of bond pads in specified positions relative to the previous bond pad added, the equivalent of repeated calls to ADDBONDPADREL command.

count := how many bond pads to add
dx,dy := delta “step” between each pad in x & y
angle := pad angle

ADDPAD,x,y

Add a pad in a particular position. This command allows you to specify pad positioning that the footprint integration would not normally create. The co-ordinates supplied are relative to the position of the symbol origin.

Note: This list of ‘irregular’ pads is used in place of (not as well as) the default pads for the footprint type. Also, it is only applied to the current footprint; the list is deleted as soon as the current footprint has been generated.

ADDPADREL,dx,dy

Add a pad in a particular position relative to the previous pad added.

ADDPADSET,count,sx,sy,dx,dy

Add a number of pads in specified positions, the equivalent of repeated calls to ADDPAD command.

count := how many pads to add
sx,sy := co-ordinates of first pad
dx,dy := delta “step” between each pad in x & y

ADDPADSETREL,count,dx,dy

Add a number of pads in specified positions relative to the previous pad added, the equivalent of repeated calls to ADDPADREL command.

count := how many pads to add
 dx,dy := delta “step” between each pad in x & y

BONDPADDRILL,size
 Bond pad drill diameter.

BONDPADLAYER,layer_name
 Specify the name of the layer on which bond pads should be created. The value for this field is generally one of:
 <Through Board>
 <Top Side>
 <Bottom Side>

BONDPADLENGTH,size
 The length for some pad style shapes (rectangle, rounded rectangle, triangle, oval, bullet, pointer, diamond).

BONDPADSHAPE,pad_shape
 The shape of the bond pads. Note that presently only some of the possible pad shapes are supported by the toolkit.
 pad_shape := DIAMOND | OVAL | RECT | ROUND | ROUNDRECT | SQUARE | CROSS | CHAMFERRECT | BULLET | TRIANGLE | PLUS

BONDPADSIZE,size
 The width of bond pads.

BONDPADSTYLE,style_name
 The name to use for the bond pad style.

DRILL,size
 Pad drill diameter.

FIRSTPIN,pin_num_pos
 Specify the position for the first pin number.

| | | |
|----------------|----|---------------|
| pin_num_pos := | 0 | bottom left |
| | 1 | bottom middle |
| | 2 | bottom right |
| | 3 | right bottom |
| | 4 | right middle |
| | 5 | right top |
| | 6 | top right |
| | 7 | top middle |
| | 8 | top left |
| | 9 | left top |
| | 10 | left middle |
| | 11 | left bottom |

NAME,footprint_name
 The name by which the footprint is to be known. This command is the one that triggers generation of the footprint and writing it to the library, so positioning of this command in the file is critical to achieving the desired footprints.

NAMEPOS,name_pos

Specify the position for the component name.

| | | |
|----------|---|--------------|
| name_pos | 0 | bottom left |
| | 1 | bottom right |
| | 2 | top right |
| | 3 | top left |
| | 4 | centre |

ORIGIN,origin_pos

The position of the footprint (symbol) origin.

| | | |
|------------|---|--------|
| origin_pos | 0 | centre |
| | 1 | pin 1 |

PADCOUNT,count

The number of pads in the footprint.

PADDIM3,size

The size of the third size parameter required for some pad style shapes. This is the thickness for target, the inner size for annulus, and the corner radius for rounded rectangle.

PADLAYER,layer_name

Specify the name of the layer on which pads should be created. The value for this field is generally one of:

<Through Board>
<Top Side>
<Bottom Side>

unless creating die pads for die/bond wiring, in which case this should specify the appropriate wire layer name from your technology file (e.g. Wires Top).

PADLENGTH,size

The length for some pad style shapes (rectangle, rounded rectangle, triangle, oval, bullet, pointer, diamond).

PADPITCH,size

The size of the step between adjacent pads.

PADSHAPE,pad_shape

The shape of the pads. Note that presently only some of the possible pad shapes are supported by the toolkit.

pad_shape := DIAMOND | OVAL | RECT | ROUND | ROUNDRECT |
SQUARE | CROSS | CHAMFERRECT | BULLET | TRIANGLE | PLUS

PAD1SHAPE,bool,pad_shape

If true, the first pad on the footprint is created using the specified pad shape. Otherwise, all pads are created using the same basic pad shape.

PADSIZE,size

The width of pads.

PADSTYLE,style_name

The name to use for the pad style.

PINORDER,pin_order

The order in which pins should be numbered.

| | | |
|-----------|---|-------------------|
| pin_order | 0 | clockwise |
| | 1 | counter-clockwise |

PLACEBOX,llx,lly,urx,ury

Specify the location of the enclosing box of the placement shape. This box is discarded after each footprint is generated. The co-ordinates supplied are relative to the position of the symbol origin.

