SIEMENS EDA

ODB++ Inside for Cadence® Allegro®

Release vNPI 2403
March 2024

SIEMENS

Unpublished work. © 2024 Siemens

This Documentation contains trade secrets or otherwise confidential information owned by Siemens Industry Software Inc. or its affiliates (collectively, “Siemens”), or its licensors. Access to and use of this Documentation is strictly limited as set forth in Customer’s applicable agreement(s) with Siemens. This Documentation may not be copied, distributed, or otherwise disclosed by Customer without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This Documentation is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this Documentation without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made.

No representation or other affirmation of fact contained in this Documentation shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

If you have a signed license agreement with Siemens for the product with which this Documentation will be used, your use of this Documentation is subject to the scope of license and the software protection and security provisions of that agreement. If you do not have such a signed license agreement, your use is subject to the Siemens Universal Customer Agreement, which may be viewed at https://www.sw.siemens.com/en-US/sw-terms/base/uca/, as supplemented by the product specific terms which may be viewed at https://www.sw.siemens.com/en-US/sw-terms/supplements/.

SIEMENS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS DOCUMENTATION INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. SIEMENS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, LOST DATA OR PROFITS, EVEN IF SUCH DAMAGES WERE FORESEEABLE, ARISING OUT OF OR RELATED TO THIS DOCUMENTATION OR THE INFORMATION CONTAINED IN IT, EVEN IF SIEMENS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TRADEMARKS: The trademarks, logos, and service marks (collectively, "Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' Marks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Table of Contents

Chapter 1
ODB++Design Export 5
Translating a Design to ODB++Design Format 5
Saving the Configuration 7
Editing the Matrix File 8
ODB++ Inside Wizard Pages 11
Specifying File Options and Output Options Page 12
Specifying Partial Export Parameters Page 16
Specifying Additional Parameters Pages 18

Chapter 2
System Administrator Notes 31
Running the Translator from Design Workbench 31
ODB++Design Entity Naming Rules 31
ODB++ Inside Environment Variables 33
Configuration Parameters 34
Command Line Parameters 39
Thermal Model Configuration 45
Structure of the Thermal Model File 45
Thermal Model Examples 48
Information Acquired from Cadence Allegro Data 50
Generated Extract Files 50
Importing Allegro Geometry Properties 58
Importing Allegro Component Properties 60
Deriving Component Outline From Specific Subclasses 62
Cadence Allegro DFA Table 64
Supported Features 67
Support for Backdrill Depth, Stub Length, and Must Not Cut Layer 67
Support for Improved Component Placement 67
Support for Mask Layers Associated With Inner Copper Layers 70
Support for Padstack Types and Usage 71
Support for Test Point Shapes 71
Support for Intentional Shorts 71
Support for Components Excluded From BOM 71
Support for DFA Boundaries 71
Support for Partial ODB++Design Output 72
Support for Flex Subtypes 72
Support for Boundary Elements 72
Support for Backdrill Size 72
Support for Dielectric Layer Subtypes 72
Support for Bend Areas 73
Support for Skipping Extraction of Net Impedance Average 73
Package Height Properties 73
Support for CLASS_CONSTRAINT_REGION 79

Support for Translating Back-Drill Information....79
Support for Mirrored Padstacks....80
Support for the COMPONENT KEEPOUT Class....80

Chapter 1
ODB++Design Export

ODB++Design format can capture all CAD or EDA assembly and PCB fabrication information in a single, unified file structure. ODB++ Inside is installed as part of Cadence Allegro to allow you to export a design to ODB++Design and to view the resulting ODB++ product model.

ODB++ Inside for Cadence Allegro contains the following components:

- **BRD2ODB translator** — Converts *.out* files, generated by Cadence Allegro, to ODB++Design version 8 or ODB++ version 7. The name of the Cadence Allegro design is contained in the names of the *.out* files. See "Generated Extract Files" on page 50.
    - If you are running the translator from within Cadence Allegro, you can specify a *.brd* file as the input path.
    - If you are running ODB++ Inside stand-alone, you must specify a directory containing the *.out* files that have been extracted from Cadence Allegro.
- **ODB++ Viewer** — Displays the resulting ODB++Design information, graphically. See *ODB++ Viewer User Guide*.

When Allegro is to be launched from the Allegro Design Workbench, environment variable PCBDW_USER_PATH must be set when ODB++ Inside is installed, as described in "Running the Translator from Design Workbench" on page 31.

The translator supports files from version 11 through 17.2 of these Cadence Allegro products:

- *.brd* file — From Cadence Allegro PCB Designer
- *.mcm* file — From Cadence Allegro Package Designer (APD)
- *.sip* file — From Cadence SIP

The translator does not include the option to save as the earlier ODB++ Version 6. This functionality was removed so that there is no confusion over what should be sent to manufacturing. Manufacturers must use a software version capable of reading ODB++ Version 7 or ODB++Design Version 8 format. Mentor Graphics Frontline applications such as Genesis work with a variation of the ODB++Design format, but they can import and use the ODB++ Version 7 and the ODB++Design Version 8 format.

Translating a Design to ODB++Design Format
Saving the Configuration
Editing the Matrix File
ODB++ Inside Wizard Pages

Translating a Design to ODB++Design Format

You use the ODB++ Inside for Cadence Allegro translator to specify parameters and to run the translation, to export a Cadence Allegro design to an ODB++ product model.

Prerequisites

- Environment variables have been set. See “ODB++ Inside Environment Variables” on page 33.
- Configuration parameters have been set. See “Configuration Parameters” on page 34.
- Thermal models have been configured. See “Thermal Model Configuration” on page 45.

Procedure

1. Launch ODB++ Inside from within Cadence Allegro or stand-alone:
   - From within Cadence Allegro, choose **File > Export > ODB++ Inside** or click .

   - Use a line mode command to activate the stand-alone translator:
     - Windows:

       ```
       "%ALLEGRO_BRD2ODB%/brd2odb.exe" -gui
       ```

     - UNIX:

       ```
       $ALLEGRO_BRD2ODB/brd2odb -gui
       ```

     See “Command Line Parameters” on page 39.
2. Specify the information described in “Specifying File Options and Output Options Page” on page 12.

   Restriction:
   Non-ASCII characters in the pathname are not supported.

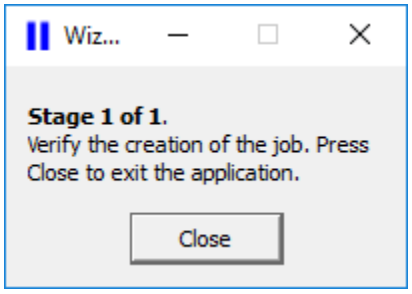
   - If you are running ODB++ Inside from within Cadence Allegro, you can specify a *.brd* file as the input path.
   - If you are running ODB++ Inside stand-alone, you must specify a directory containing the *.out* files that have been extracted from Cadence Allegro. See “Generated Extract Files” on page 50.
3. If you have selected Export Option = Partial, specify the partial export parameters as described in “Specifying Partial Export Parameters Page” on page 16.
4. If you have selected Show more options = Yes, specify the information as described in “Specifying Additional Parameters Pages” on page 18.

5. Click **Next** on the last page of parameters to perform the translation.
6. If you want to restart the wizard and re-enter the options, click **Setting > Reset Wizard**.

Results

The product model is written in ODB++Design format to the specified location.

If you have selected Open ODB++ viewer = Yes, the ODB++ Inside wizard pauses and prompts you to verify the creation of the job. Click **Close** to exit the application:



ODB++ Viewer opens, displaying the product model as described in *ODB++ Viewer User Guide*.

Saving the Configuration

If you will be using the same configuration parameters for several translations, you can save the configuration to a file.

You can save the configuration to the standard user location, to the standard system location, or to another location. The user-level configuration, if it exists, is loaded when ODB++ Inside starts. Otherwise, the system-level configuration is used to supply default values for the wizard.