PLACECREATE,bool

Whether or not to generate a placement shape.

PLACEGAP,size

The gap between the pad edges and the placement shape.

PLACEISAREA,bool

Whether or not the placement shape is to be created as an Area.

PLACELAYER,layer_name

The name of the layer on which the placement shape is to be added.

ROWPITCH,size

The distance between rows of pads.

Note: Setting this value also sets the VERTROWPITCH parameter to this size. To specify horizontal and vertical row pitches separately, the HORIZROWPITCH and VERTROWPITCH commands should be used.

SHOWPINNAMES,bool

Whether or not to set pin names to be visible.

SILKCREATE,bool

Whether or not a silkscreen shape should be created.

SILKDOT,bool

Whether or not to add a 'dot' at pad 1 on the silkscreen.

SILKGAP,size

The gap between the pad edges and the silkscreen shape.

SILK_LAYER,layer_name

The name of the layer on which the silkscreen shape should be created.

SILKMITRE,bool

Whether or not the first corner of the silkscreen shape should be mitred.

SILKMITRESIZE,size

The size of the silkscreen corner mitre.

SILKNOTCH,bool

Whether or not the end of the silkscreen shape should be mitred.

SILKNOTCHSIZE,size

The size of the silkscreen notch.

18 Footprint Files

SILKOUTSIDE,bool
Indicates whether the silkscreen shape is outside the pads or inside.

WIRELAYER,layer_name
The name of the layer on which bond wires should be created.

WIRESIZE,size
The size of the line style used for bond wires.

WIRESTYLE,style-name
The name of the line style for bond wires.

Axial footprints

AXIALLEGS,bool
Specify whether or not 'legs' are to be drawn between the component outline 'box' and the pads.

BOXLENGTH,x
Horizontal length of component outline box.

BOXROUND,bool
If true, the component outline will have semi-circular ends instead of flat.

BOXWIDTH,y
Vertical size of component outline box.

PGA footprints

COLCOUNT,count
The number of columns.

ROWCOUNT,count
The number of rows.

STAGGERODD,bool
Whether or not odd rows of the footprint are 'staggered'.

STAGGERPITCH,size
The amount by which alternate rows of the footprint are staggered.

QUAD footprints

HORIZPADCOUNT,count
The number of pads along the horizontal sides.

HORIZROWPITCH,size
The distance between the rows of pads on horizontal sides of the footprint.

VERTPADCOUNT,count
The number of pads along the vertical sides.

VERTROWPITCH,size
The distance between the rows of pads on vertical sides of the footprint.

RADIAL footprints

DIAMETER,size

The radius of the component outline shape.

RADIUS,size

The radius of the component outline shape. This command is provided as an alternative to the DIAMETER command; both commands affect the same underlying value.

STARTANGLE,angle

The angle at which the first pin is positioned relative to the centre of the footprint.

TABANGLE,angle

The angle of the 'tab' on the silkscreen shape.

TABSIZE,size

The size of the 'tab' on the silkscreen shape.

Standard parameter types

angle

Angle as a floating-point number.

bool

Boolean true/false value. Specify Y or y for true, any other value for false.

count

Number of items as a positive integer, generally greater than zero.

size

Item size, width or diameter in current units.

x

X-coordinate in current units.

y

Y-coordinate in current units.

Chapter 4. Part Files

General commands

Text_here

The hash (#) character can be used at the start of any new line to write a comment. It should be added to every line which needs a comment. It can be used at any point in the PGF file. Comments can also be made using the semi-colon character ; at the start of a line.

EXIT

The EXIT command can be placed anywhere in the format file to stop the import of the file. You may wish to use the command during a multi-item import to debug the items above it or to restrict import to a particular number of items.

Header section commands

FORMAT,PGF,3

The first line of the file should contain this command, which specifies that this is a part library toolkit file, and also gives the format version number.

The current version number is version 3.

IMPORTANT: for format 3 and later, new parts will default to being added together into the same group as they are created. To prevent this, either use format 2, or add the command PARTGROUP,NONE near the start of the file.

FILE,filename

Include another integration file. Each line in the specified file is processed, then processing of the current file resumes.

Part section commands

Finding or creating Parts

ASSOCPART,name[,desc[,family[,namestem]]]

Generate a new Associated part using the supplied parameters.

| | |
|----------|---------------------------|
| name := | part name |
| desc := | optional part description |
| family:= | optional part family |

FINDPART,name

Find and open an existing part using the supplied name. This is now available for edit (adding attributes to for example).

| | |
|---------|-----------|
| name := | part name |
|---------|-----------|

NEWPART,name,desc,mfr,scmsym,[scmsympin],footprint,num_pins[,pin_num_list]
Generate a new part using the supplied parameters.

If specifying pin number entries for ‘multiply mapped’ pins, the entry should be enclosed in double-quotes, for example: 1,2,3,”4,5,6”,10,9,8,7

| | |
|-----------------|---|
| name := | part name |
| desc := | optional part description |
| mfr := | optional manufacturer name |
| scmsym := | the name of the schematic symbol |
| scmsympins:= | optional, the number of schematic symbol pins |
| footprint := | the name of the pcb footprint |
| num_pins := | the count of the number of pins |
| pin_num_list := | optional comma-separated list of pin numbers, this can be a numeric or alphanumeric list. |

PART,name,desc,mfr,footprint,num_pins[,pin_num_list]
Generate a new part using the supplied parameters, without adding any gates to the part.

| | |
|-----------------|---|
| name := | part name |
| desc := | optional part description |
| mfr := | optional manufacturer name |
| footprint := | the name of the pcb footprint |
| num_pins := | the count of the number of pins |
| pin_num_list := | optional comma-separated list of pin numbers, this can be a numeric or alphanumeric list. |

Other commands

ADDATTR,name,value

Add an attribute to the previously generated part. This command must come after a **NEWPART**, **FINDPART** or **ADDPART** command so that a ‘current part’ is available to which the attribute can be added.

The following ‘special’ attribute names are recognised and will be added to the part as the equivalent built-in attributes:

3DPACKAGE
FAMILY
HEIGHT
HYPERLINK
SPICEDEVICE
SPICEVALUE
SPICETEMPLATE
SPICEFUNCTION
SPICEPARAMETERS
SPICEDEFINITION
SPICETOLERANCE

Use **ADDATTR**,**FAMILY**,name to define the Part Family name on the Part.
Use **ADDATTR**,description,description_here to overwrite the Description field added to a Part.

Use ADDATTR,Mfr,manufacturer_name to overwrite the Mfr field added to a Part.

When using HYPERLINK,value this will replace the built-in <Hyperlink> attribute field. You can also specify it in brackets < > to replace the attribute value. This also applies to the built-in attribute <Component Height> attribute and HEIGHT.

ADDGATE,scmsym,num_pins[,pin_num_list]

Add a gate to the current part.

If specifying pin number entries for 'multiply mapped' pins, the entry should be enclosed in double-quotes, for example: 1,2,3,"4,5,6",10,9,8,7

| | |
|-----------------|---|
| scmsym := | the name of the schematic symbol |
| num_pins := | the count of the number of pins on the symbol |
| pin_num_list := | optional comma-separated list of pin numbers |

ADDPART,name

Generate another part using the given part name, copying all remaining parameters from the previously generated part. This command must come after a NEWPART or ADDPART command so that a 'current part' is available for the part details.

ADDPARTREP,partrep_name[,scmsym,num_pins[,pin_num_list]]

Add a part representation and (optionally) its first gate to the current part. Leave the partrep_name blank to get the current part representation name as defined by the PARTREP command. If not blank, this will update the value of the current part representation name.

If specifying pin number entries for 'multiply mapped' pins, the entry should be enclosed in double-quotes, for example: 1,2,3,"4,5,6",10,9,8,7

| | |
|-----------------|---|
| partrep_name := | the name of the part representation |
| scmsym := | the name of the schematic symbol |
| num_pins := | the count of the number of pins on the symbol |
| pin_num_list := | optional comma-separated list of pin numbers |

ADDPINATTR, pin_name, pin_attr=attr_value [, pin_attr=attr_value ...]

Add an attribute or property to a part pin.

| | |
|-------------|---|
| pin_name := | identifies the pin number to which attribute applies. If this is * then attribute is applied to all pins that don't have any attributes, logic name or pintype. |
|-------------|---|

| | |
|-------------|---|
| pin_attr := | pin attribute or property name. Pre-defined names are: LOGICNAME, NETNAME, PINTYPE (see below) and PINSWAP. Other names are assumed to be user-defined attributes to be added to the pin. Note: SWAPGROUP has |
|-------------|---|

been superseded by the PINSWAP command but is still supported in existing Library Toolkit files.
 attr_value := attribute or property value

For PINTYPE, the values for attr_value are:

- UNDEFINED
- OUTPUT
- INPUT
- BI-DIRECTIONAL
- OPEN COLLECTOR
- OR-TIEABLE
- TRI-STATE
- TERMINATOR
- POWER
- GROUND
- OPEN EMITTER
- NO CONNECT
- SOURCE
- LOAD
- PASSIVE
- PCB CONNECT

ADDUNGATEDPINS, pin_type, net, pin_name [, pin_name ...]

Add one or more ungated power/ground pins to a part.

- pin_type := see list of pintypes under ADDPINATTR
- net_name := name of the net to apply to the pins
- pin_name := identifies the pin to add

ALLOWADDPINS, bool

Specifies whether the ADDPINATTR command should add new (ungated) pins for any pin names specified that don't already exist on the part. Default value is True.