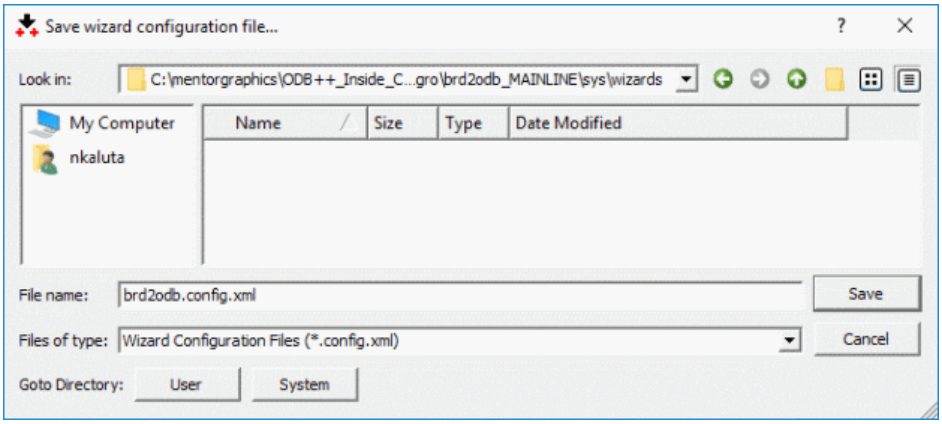
Prerequisites

Run the ODB++ Inside wizard as described in “Translating a Design to ODB++Design Format” on page 5.

Procedure

1. Choose **Setting > Save Config**.

   The Save wizard configuration file dialog box opens.



2. Specify the parameter values:

Parameter	Description
Look in:	The directory in which to save the configuration file. This can be the standard user location or the standard system location, or another location. • To store files at the user location, click the **User** button. The user location is displayed in this field. If there is a file saved at the user location, it is loaded when ODB++ Inside opens. • To store the file at the system location, click the **System** button.The system location is displayed in this field. This configuration is loaded when ODB++ Inside opens, if there is no configuration file in the user location. • You can save the configuration file in another location. To use the parameter settings, copy the file to the standard name, in either the user location or the system location, before opening ODB++ Inside.
File name	The file name to which to save the configuration. If you are storing the file at the user location or at the system location, and you want the wizard to load the default values from this file on startup, save the configuration to the standard file name: *brd2odb.config.xml*. If you are storing the configuration at a different location, to a file that will be copied to the user location or the system location as needed, you can specify any file name.
Files of type	If you are storing the file at the user location or at the system location, leave the default file type.

Editing the Matrix File

If necessary, you can edit the information about the layers that were extracted from the Cadence Allegro design. For each layer you can edit the context, type, polarity, and side.

The options of the matrix file editor are equivalent to the options on the Artwork Control Form dialog box of Cadence Allegro.

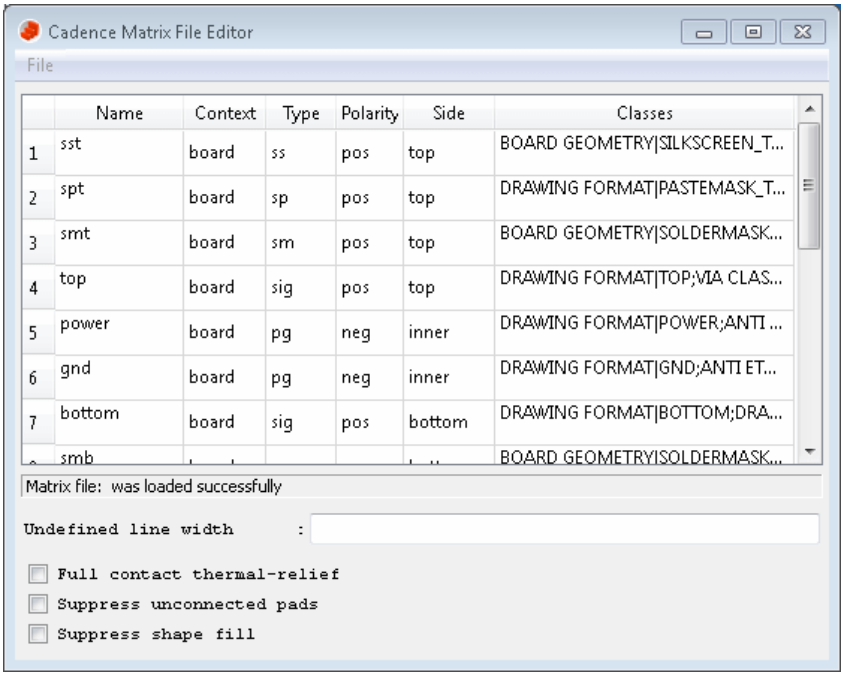
Layers are translated according to the data taken from the files *layers_<product_model>.out* and *films_<product_model>.out*. It is not unusual to find data for copper layers mixed with document layers.

The translator designates the top and bottom layers according to the pairs of class | sub-class ETCH|*<layer_name>*. If several layers contain these pairs, the first one found is used. To avoid the mixing and duplication of layer data, it is necessary to edit the matrix file before translation.

The first time a design is translated, it does not usually contain a matrix file.

Procedure

1. Use the drop-down lists in the Cadence Matrix File Editor window to edit parameters so that each layer is correctly defined.



   - If you change a top or bottom layer to a document layer, its name is changed to what it was originally.
   - If you change a document layer to a signal layer, its name is assigned according to the ETCH sub-class found in it.

   Make sure that changes to layers remain synchronized. For example, signal must be assigned side = top or bottom and power and ground layers must be side = inner. They cannot be of context misc. Document layers must be assigned side = auto and polarity = pos. Unsynchronized data causes incorrect translation.

2. Set options for thermal relief, unconnected pads, and shape fill for each layer.

Option	Explanation
Full contact thermal-relief	Controls the creation of thermal symbols on a specific layer.

Option	Explanation

	• selected — Suppresses the creation of thermal symbols.
	• cleared — Creates thermal symbols, if they are defined, in this way:
	  • If there are *<thermal symbol name>.outdra* files, thermal symbols are added as defined in these files.
	  • If there are no *outdra* files, and Use thermal model file = Use file was specified in the wizard, the thermal model specified in Set filename of thermal model is searched. If there are thermal symbols defined there, they are added.
	The file *valor_ex.il* creates ASCII files named *<thermal symbol name>.outdra* if there are DRA files with the design. These files are used to create thermal symbols. Each file defines one thermal symbol. Only the thermals for which there are *outdra* files are replaced.

Suppress unconnected pads	Controls whether unconnected pads are suppressed for the selected layer.
Suppress shape fill	Controls the creation of the laminate area for the selected layer during translation. • selected — Creation of the laminate area is suppressed. The design must have filled areas replaced with separation lines in Power & Ground layers. • cleared — By default, text on P&G layers is translated with negative polarity. This reads product models in the same way the -s switch is used in the Allegro Artwork command. The laminate area is created for all negative layers by creating a single surface consisting of the board outline (filled) with all split plane areas subtracted from it. Creation of the laminate area in ODB++ is equivalent to the “shapefill” algorithm in Allegro (the -s switch is used to suppress the shapefill algorithm).

3. Choose **File > Save** to save the corrections. You can specify the edited matrix file in Matrix file so that the translation creates layers according to the file.

ODB++ Inside Wizard Pages

The ODB++ Inside wizard comprises a set of pages that lead you through the translation stages: setting the file options and output options, configuring partial output, setting additional translation parameters, and running the translation.

The pages are listed in order of execution of the wizard stages:

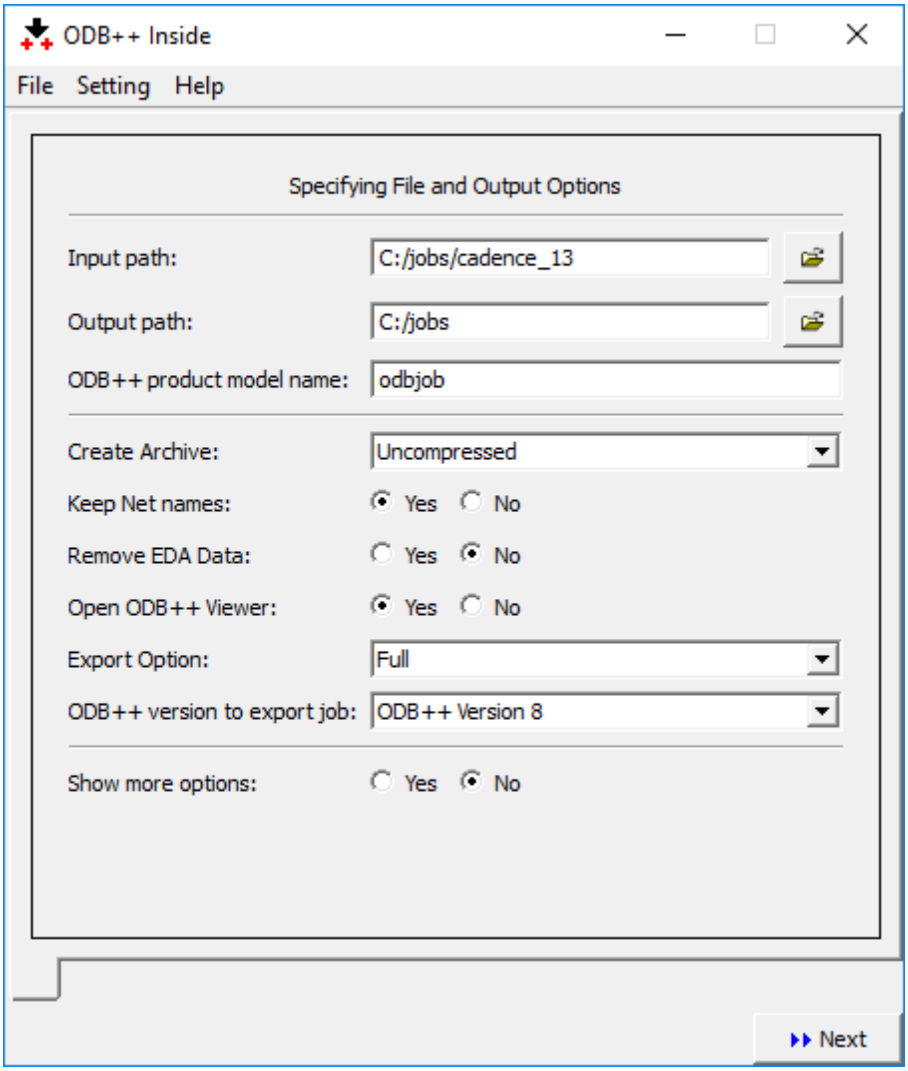
Specifying File Options and Output Options Page
Specifying Partial Export Parameters Page
Specifying Additional Parameters Pages

Specifying File Options and Output Options Page

You access this page while performing Step 2 of the procedure “Translating a Design to ODB++Design Format”.

You must provide input and output paths and output options needed by the translator, and select actions to be performed by the translator.

Figure 1. Specifying File Options and Output Options Page



Objects

Table 1. File Options

Object	Description
Input path	The input path of the Allegro design.

Table 1. File Options (continued)

Object	Description
	Click to browse to a file or a directory. If you are running ODB++ Inside from within Cadence Allegro, you can specify a *.brd* file as the input path. If you are running ODB++ Inside stand-alone, you must specify a directory containing *.out* files that have been extracted from Cadence Allegro. See "Generated Extract Files" on page 50.
Output path	The path for the ODB++Design output. Click to browse to a file or a directory.
Output product model name	The name of the ODB++Design product model to be created.

Table 2. Translator Actions

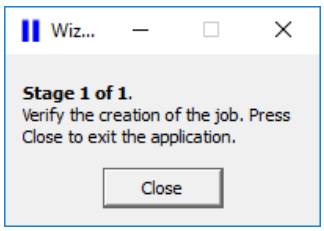
Field	Description
Create Archive	Controls the format of the ODB++Design output. • **Uncompressed** (default) • **Tar** — Compresses the ODB++Design folders into a tared file. • **Tar gzip (.tgz)** — Compresses the ODB++Design folders into a tared and zipped *tgz* file.
Keep Net names	Controls whether net names are renamed numerically or are kept as their original names.
Remove EDA Data	Removes component/package data.
Open ODB++ Viewer	Opens the ODB++ Viewer application to display the imported design, when the translation completes. The ODB++ Inside wizard remains open, but is paused. The wizard displays the message Verify the creation of the job. Click **Close** to exit the application.  The ODB++ Viewer opens, displaying the resulting ODB++Design data. To close ODB++ Viewer and the ODB++ Inside wizard, perform one of these tasks: • Choose **File > Exit** to close the ODB++ Viewer. • In the Wizard Paused message box, click **Close**.

Table 2. Translator Actions (continued)

Field	Description
	If your examination of the ODB++Design data indicates that you need to change import parameters, you can click **Setting > Reset Wizard** in the ODB++ Inside wizard to restart the wizard. If you have saved your configuration, you only need to enter the parameters that need to be changed. See *ODB++ Viewer User Guide*.

Export Option	Controls how much data is exported to ODB++Design:
	• **Full** — All information in the design Export Fabrication.
	• **Partial** — You can select which data is exported. See "Specifying Partial Export Parameters Page" on page 16.
	• **FAB** — Exports layers and data options for fabrication:
	  • Physical nets - output for net points
	  • Outer copper layers
	  • Silk Screen layers
	  • Solder Paste layers
	  • Solder Mask layers
	  • Drill / Rout layers
	  • Document layers
	  • Inner layers
	• **ASSY** — Exports layers and data options for assembly:
	  • Components/Packages & Logical nets - components + logical nets (net nodes/net attributes/net properties)
	  • Physical nets - output for net points
	  • Outer copper layers
	  • Silk Screen layers
	  • Solder Paste layers
	  • Solder Mask layers
	  • Drill / Rout layers
	  • Document layers

ODB++Design version to export job	One of these ODB++Design versions: • ODB++Design Version 8 • ODB++ Version 7
Show more options	Activates the options for setting additional parameters as described in Table 3: Specifying Additional Parameters - Page 1 through Table 6: Specifying Configuration Parameters Page.
Next	Does one of the following:

Table 2. Translator Actions (continued)

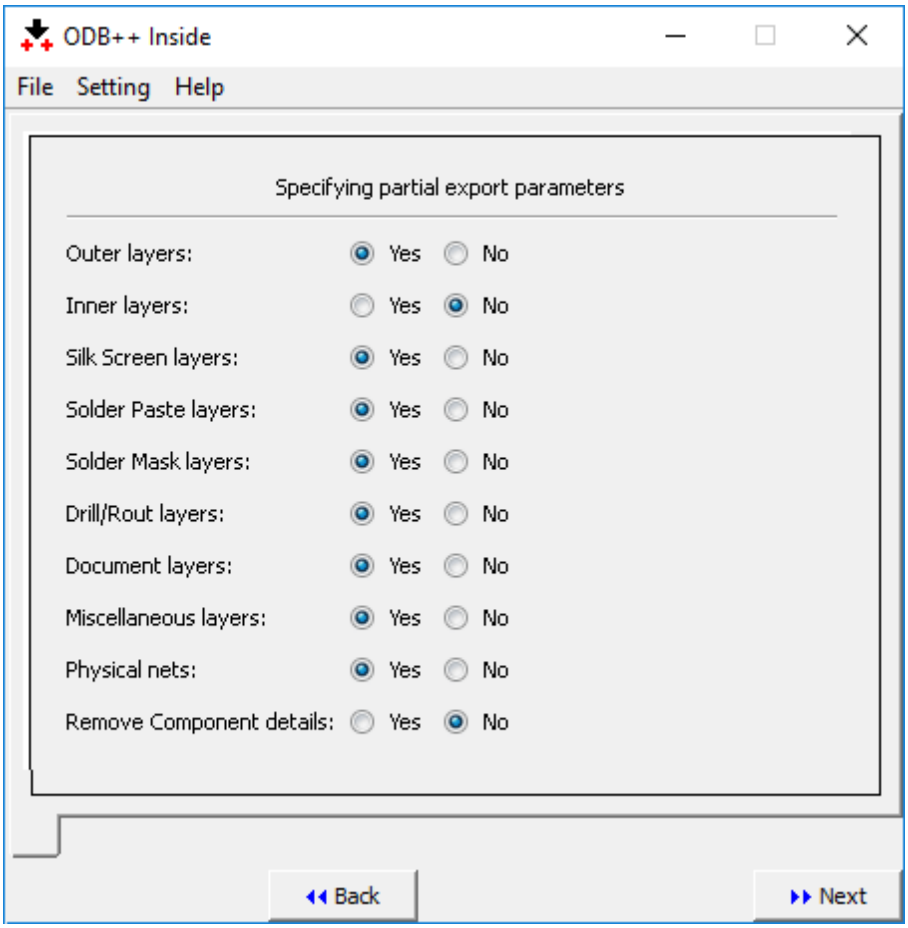
Field	Description
Next	• If you selected Export Option **= Partial**, displays Specifying Partial Export Parameters Page. • If you selected Export Option **≠ Partial** AND Show more options **= Yes**, displays Specifying Additional Parameters Pages. • If you selected Export Option **≠ Partial** AND Show more options **= No**, runs the translation.

Specifying Partial Export Parameters Page

You access this page while performing Step 3 of the procedure “Translating a Design to ODB++ Format”.