AUTONAME,separator,stem,attr_name[,attr_name...]

Define the rules for auto-naming parts based on their attribute values.

- separator := text to separate each part of name
- stem := text to place at start of each name
- attr_name := name of attribute holding value to add to name

To stop autonaming (i.e. to revert to 'manually' named parts), include an AUTONAME command with NONE as the only parameter, like this:

AUTONAME,NONE

AUTOPART,attr_name=attr_value [, attr_name=attr_value]

Create an auto-named part (using the name rules specified by the preceding AUTONAME command), adding the given attributes to the part as it is created.

CONNECTOR, bool [, sym_name]

Specifies whether parts created using PART command are connector parts. If True, sym_name should be included to specify the name of the appropriate schematic symbol. Default value is False.

COPYPART, name, newname [, description [, attr_name = attr_value ...]
 Creates a copy of an existing part to a new name.
 name := name of existing part
 newname := name of new part
 description := part description (optional)
 attr_name = attr_value := one or more pairs of attribute settings

DELATTR, attr_name
 Deletes the named attribute from the current part (as located using the **FINDPART** command).

GATESWAP, [partrep], swap_group , gate_name
 partrep := Part Representation name (optional)
 swap_group := letter or number to represent the swap group
 gate_name := the Gate letter to add the swap group to

Use this command after each gate. It uses the Gate letter to identify the gate, not the symbol name.

MAKEGROUP,name [, name ...]
 Groups all the named parts together into a single part group. If you name a part that is already in a group, then all parts listed for this command will be added to that group. Parts with different pincounts will not be added to the group, and neither will parts that already belong to a second group if you first name a part that is already in a different group.

name := name of existing part

NAMESTEM,name
 Sets the default component name stem to be used when adding parts.

PARTGROUP,group_name
 Sets the name of the group to which parts will belong. This name will be used for all subsequent parts that are added using any of the commands that create parts. To reset to the default behaviour (each part in its own group), use **NONE** for group_name.

PARTREP,rep_name
 Sets the name of the current Part Representation. As new parts are created they will have this name applied to their first representation.
 Also sets the 'context' in which other part commands operate when manipulating existing parts found using the **FINDPART** command. When an existing part is identified using **FINDPART**, various commands such as **ADDGATE** will operate on the named part representation.

To reset to the default behaviour (no specified part representation), use **NONE** for the rep_name. Then all commands will operate on the default (or only) part representation.

RENAMEPART, oldname [, oldname ...], newname

IMPORTANT: this command acts directly on the currently selected library.

Also, as this command can remove parts from your library without asking for confirmation, it is recommended that you take a backup copy of your library first just in case the results are not what you expect.

Renames the specified existing part(s) to a new part name. If the new part name already exists (as it will always do if you supply more than one oldname in the command), the list of footprints on the old part will be added to the list of footprints on the new part, and the old part will be deleted. This will therefore act as a 'merge' command.

You can also use wildcards in the oldname fields, for example:

RENAMEPART, 1466*, AMX-1466

will end up with a single part called AMX-1466 that is the same as the first 1466* part found in the library, but with the footprints of the following 1466* parts added to its footprint list.

oldname := existing part name

newname := new part name

RENAMEPARTMATCHATTRIBUTE, attrname

This command is used when applying a PGF file containing RENAMEPART commands to a Design confirmation, and is used to alter the value of the stated attribute at the same time as altering the part name.

This can be used for example if you have an attribute that contains the part number (perhaps for the Pulsonix Database Connection) that has to have a value that is the same as the partname. As a part is found and renamed by the RENAMEPART command, the specified attribute will have its value changed to match.

SETDESCRIPTION, name, string

Sets the description of the specified part to the given string.

name := part name

SETFOOTPRINT, name, footprint_name [, footprint_name ...]

Replaces the list of footprints referenced by the specified part. Any existing footprint names on the part will be removed.

name := part name

footprint_name ::= name of pcb footprint

SETGATE, name, gate_name, pin_name, logic_name [, pin_name, logic_name ...]

Replaces the contents of the specified gate on an existing part. The pin name and logic name pairs are applied in schematic symbol pin order, i.e. the first pair apply to symbol pin 1, the second pair to pin 2, etc.

name := part name

gate_name ::= name of gate (e.g. A, B, C)

pin_name ::= pin name to apply to gate pin

logic_name ::= logic name to apply to gate pin

SETGATEPINS, name, ...

Same as SETGATE but doesn't remove existing pin details.

SETGATESYMBOLS, name, part_rep_name, gate_name, sym_name [, gate_name...]