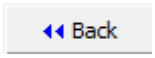
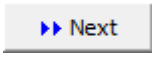
You can specify which information should be extracted from the design.

Figure 2. Specifying Partial Export Parameters Page



Objects

Object	Description
Outer layers	Yes / No Controls whether to include the outer layers in the ODB++ product model.
Inner layers	Yes / No Controls whether to include the inner layers in the ODB++ product model
Silk Screen layers	Yes / No Controls whether to include the silk screen layers in the ODB++ product model.

Object	Description
Solder Paste layers	Yes / No Controls whether to include the solder paste layers in the ODB++ product model.
Solder Mask layers	Yes / No Controls whether to include the solder mask layers in the ODB++ product model.
Drill/Rout layers	Yes / No Controls whether to include the drill/rout layers in the ODB++ product model.
Document layers	Yes / No Controls whether to include the document layers in the ODB++ product model.
Physical nets	Yes / No Controls whether to include the net names in the ODB++ product model.
Miscellaneous layers	Yes / No Controls whether to include the miscellaneous layers in the ODB++ product model.
Remove component details	Yes / No Controls whether to exclude attributes and properties of the components from the ODB++ product model.
Back 	Displays the Specifying File Options and Output Options Page.
Next 	Does one of the following: • If you selected Show more options **= Yes**, displays the Specifying Additional Parameters Pages. • If you selected Show more options **= No**, runs the translation.

Specifying Additional Parameters Pages

You access these pages while performing Step 4 of the procedure “Translating a Design to ODB++ Format.”

You can specify additional and configuration parameters.

Figure 3. Specifying Additional Parameters - Page 1

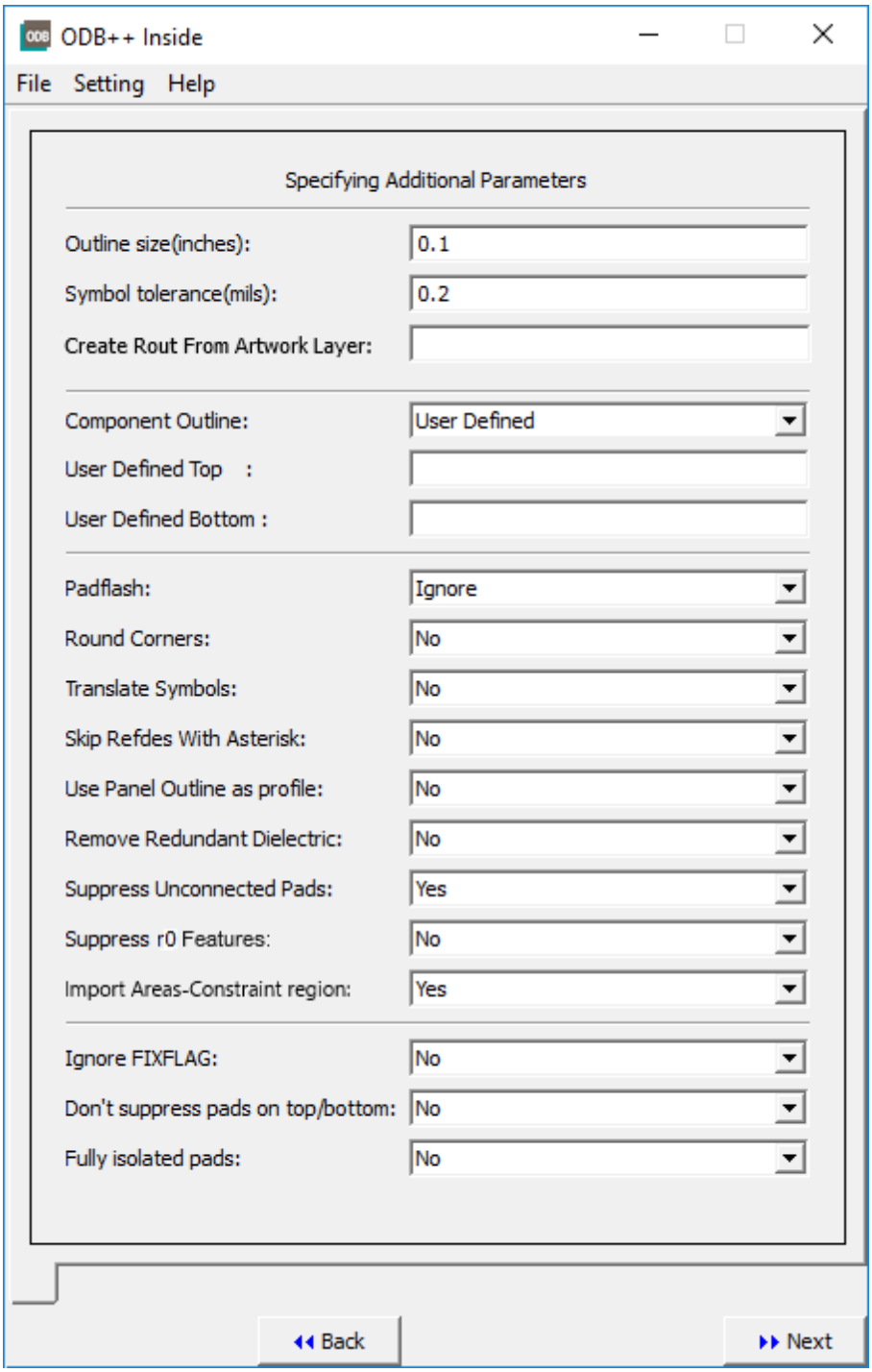


Figure 4. Specifying Additional Parameters - Page 2

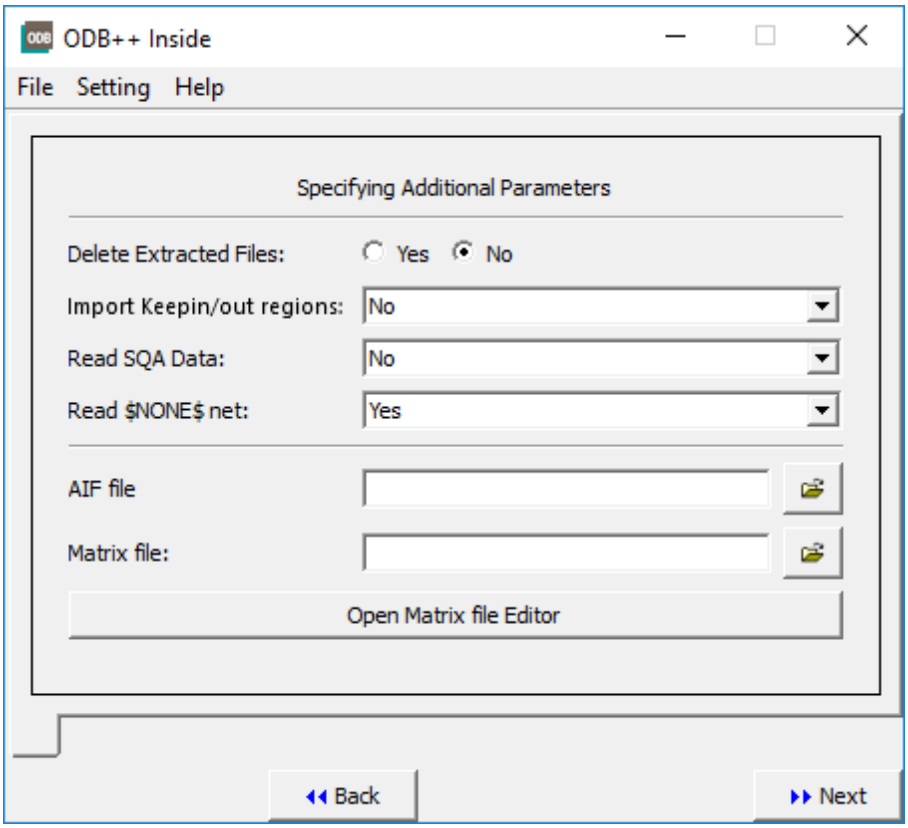
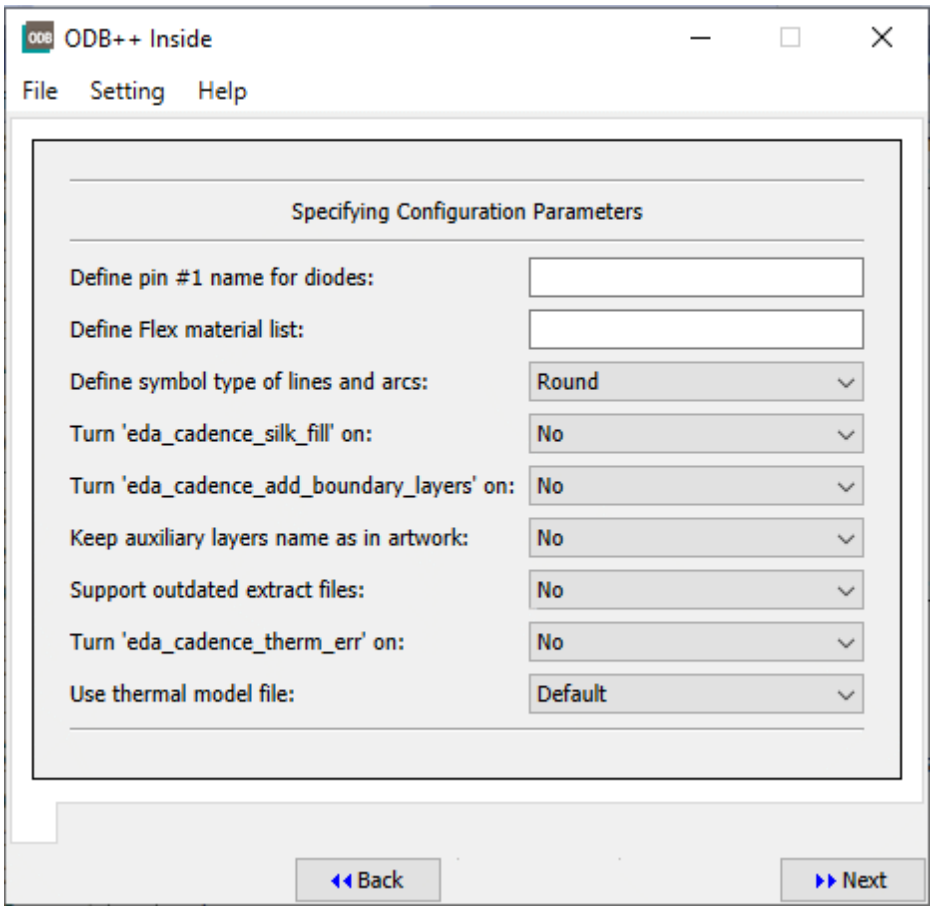