Replaces the symbol(s) used by the specified part.

| | |
|------------------|-----------------------------|
| name := | part name |
| part_rep_name := | name of part representation |
| gate_name := | name of gate (e.g. A, B, C) |
| sym_name := | symbol name |

SETSTEM, [name,] string

Changes the default component name stem of the specified part (or if name not supplied then the current part as found by FINDPART) to the given string. This command can be applied to normal or connector parts.

| | |
|-----------|---|
| name := | part name |
| string := | stem letter(s) to be used, U, R, C, PL, IC etc. |

SETSYPHMBOL, [name,] sym_name

Changes the symbol used by the specified part (or if name not supplied then the current part as found by FINDPART) to the given symbol. This command can be applied to normal or connector parts.

| | |
|-------------|-------------|
| name := | part name |
| sym_name := | symbol name |

++

When this command is added to the end of a line, the line is "joined" to the following line to form a single command. This allows very long commands to be split across multiple lines in the input file to make it easier to read. See the Part section of the Appendix A for an example of how to use this command.

Appendix A. Sample Files

Symbol Files

| | |
|--------------------|-------------------------------------|
| FORMAT,SGF,1 | |
| TYPE,RECT | Rectangular type |
| UNITS,1,1,2 | |
| | |
| LINESTYLE,Normal,8 | |
| | |
| PADSTYLE,Cross | Use a standard 'cross' |
| PADSHAPE,Cross, | for the terminals |
| PADSIZE,60 | |
| PADDIM3,5 | |
| | |
| CLKPIN,30 | Size of clock, |
| POLPIN,40 | polarity |
| INVPIN,60 | and inverter markers |
| NAMEPOS,0 | |
| SPICENAMEPOS,0 | |
| DISTPINS,200 | |
| SYMWIDTH,1200 | |
| LEGLEN,200 | |
| LOGICNAMES,Y | |
| NUMBERSOUTSIDE,Y | |
| | |
| LEFTPINS,10 | 10 pins on left side |
| RIGHTPINS,8 | 8 on right |
| TOPPINS,0 | With none on top |
| BOTTOMPINS,0 | or bottom |
| LEFTPINORDER,1 | Make sure they're ordered |
| LEFTPINDIRUP,N | in the right direction |
| RIGHTPINORDER,2 | |
| RIGHTPINDIRUP,Y | |
| MARKPIN,1,CLK | Put clock marker on first pin |
| MARKPIN,10,INV | And invert the 10 th one |
| NAME,ATMEL20 | |
| | |
| LEFTPINS,12 | |
| RIGHTPINS,10 | |
| MARKPIN,1,CLK | |
| NAME,ATMEL24 | |

Footprint Files

Through-hole DIP

| | |
|--------------------------|---------------------------|
| FORMAT,FGF,1 | Always first command |
| UNITS,1,1,2 | Thou |
| TYPE,DIP | |
| | |
| PADLAYER,<Through Board> | Make pads through-hole |
| DRILL,32 | Drill size 32 thou |
| PADSHAPE,ROUND | Round pads |
| PAD1SHAPE,Y,SQUARE | Pad 1 is square instead |
| PADSIZE,55 | 55 thou pad diameter |
| PADSTYLE,Pad1 | Call the pad style "Pad1" |
| ORIGIN,1 | Symbol origin at pin 1 |

```

SILKCREATE,Y          Create silkscreen
SILKNOTCH,Y          Include a notch
SILKDOT,N            But no dot
SILKGAP,25           25 thou outside pads
SILKNOTCHSIZE,25     Size of notch
SILKLAYER,"Top Silkscreen" Name of layer for silkscreen

PLACECREATE,Y        Also create placement shape
PLACEGAP,25

;-----
; Standard DIPS

PADPITCH,100         100 thou between pads
ROWPITCH,300         300 thou across rows

PADCOUNT,6          6 pads
NAME,DIP6            Generate "DIP6" in library
PADCOUNT,8          8 pads
NAME,DIP8            Generate "DIP8"
                    Etc

PADCOUNT,14
NAME,DIP14
PADCOUNT,16
NAME,DIP16
PADCOUNT,18
NAME,DIP18

ROWPITCH,600         Now change row pitch
                    Other params stay the same
PADCOUNT,14
NAME,DIP14W
PADCOUNT,16
NAME,DIP16W
PADCOUNT,18
NAME,DIP18W

```

Surface-mount PLCC

```

FORMAT,FGF,1         Always first command
UNITS,3,1,2          Units are mm
PADLAYER,<Top Side>  Pads on top side only
DRILL,0              No drill
ORIGIN,0             Origin in footprint centre

SILKCREATE,Y        Create silkscreen
SILKMITRE,Y         With a mitred corner
SILKNOTCH,Y
SILKDOT,N
SILKGAP,1           1mm gap

PLACECREATE,Y        Also create placement shape
PLACEGAP,1.1        Slightly larger

;-----
; PLCC
TYPE,QUAD            Quad components
PADDIM3,0.1         0.1mm rounded pad corners
PADSHAPE,ROUNDRECT
PINORDER,1          Number pins anti-clockwise
PADPITCH,1.27       1.27 mm between pads
PADSIZE,0.55        Pad size 0.55mm
PADLENGTH,2.2       And length 2.2mm

```

30 Examples

```
PADCOUNT,20
ROWPITCH,8.6
NAME,PLCC20
PADCOUNT,28
ROWPITCH,11.2
NAME,PLCC28
PADCOUNT,44
ROWPITCH,16.2
NAME,PLCC44
PADCOUNT,52
ROWPITCH,18.8
NAME,PLCC52
```

20-pin footprint
8.6mm across rows
Generate PLCC20
etc...