Figure 5. Specifying Configuration Parameters Page



Objects

Table 3. Specifying Additional Parameters - Page 1

Object	Description
Outline size (inches)	When creating negative plane layers, the size of the frame is the value of this parameter. For accurate translation this value should match the -o option in the Cadence Allegro artwork program. If these two parameters differ, the frame will be created according to the value in Outline size. The value is in inches. The field allows a precision of up to four digits. For example, 0.4321.
Symbol tolerance (mils)	The system compares shapes that are input, with symbols previously input in the same session, and with standard and semi-standard system symbols. • 0 — only if the input shape exactly matches a system symbol, is the system symbol used. If it does not match, the input shape is used “as is” without change. • positive value — the input shape is compared to system symbols within the tolerance specified. If it can be matched, the system symbol is used.

Table 3. Specifying Additional Parameters - Page 1 (continued)

Object	Description
	Use this parameter as appropriate for the type of file you expect to input. The lower the tolerance the more critical the system is in judging that shapes are equivalent. The value is specified in mils. The field allows a precision of up to four digits. For example, 0.2134.

Create Rout From Artwork Layer	Controls how the rout is created:
	• If the field contains the name of a valid layer, as specified in the Allegro artwork, the features in that layer are used to create an ODB++ rout layer with the original name.
	• If this field is empty, or contains the name of a non-existing layer, the translation creates a rout layer named “profile” by merging the features from the following Allegro artwork CLASS/SUBCLASS in the *geoms_<pm>.out* file:
	  • If DESIGN_OUTLINE data exists (Allegro 17.2 or later):
	  BOARD GEOMETRY:DESIGN_OUTLINE“&”BOARD GEOMETRY:CUTOUT
	  • If DESIGN_OUTLINE data does not exist (older versions):
	  BOARD GEOMETRY:OUTLINE“&”BOARD GEOMETRY:CUTOUT
	Related line mode command switch is -ral. See “Command Line Parameters” on page 39.

Component Outline	Controls how the component outline is created. • **Placebound** — (Recommended) When place bound shapes are available (PART GEOMETRY sub-classes PLACE_BOUND_TOP and PLACE_BOUND_BOTTOM) they are used for the component outline. Otherwise, the limits of the assembly features are used. • **Assembly** — The limits of the assembly features are used. A heuristic algorithm attempts to determine the actual component outline from the collection of data on the sub-classes ASSEMBLY_TOP and ASSEMBLY_BOTTOM of the package geometry. This may result in an unexpected component outline if the data defining it is not complete in terms of ODB++, that is, a well defined closed polygon. • **DFA** — If DFA boundaries data exists, the component outline is taken from the PART GEOMETRY sub-classes DFA_BOUND_TOP and DFA_BOUND_BOTTOM. Otherwise, pin bounding boxes are used. • **User Defined** — The component outline is taken from the sub-classes entered into User Defined Top and User Defined Bottom.
User Defined Top User Defined Bottom	Available only when Component Outline = User Defined. Specify the top and bottom subclasses from which the component outline is taken. These fields must be set with the values specified in the component subclasses file. See “Deriving Component Outline From Specific Subclasses” on page 62.

Table 3. Specifying Additional Parameters - Page 1 (continued)

Object	Description

Padflash	Allegro pad definitions can have padflash codes that override the pad size information for the padstack. This information is extracted into the twelfth field of the pad extract file (*pads_<brd name>.out*).
	For instance, on fiducials, a designer defines a padstack called FID120RD40RD that appears in Allegro as a 120 mil diameter pad with a 120 mil diameter solder mask. It also has a padflash definition of RD40.
	• **Ignore** — (default) The Padflash field is ignored and instead, the pad size is used. In the example, the example padstack would be constructed of 120 mil diameter pads during EDA translation.
	• **Substitute (Ignore missing)** — (recommended) Sets the Padflash definition using the following method:
	  • If the Padflash code exists and the *thermal_models* file using Cadence Allegro padflashes exists, the name in the PADFLASH field is used in conjunction with the thermal models file to determine what is placed at the location. In the example, the PADFLASH name RD40 would determine the actual fiducial on the copper layer based on the current thermal model.
	  • In other cases, the configuration parameters eda_cadence_read_dra and eda_cadence_therm_err control the translator behavior:
	  eda_cadence_read_dra = yes — The file *<thermal symbol name>.outdra* is read (if exists) during translation, providing the PADFLASH symbol definition.
	  eda_cadence_read_dra = no — The file *<thermal symbol name>.outdra* is ignored.
	  eda_cadence_therm_err = no — If the Padflash does not exist in the *thermal_models* file or as an *outdra* file, the original pad size is used.
	  eda_cadence_therm_err = yes — If the Padflash does not exist in the *thermal_models* file or as *outdra* file, the translation fails with a message listing the padstack name.

Round Corners	Indicates whether corners should be rounded. • **No** — (default) process precise (square) corners. • **Yes** — round corners of polygons (contours).
Translate Symbols	Indicates whether symbols should be translated as components. • **Yes** — (default) symbols are translated as components. If there are multiple shapes, each will be translated as a separate component. • **No** — symbols are not translated.
Skip Refdes With Asterisk	Controls whether components with names containing an asterisk (*) should be translated. • **No** — All components are translated. (default) • **Yes** — Components with names containing an asterisk are not translated. • **Part** — The translation excludes components whose RefDes contains an asterisk (*) but includes their pad and drill features.

Table 3. Specifying Additional Parameters - Page 1 (continued)

Object	Description
Use Panel Outline as profile	Controls which data with CLASS = BOARD GEOMETRY in the *geoms_<pm>.out* file is used to define the step profile: • **Yes** — Step profile is taken from one of the following subclasses, in this order of priority: a. PANEL_OUTLINE b. DESIGN_OUTLINE c. OUTLINE • **No** — Step profile is taken from one of the following subclasses, in this order: a. DESIGN_OUTLINE b. OUTLINE c. PANEL_OUTLINE Related line mode command switch is -up. See “Command Line Parameters” on page 39.
Remove Redundant Dielectric	Controls whether successive dielectric layers are combined: • **No** — (default) Combines successive dielectric layers. • **Yes** — Does not combine successive dielectric layers, which may result in the wrong calculation of back drill spans. Related line mode command switch is -rrd. See “Command Line Parameters” on page 39.
Suppress Unconnected Pads	Controls whether unconnected pads are included in the translated design if the Allegro design contains the following: • In the films file, SUPPRESS_UNCONNECTED_PADS = Yes. • In the pads extract file, FIXFLAG = o (optional). If FIXFLAG = f (fixed), it can be ignored by selecting Ignore FIXFLAG. This sets configuration parameter eda_cadence_suppress. Pads on negative layers and pads associated with embedded components are never suppressed. • **Yes** — Unconnected pads are not translated. This option activates the advanced suppressing options listed in Table 4: Specifying Additional Parameters - Suppress Unconnected Pads. • **No** (default) — Unconnected pads are translated. Related line mode command switches are -iff, -bb, -fi, and -ups. See “Command Line Parameters” on page 39.
Suppress r0 Features	Controls whether to suppress the creation of r0 lines and arcs on copper and solder layers.
Import Areas-Constraint region	When set to Yes, triggers the generation of a single ODB++ product model layer by the name of “fab_drc.” The layer is generated based on the Allegro layer group called Constraint region.

Table 3. Specifying Additional Parameters - Page 1 (continued)

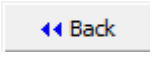
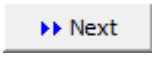
Object	Description
	Related line mode command switch is -rr. See “Command Line Parameters” on page 39.
Back 	Does one of the following: • If you selected Export Option **= Partial**, displays the Specifying Partial Export Parameters Page. • If you selected Export Option **≠ Partial**, displays the Specifying File Options and Output Options Page.
Next 	Displays Page 2 of Specifying Additional Parameters. See Table 5: Specifying Additional Parameters - Page 2.

Table 4. Specifying Additional Parameters - Suppress Unconnected Pads

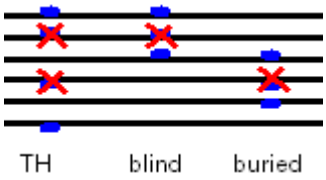
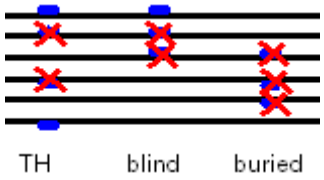
Object	Description
Prerequisite: Suppress Unconnected Pads = Yes (See Table 3: Specifying Additional Parameters - Page 1) **Note:** The following options are available to control which pads are suppressed, unless you are using options in Cadence Allegro V16.2 to control how to treat unconnected pads, and you access the translator from within Allegro.	
Ignore FIXFLAG	Ignores the setting of FIXFLAG in the pads extract file.
Don”t suppress pads on top/bottom	Controls whether unconnected pads at the top and bottom of a drill are suppressed. Available only if Suppress Unconnected Pads = Yes. • **Yes** — Unconnected pads at the top and bottom of a drill are not suppressed. (All unconnected pads other than those on the top and bottom of a drill are suppressed)  • **No** (default) — Unconnected pads at the top and bottom of a drill are suppressed. (All unconnected pads other than those on the top and bottom layers of a board are suppressed.) 
Fully isolated pads	Controls which pads are considered to be unconnected. Available only if Suppress Unconnected Pads = Yes.

Table 4. Specifying Additional Parameters - Suppress Unconnected Pads (continued)

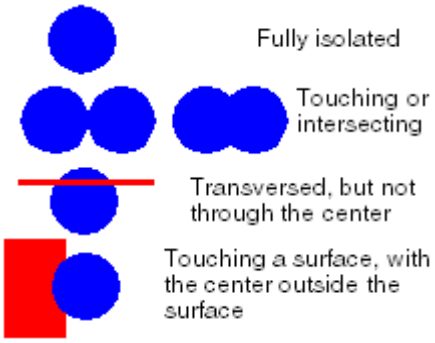
Object	Description
	• **Yes** — Only single pads, touching no other feature on the layer, are considered to be unconnected. Fully isolated • **No** — (default) All of these pads are considered to be unconnected: • A single totally isolated pad. • Two pads touching or intersecting. • A pad transversed not through its center by a trace that does not cross the center of the drill and the center of the pin (if exists). • A pad touching a surface where its center is not inside the surface.  Fully isolated Touching or intersecting Transversed, but not through the center Touching a surface, with the center outside the surface

Table 5. Specifying Additional Parameters - Page 2

Object	Description
Delete Extracted Files	Controls whether temporary extract files created during translation are deleted.
Import Keepin/out regions	When set to Yes, triggers the generation of multiple DRC layers beginning with the prefix “drc_”. • Allegro layers Route keepout, Route keepin and Via keepout are used to generate an ODB++ layer called drc_route. • Allegro layers Package keepout, Package keepin, Component keepout and Component keepin are used to create drc_comp_top and drc_comp_bottom. • Allegro layers No_Probe_Top and No_Probe_Bottom are used to create drc_tp_top and drc_tp_bottom.
Read SQA Data	Controls whether Signal Quality Analysis data should be read. • **Yes** — SQA data is read and a signal quality layer is created. • **No** — A signal quality data layer is not created and the tech file is not read. The translation takes less time.
Read $NONE$ net	Controls whether to assign features with no net to the $NONE$ net.

Table 5. Specifying Additional Parameters - Page 2 (continued)

Object	Description
	• **Yes** — Assign features with no net to the $NONE$ net (default). • **No** — Do not assign features with no net to the $NONE$ net.

Matrix file	To indicate the matrix file to use, perform one of these actions:
	• To use the matrix file generated from the product model, leave this field empty.
	• To use an existing matrix file, type the full path to the file.
	• To edit the matrix generated from the product model, and use the edited matrix file, perform these actions:
	  • Click **Open Matrix file Editor** to open the Cadence Matrix File Editor. Edit the file and save it. See "Editing the Matrix File" on page 8.
	  • Type the full path to the matrix file in the Matrix file field.

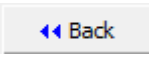
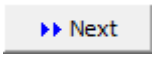
AIF File	HDI net information can be translated, and can be used to perform HDI net validation. By default, an AIF file residing in the same folder as the *out* files is used during translation. If your AIF file is located in a different folder, specify the location in the AIF File box. This location is used for subsequent translations even if there is an AIF file in the same folder as the *out* files, so be sure and change this location for subsequent translations if necessary. Make sure that you are using the current Skill script. Before opening Cadence Allegro to export information, copy the current script to the Cadence directory from this location: *<installation folder>\all\eda\cadence\set_allegro*
Back 	Displays Page 1 of Specifying Additional Parameters. See Table 3: Specifying Additional Parameters - Page 1.
Next 	Displays the Specifying Configuration Parameters Page. See Table 6: Specifying Configuration Parameters Page.

Table 6. Specifying Configuration Parameters Page

Object	Description
Define pin #1 name for diodes	Lists the names of diode leads that will be designated as pins #1. Multiple values are separated by semicolons (;). Sets configuration parameter diodes_pin1_name. For example, if you want to designate all leads named "K" and "C" as pins #1, type the following values: K;C
Define Flex material list	Lists the names of flexible dielectric materials. Multiple values are separated by semicolons (;). For example:

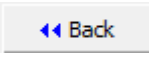
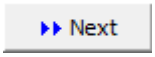
Table 6. Specifying Configuration Parameters Page (continued)

Object	Description
	POLYIMIDE;POLYIMIDE_FILM At import of Cadence Allegro data, the copper layers below and above a dielectric layer whose material definition in the *layers_<pm>.out* file matches one of the flexible dielectric material names, are assigned appropriate flex subtypes according to their base type. See “Subtypes to Support Flex/Rigid Flex Manufacturing” in the *Getting Started With ODB++Design*. Sets configuration parameter eda_flex_material.
Symbol type for lines and arcs	Sets the symbol type for lines and arcs of the step profile polygon. • **Round** — The symbol type is round. • **Square** — The symbol type is square. Sets configuration parameter eda_cadence_profile_sym_type.
Turn eda_cadence_silk_fill on	Fills surfaces on Allegro silk screen layers. Sets configuration parameter eda_cadence_silk_fill.
Turn eda_cadence_add_boundary_layers on	Controls whether to use data with CLASS = BOUNDARY in the *geom_<pm>.out* file to create boundary layers. • **Yes** — Creates a boundary layer for each line with CLASS = BOUNDARY. • **No** — Does not create boundary layers. Sets configuration parameter eda_cadence_add_boundary_layers.