Use of ADDPAD command

```
FORMAT,FGF,1
TYPE,DIP
UNITS,3,1,2
ORIGIN,1
SILKCREATE,Y
SILKNOTCH,Y
SILKDOT,N
SILKGAP,0.5
PADSHAPE,ROUND
PADSIZE,1.5
PADLAYER,<Through Board>
DRILL,0.7

ADDPAD,4,4
ADDPAD,0,0
PLACEBOX,-2,-2,9,5
NAME,ODD2
```

Parts Files

Normal use to create new parts

```
FORMAT,PGF,3

; create three 'normal' capacitor parts
NEWPART,C0402,Capacitor,Ceramic Surface Mount,C,C0402,2
NEWPART,C0805,Capacitor,Ceramic Surface Mount,C,C0805,2
NEWPART,C1206,Capacitor,Ceramic Surface Mount,C,C1206,2

; create new part group with parts auto-named using attributes
PARTGROUP,CERCAP
AUTONAME,"-/-",C,Value,L
NEWPART,C,Capacitor,Generic,C,C1206,2
ADDATTR,Value,1.0k
AUTOPART,Value=2.2k,L=1.6
AUTOPART,Value=3.3k,L=1.6
AUTOPART,Value=4.7k,L=1.6
AUTONAME,NONE
PARTGROUP,NONE

; create a part with specific pin name order and some pin attributes
NAMESTEM,RX
NEWPART,RX,,,ADD,DIP6,6,1,3,5,2,4,6
ADDATTR,VALUE,2
ADDPINATTR,1,NETNAME=VSS,LOGICNAME=VSS,PINTYPE=POWER
ADDPINATTR,2,LOGICNAME=CLK,PINTYPE=SOURCE
```

```

ADDPINATTR,3,PINTYPE=Tri-State
ADDPINATTR,2,PINSWAP=1
ADDPINATTR,4,PINSWAP=1
ADDPINATTR,6,PINSWAP=1
ADDPINATTR,5,LOGICNAME=DRIVER,PINTYPE=SOURCE

; Create a multi-gate part with a power gate
PART,MY74,,DI14,14
ADDGATE,AND2,3,1,2,3
ADDGATE,AND2,3,4,5,6
ADDGATE,AND2,3,9,8,10
ADDGATE,AND2,3,12,11,13
GATESWAP,,1,A,B,C,D
ADDGATE,PWR2,2,7,14
GATESWAP,,2,E
ADDPINATTR,7,LOGICNAME=GND,PINTYPE=GROUND,NETNAME=GND
ADDPINATTR,14,LOGICNAME=VCC,PINTYPE=POWER,NETNAME=VCC

```

Use of ADDPINATTR, ADDUNGATEDPINS and ++ commands

```

FORMAT,PGF,3
PART,EP600IPC-45,CLASSIC EP,Altera,DIP24,24
ADDATTR,Category,FPGA/CLASSIC EP
ADDGATE,ALTERA-1,22,1,13,23,14,2,11,3,4,++
5,6,7,8,9,10,22,21,20,19,18,17,16,15,22,++
1,13,23,14,2,11,3,4,5,6,7,8,9,10,22,21,20,19,++
18,17,16,15,22,1,13,23,14,2,11,3,4,5,6,7,8,9,10,++
22,21,20,19,18,17,16,11,22,1,13,23,14,2,11,3,4,5,6,++
7,8,9,10,22,21,20,19,18,17,16,15

ADDPINATTR,1,LOGICNAME=CLK1,PINTYPE=INPUT
ADDPINATTR,13,LOGICNAME=CLK2,PINTYPE=LOAD
ADDPINATTR,23,LOGICNAME=IN,PINTYPE=LOAD
ADDPINATTR,14,LOGICNAME=IN,PINTYPE=LOAD
ADDPINATTR,2,LOGICNAME=IN,PINTYPE=LOAD
ADDPINATTR,11,LOGICNAME=IN,PINTYPE=LOAD
ADDUNGATEDPINS,GROUND,GND,12
ADDUNGATEDPINS,POWER,VCC,24
ADDPINATTR,*,LOGICNAME=I/O,PINTYPE=BI-DIRECTIONAL