Keep auxiliary layers name as in artwork	Controls whether to translate the names of silk screen, solder paste, and solder mask layers.
	• **No** — Renames auxiliary layers based on their subclass in the *films_<pm>.out* file:
	  • “SILKSCREEN_TOP”, “sst”
	  • “AUTOSILK_TOP”, “sst”
	  • “SILKSCREEN_BOTTOM”, “ssb”
	  • “AUTOSILK_BOTTOM”, “ssb”
	  • “PASTEMASK_TOP”, “spt”
	  • “PASTEMASK_BOTTOM”, “spb”
	  • “SOLDERMASK_TOP”, “smt”
	  • “SOLDERMASK_BOTTOM”, “smb”
	• **Yes** — Keeps the auxiliary layer names as defined in the *films_<pm>.out* file.
	Sets configuration parameter eda_cadence_keep_auxiliary_layers_name.
	Related line mode command switch is -kal. See “Command Line Parameters” on page 39.

Support outdated extract files	Controls whether to translate extract files that were created with a *valor_ext.il* version prior to 2305.

Table 6. Specifying Configuration Parameters Page (continued)

Object	Description
	• **Yes** — Translate extract files regardless of their version. In rare cases, the logic used in old extract files may place some components on the wrong side of the board. • **No** — (default) Translate only current extract files. The translation of outdated extract files fails. Sets configuration parameter eda_cadence_support_old_extract.
Turn eda_cadence_thermal_error on	The translator aborts with a message listing the padstack / thermal names that did not have a match in the models file. Sets configuration parameter eda_cadence_thermal_error.
Use thermal model file	Controls whether a thermal file is used. • **Default** — Use a default model that uses direct connect and no thermals. • **Use file** — Use a model stored in a thermal model file.
Set file name of thermal model	The file to be used when Use thermal model file = Use file. Click **Select Model** to select the model in the file.
Back 	Displays Page 2 of Specifying Additional Parameters. See Table 5: Specifying Additional Parameters - Page 2.
Next 	Runs the translation.

Chapter 2
System Administrator Notes

Information is provided that might be of interest to you as you convert a Cadance Allegro design to an ODB++ product model.

Running the Translator from Design Workbench
ODB++Design Entity Naming Rules
ODB++ Inside Environment Variables
Configuration Parameters
Command Line Parameters
Thermal Model Configuration
Information Acquired from Cadence Allegro Data
Supported Features

Running the Translator from Design Workbench

When Allegro is to be launched from the Allegro Design Workbench, environment variable PCBDW_USER_PATH must be set when ODB++ Inside is installed.

Procedure

1. Locate the Allegro Design Workbench launch wrapper file *adwstart.bat*. This file is typically located under the install tree.
2. Edit *adwstart.bat* to include this line:

   ```
   set PCBDW_USER_PATH=<path to ODB++ Inside>\nv\bin
   ```

   where *<path to ODB++ Inside>* is the path to the ODB++ Inside module, typically *C:\MentorGraphics\Allegro Export ODB++Design*.

ODB++Design Entity Naming Rules

ODB++Design entity names must follow the naming conventions.

- The length of any name must not exceed 64 characters.
- Only these characters are legal in an ODB++Design entity name:

    - lower case letters (a - z)
    - digits (0 - 9)
    - hyphen (-), underscore (_), dot (.), plus (+)

- Names must not start with hyphen (-), dot (.), or plus (+), and must not end with a dot (.). The one exception is system attributes, which start with a dot. User attributes must not start with a dot.

ODB++ Inside Environment Variables

ODB++ Inside directory locations are governed by environment variable settings specified at installation time.

ALLEGRO_BRD2ODB

On Windows, this environment variable is set during installation with the path to the ODB++ Inside Application Directory.

On Linux, use the setenv command to set the variable:

```
setenv ALLEGRO_BRD2ODB <path to Application Directory>
```

Default:

- (Windows) *C:\MentorGraphics\ODB++_Inside_Cadence_Allegro\brd2odb_<ver>*
- (Linux) */usr/local/BRD2ODB/brd2odv_<ver>*

The ODB++ Inside Application Directory contains a file named *env_file*. This file contains Valor environment variables set during installation: VALOR_DIR, VALOR_HOME, and VALOR_TMP. The paths defined in *env_file* are not overwritten on upgrade.

VALOR_DIR

Placed in the *env_file* file by the installer. Specifies the directory where ODB++ Inside system, configuration, and work files are stored. This directory can be installed locally or in a remote location accessible for reading and writing by all Allegro users.

Default:

- (Windows) <*ODB++ Inside Installation Directory*>*brd2odb_dir*
- (Linux) *BRD2ODB/brd2odb_dir*

VALOR_HOME

Placed in the *env_file* file by the installer. Once the user starts working with ODB++ Inside, a directory named *.genesis* is created under VALOR_HOME to store user-level configuration files.

Default:

- (Windows) <*ODB++ Inside Installation Directory*>*brd2odb_dir*
- (Linux) *BRD2ODB/brd2odb_dir*

VALOR_TMP

Placed in the *env_file* file by the installer. Specifies the location for storing temporary files. Set this to a local directory to improve performance on busy networks.

Default:

- (Windows) <*ODB++ Inside Installation Directory*>*brd2odb_dir**tmp*
- (Linux) *BRD2ODB/brd2odb_dir/tmp*

Configuration Parameters

Some aspects of translation are controlled by the values of configuration parameters. Default values can be used in most cases, but you can modify the parameter values to customize your environment.

A file named *config* in the *$VALOR_DIR/sys* directory supplies the default configuration parameter values when you run the translator for the first time. After that, another file with the same name is created in your user location *$VALOR_HOME/.genesis*. This newly created file is read on each subsequent run.

Parameters not defined in the user location are read from the system location.

If you store the *config* file in a non-default location, specify its path with the -cfg parameter when running ODB++ Inside from the command line. See “Command Line Parameters” on page 39.

You can edit the *config* file to set the appropriate values before launching the translator. If you change the file when the translator is running, the new values will not be used.

Each line of the *config* file has this format:

```
<parameter name>=<parameter value>
```

The following table lists the general configuration parameters and the configuration parameters that are specific to the Cadence Allegro translation.

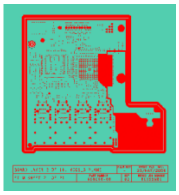
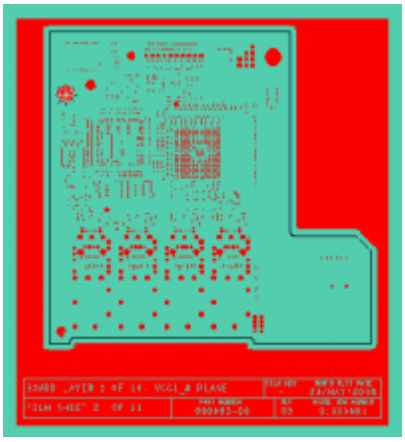
Configuration Parameter	Type	Default	Description
attr_value_correct	Boolean	yes	Controls how the translator handles illegal attribute values (out of range, etc.) that are input with a design. • **yes** — The attribute is reset to its default value, and a message is written to the log. • **no** — Translation is halted.
drc_comp_height	Text	drc_comp	Document layer containing component height restriction areas (used to check for components above or below height limits in height restricted areas).
drc_comp_keepin	Text	drc_comp	Document layer containing component keepin areas (to check for components outside keepin areas).
drc_comp_keepout	Text	drc_comp	Document layer containing component keepout areas (to check for components inside keepout areas).
drc_pad_keepout	Text	drc_route	Document layer containing pad keepout areas (to check for pads inside keepout areas).

Configuration Parameter	Type	Default	Description
drc_plane_keepout	Text	drc_route	Document layer containing plane keepout areas (to check for planes inside keepout areas).
drc_route_keepin	Text	drc_route	Document layer containing rout keepin areas (to check for traces, planes, pads, vias outside the keepin areas).
drc_route_keepout	Text	drc_route	Document layer containing rout keepout areas (to check for traces, planes, pads, trace-bends inside keepout areas).
drc_tp_keepin	Text	drc_tp	Document layer containing testpoint keepin areas (to check for testpoints outside keepin areas).
drc_tp_keepout	Text	drc_tp	Document layer containing testpoint keepout areas (to check for testpoints inside keepout areas).
drc_trace_keepout	Text	drc_route	Document layer containing trace keepout areas (to check for traces inside keepout areas).
drc_via_keepout	Text	drc_route	Document layer containing via keepout areas (to check for vias inside keepout areas).
eda_cadence_add_nets_from_geometry	Boolean	yes	Controls whether net information is read from the connectivity out file or from the geometry out file. **yes** — Work as before and take net information from the geometry file *geoms_<pm>.out*. (Default) **no** — Take net information from the connectivity file *conn_<pm>.out*.
eda_cadence_apd_bot_name	Text	base	The name used to indicate the bottom layer in APD files. When Cadence APD files are translated, the default name for the bottom layer is base. If you have APD files that use a layer name other than base for the bottom layer, you can specify an alternate name. Wildcard characters (*) are accepted. For example, *base*.
eda_cadence_apd_top_name	Text	surface	The name used to indicate the top layer in APD files. When Cadence APD files are translated, the default name for the top layer is surface. If you have APD files that use a layer name other than surface for the top layer, you can specify an alternate name. Wildcard characters (*) are accepted. For example, *surface*.
eda_cadence_check_package_shape	Boolean	no	If a design is likely to have components with identical package names but different geometries, you can have the translator check the geometry of a component if its package name is identical to that of another component.