```

Use of ADDPART command

```

FORMAT,PGF,3
NAMESTEM,L
; create a 'normal' part
NEWPART,LQW04AN1N1C00p,RF Inductor wire wound type,Murata,L,L0804,2
ADDATTR,Category, Inductor/Murata LQW04AN_00
ADDATTR,Family,Murata LQW04AN_00 Series
ADDATTR,<Hyperlink>,www.Murata.com
ADDATTR,<Component Height>,0.4mm
ADDATTR,IndTestFreq,100MHz
ADDATTR,QTestFreq,250MHz
ADDATTR,Qmin,15
ADDATTR,L,1.1nH
ADDATTR,RatedCurrent,990mA
ADDATTR,DCRes,0.03
ADDATTR,Freq,20.0GHz

; create a second part using characteristics from the first part
; but overwrite attributes using different values
ADDPART,LQW04AN1N8C00p
ADDATT,L,1.8nH
ADDATTR,RatedCurrent,700mA
ADDATTR,DCRes,0.06
ADDATTR,Freq,17.0GHz

```

32 Examples

Adding Alphanumeric Pins to Parts

```
FORMAT,PGF,3

; create a new part with alphanumeric pin names
PART,NewAlpha,Alphanumeric Pins Example,ST,SOIC8,8,O1,I1,I2,P1,P2,++
O2,I21,I22

SETSTEM,U
; add two gates and name the pins using the alphanumeric pin names
ADDGATE AMP,5,O1,P1,I1,I2,P2
ADDPINATTR,O1,PINTYPE=OUTPUT,LOGICNAME=OUT1
ADDGATE,AMP2,3,I21,I22,O2
ADDPINATTR,O2,PINTYPE=OUTPUT,LOGICNAME=OUT2
```


Appendix B. Change History

Issue 27-Aug-14 [Pulsonix 8.0]

Addition of the following commands for adding gate and pin swap information to gates; GATESWAP and PINSWAP.

Note: The existing SWAPGROUP will still function but has been replaced with PINSWAP.

Issue 03-May-13 [Pulsonix 8.0]

SETGATESYMBOLS command was missing the part representation name from the list of parameters.

Issue 03-Sept-12 [Pulsonix 7.5]

Clarification of some commands within the Part files section.

Description of existing EXIT command added.

Issue 20-Oct-10 [Pulsonix 6.1]

Addition of RENAMEPART and RENAMEPARTMATCHATTR commands for PGF files, allowing part rename operations on designs and libraries.

Issue 30-Apr-09 [Pulsonix 6.0]

Show how to add comments to the file using # as the first character on the line.

Correction to the NEWPART command on a part, to show the missing optional field for the number of Symbol pins.

Handling of multiply-mapped pins in NEWPART, ADDGATE and ADDPARTREP commands.

Change of default handling for part grouping for format 3 part files, see page 21.

Addition of the following commands:

ASSOCPART : create a new Associated Part.

COPYPART : make a copy of an existing part under a new name.

MAKEGROUP : collect existing parts together into a part group.

PARTGROUP : collect new parts together into a part group.

SETGATESYMBOLS : set the list of symbol names to apply to a gate.

Issue 20-Oct-08 [Pulsonix 5.1]

New SETGATESYMBOLS command on a part, allowing the addition of alternate symbol names to an existing gate definition.

Issue 08-May-08 [Pulsonix 5.0]

ADDATTR command on a part can now add 'built-in' attributes such as Family, Height and Hyperlink.

PADSHAPE commands now recognise a larger range of available pad shapes.

Issue 27-Mar-08

SETFOOTPRINT command can now accept multiple footprint names, to specify a list of alternate footprints for a part. First footprint name is used as the default footprint for the part.

Issue 18-Feb-04

Changes added for Pulsonix version 3.0.

Additional commands for generating new parts: ADDUNGATEDPINS, ALLOWADDPINS, CONNECTOR,

Additional commands for amending existing parts: DELATTR, SETDESCRIPTION, SETFOOTPRINT, SETGATE, SETSTEM, SETSYMBOL.

Part command line continuation suffix "++"

Issue 01-Sep-03

Changes for handling symbol integration files, added for Pulsonix version 2.1.

Issue 02-June-03

Addition of command for finding Parts (FINDPART)

Issue 26-Feb-03

Addition of commands for bond pads and wires (ADDBONDPAD, ADDBONDPADSET, BONDPADDRILL, BONDPADLAYER, BONDPADLENGTH, BONDPADSHAPE, BONDPADSIZE, BONDPADSTYLE, WIRELAYER, WIRESIZE, WIRESTYLE).

Addition of command ADDPADSET for adding multiple 'arrays' of pads in specific positions.

Issue 09-Sep-02

Addition of PART and ADDGATE commands to facilitate multi-gate parts.

Issue 21-Aug-02

Addition of command ADDPINATTR for part pin attributes and properties.

Addition of command PARTGROUP for grouping parts.

Addition of commands AUTONAME and AUTOPART for creating auto-named parts.

Addition of command PLACEBOX for user-defined placement shape in footprint.

Changes to sample listings for above commands.

Issue 06-Aug-02

Values shown for PADLAYER changed from <All>, <Top>, <Bottom> to <Through Board>, <Top Side> and <Bottom Side> respectively.

Note that ADDPAD command over-rides default pad pattern for footprint.

Addition of sample footprint commands for use of ADDPAD command.

Change to ADDPART command to match actual program behaviour. Copies details from previously added part, does not add to same part group.

Index

A

Alphanumeric pin names, 22, 32

C

Change History, 33

Comment command (#), 9, 13, 21

E

EXIT command (#), 9, 13, 21

K

Keywords

#, 9, 13, 21

++, 27, 31

ADDATTR, 22

ADDBONDPAD, 14

ADDBONDPADSET, 14

ADDGATE, 23

ADDPAD, 14, 18, 30, 35

ADDPADSET, 14

ADDPART, 23, 31

ADDPINATTR, 23, 31

ADDUNGATEDPINS, 24, 31

ALLOWADDPINS, 24

ASSOCPART, 21

AUTONAME, 24

AUTOPART, 24

AXIALLEGS, 18

BONDPADDRILL, 15

BONDPADLAYER, 15

BONDPADLENGTH, 15

BONDPADSHAPE, 15

BONDPADSIZE, 15

BONDPADSTYLE, 15

BOTTOMPINDIRLEFT, 10

BOTTOMPINORDER, 10

BOTTOMPINS, 10

BOXLENGTH, 18

BOXROUND, 18

BOXWIDTH, 18

CLKPIN, 10

COLCOUNT, 18

CONNECTOR, 24

COPYPART, 25

DELATTR, 25

DIAMETER, 19

DRILL, 15

EXIT, 9, 13, 21

FILE, 9, 13, 21

FINDPART, 21

FIRSTPIN, 15

FORMAT, 9, 13, 21

GATESWAP, 25

HORIZPADCOUNT, 18

HORIZROWPITCH, 18

INVPIN, 10

LEFTPINDIRUP, 10

LEFTPINORDER, 10

LEFTPINS, 10

LEGLEN, 10

LOGICNAME, 23

MARKPIN, 10

NAME, 10, 15

NAMEPOS, 10, 16

NAMESTEM, 25

NETNAME, 23

NUMBERSOUTSIDE, 10

ORIGIN, 10, 16

PAD1SHAPE, 16

PADCOUNT, 16

PADDIM3, 11, 16

PADFILLED, 11

PADLAYER, 16, 35

PADLENGTH, 11, 16

PADPITCH, 11, 16

PADSHAPE, 11, 16

PADSIZE, 11, 16

PADSTYLE, 11, 16

PART, 22

PARTGROUP, 25

PINORDER, 17

PINSWAP, 23

PINTYPE, 23, 24

PLACEBOX, 17

PLACECREATE, 17

PLACEGAP, 17

PLACEISAREA, 17

PLACELAYER, 17
POLPIN, 11
RADIUS, 19
RENAMEPART, 26
RIGHTPINDIRUP, 11
RIGHTPINORDER, 11
RIGHTPINS, 11
ROWCOUNT, 18
ROWPITCH, 17
SEGLEFTPINS, 12
SEGMENTS, 12
SEGRIGHTPINS, 12
SETDESCRIPTION, 26
SETFOOTPRINT, 26
SETGATE, 26
SETGATESYMBOLS, 27
SETSTEM, 27
SETSYPHOL, 27
SHOWLOGICNAMES, 11
SHOWPINNAMES, 11, 17
SILKCREATE, 17
SILKDOT, 17
SILKGAP, 17
SILKLAYER, 17
SILKMITRE, 17
SILKMITRESIZE, 17
SILKNOTCH, 17
SILKNOTCHSIZE, 17, 18
SPICENAMEPOS, 11
STAGGERODD, 18

STAGGERPITCH, 18
STARTANGLE, 19
SYMWIDTH, 11
TABANGLE, 19
TABSIZI, 19
TOPPINDIRLEFT, 12
TOPPINORDER, 12
TOPPINS, 12
TYPE, 9, 13
UNITS, 9, 13
VERTPADCOUNT, 18
VERTROWPITCH, 18
WIRELAYER, 18
WIRESIZE, 18

P

Parameter types

- angle, 19
- bool, 19
- count, 19
- size, 19
- x, 19
- y, 19

S

Sample files

- Footprints, 28
- Parts, 30
- Symbols, 28