Configuration Parameter	Type	Default	Description
			• **yes** — If a component has the same package name as another component, the package shapes in the file *comps_<product_model>.out* are compared. If they are not the same, a new package is created for the second component. The new name is created by adding a suffix consisting of a plus sign (+) and an index number. • **no** (default) — If a component has the same package name as another component, they are assumed to have the same geometry. no checking is performed.
eda_cadence_delete_sort_pins_file	Boolean	no	**yes** — Always delete temporary sort pins file. When a translation has warnings about the pins file, it does not delete the temporary pins file, so that the user can view the warnings. When running from a script, the temporary files accumulate. This parameter lets you specify that the files be deleted even if there are warnings.
eda_cadence_font_file_name	Text	ansi (the supplied font file)	The name of the font file to be used. The file must reside in $GENESIS_EDIR/all/eda/cadence/fonts. You can provide an alternate font file so that fonts used in ODB++ match the fonts used in Cadence Allegro.

eda_cadence_keep_auxiliary_layers_name	Boolean	no	Controls whether to translate the names of silk screen, solder paste, and solder mask layers.
			• **no** — Renames auxiliary layers based on their subclass in the *films_<pm>.out* file:
			  • “SILKSCREEN_TOP”, “sst”
			  • “AUTOSILK_TOP”, “sst”
			  • “SILKSCREEN_BOTTOM”, “ssb”
			  • “AUTOSILK_BOTTOM”, “ssb”
			  • “PASTEMASK_TOP”, “spt”
			  • “PASTEMASK_BOTTOM”, “spb”
			  • “SOLDERMASK_TOP”, “smt”
			  • “SOLDERMASK_BOTTOM”, “smb”
			• **yes** — Keeps the auxiliary layer names as defined in the *films_<pm>.out* file.

eda_cadence_layer_polarity_source	Text	f (film)	Informs the translator to use the suppress shape fill information from the *films_xxx.out* file or from the *layers._xxx.out* file.

Configuration Parameter	Type	Default	Description
			• **f** — films • **l** — layers (Cadence Allegro only)
eda_cadence_pos_anti_etch	Boolean	no	Controls whether ANTI ETCH surfaces will be negative or positive. • **yes** — ANTI ETCH surfaces will always be positive. (If the product model has a photoplot outline - positive surface - that covers legend text, the text will not be visible.)  • **no** (default) — ANTI ETCH surfaces will be negative. 
eda_cadence_profile_sym_type	Text	r (round)	Defines symbol type for arcs and lines of step profile polygon. • **r** — round symbol • **s** — square symbol
eda_cadence_read_dra_file	Boolean	no	Controls translation of DRA files. • **yes** — translate DRA files • **no** — do not translate DRA files
eda_cadence_silk_fill	Boolean	no	Fills surfaces on Cadence Allegro silkscreen layers. • **yes** — draws surfaces. • **no** — draws surfaces as outlines.
eda_cadence_sort_pins_file	Boolean	yes	Controls whether to sort the pins file before reading it.

Configuration Parameter	Type	Default	Description
			• **yes** — sorts the pins file alphabetically before it is read. • **no** — does not sort the pins file.
eda_cadence_sort_pins_numeric	Text	no	Controls how to sort pins. • **yes** — sorts the pins file numerically. • **no** — sorts the pins file textually. • **yes_num_last** (or any string other than yes or no) sorts the pins file numerically but with numeric pins after pins beginning with a letter.
eda_cadence_sqa_area_layer_name	String	SQA_areas	Defines the layer where an sqa area is saved during translation from Cadence. If not defined, default value sqa_areas are saved.
eda_cadence_support_exceptional_pins	Boolean	yes	To control whether to add to the **comps_XXX.out** file pins lacking names and component designation, or whose components do not appear in the file. • **yes** — (default) Adds a component named no_refdes+XX. • **no** — Ignore the pins.
eda_cadence_support_old_extract	Boolean	no	Controls whether to translate extract files that were created with a *valor_ext.il* version prior to 2305. • **yes** — Translate extract files regardless of their version. In rare cases, the logic used in old extract files may place some components on the wrong side of the board. • **no** — Translate only current extract files. The translation of outdated extract files fails.
eda_cadence_suppress	Boolean	no	Default value for option Suppress Unconnected Pads. • **yes** — Perform unconnected pad suppression. • **no** — Do not suppress unconnected pads.
eda_cadence_suppress_shape_fill_setting (obsolete)	Text	f	Obsolete - replaced by the Suppress shape fill option of the Cadence Matrix File Editor.
eda_cadence_therm_err	Boolean	no	• **yes** — The translation process will abort with a message listing the padstack / thermal names that did not have a match in the models file. (Thermals Mode in Input Parameters must be set to Use File for this configuration parameter to work). • **no** — Does not flag missing thermals.

Configuration Parameter	Type	Default	Description
eda_cadence_v14_popup (obsolete)	Boolean	yes	Obsolete.
eda_flex_material	Text		The name(s) of flexible dielectric materials. Multiple values are separated by semicolons (;). For example: POLYIMIDE;POLYIMIDE_FILM At import of Cadence Allegro data, the copper layers below and above a dielectric layer whose material definition in the *layers_<pm>.out* file matches one of the flexible dielectric material names, are assigned appropriate flex subtypes according to their base type. See “Subtypes to Support Flex/Rigid Flex Manufacturing” in *Getting Started With ODB++Design*.
gns_pdf_viewing_prog	Text		Default program path and arguments to open a PDF file. Used for standalone translators only.

Command Line Parameters

You can run ODB++ Inside for Cadence Allegro from the command line.

Syntax of Command Line Parameters

Usage: brd2odb [parameters]

Parameters are preceded by a dash. Some parameters accept values. Parameters must be separated by spaces. An unrecognized parameter is ignored.

If you are working in console mode (the -gui switch has not been set), missing or incorrect parameters cause the program to terminate.

ODB++ Inside Without the User Interface

To run the translator without displaying the user interface, provide these parameters:

- -ijp *<full path to input brd files>*
- -jp <*output path*>
- -jn <*output product model name*>

For example:

```
brd2odb   -ijp  C:\inputs\allegro\design_1   -jp  C:\my_odbs   -
jn  allegro_1
```

If any of these parameters are not provided, the GUI will open even if the -gui option has not been provided.

List of Command Line Parameters

For some command line parameters there is an equivalent GUI parameter indicated in the column Equivalent GUI Parameter. The GUI parameters are described in these sections:

- “Specifying File Options and Output Options Page” on page 12
- “Specifying Partial Export Parameters Page” on page 16
- “Specifying Additional Parameters Pages” on page 18

These are the command line parameters for ODB++ Inside for Cadence Allegro:

Parameter	Equivalent GUI Parameter	Description
-a2l -append2log		Appends log messages to existing log file *log_brd2odb*. By default, the new log file overwrites any existing log file for each translation.
-bb	Don’t suppress pads on top/bottom	Pads on top/bottom edge of blind/buried drills are not suppressed. Active only if -sp is set.
-c <*outline*> -component <*outline*>	Component Outline	Controls which geometries are used for the component outline. • **p[lacebound]** — (default and recommended) The bounding box. • **a[ssembly]** — The limits of the assembly features. • **d[fa]** — The DFA boundary. • **u[ser_defined]**<*top*> <*bottom*> — User defined.
-cfg [<*config file*>]		Read configuration file. If <*config file*> is not specified, the default name is *$VALOR_HOME/.genesis/config*.
-d -delete	Delete Extracted Files	Source extract files are deleted. By default, all intermediary files are saved. If the -gz (zip) parameter is used, the extract files are compressed.
-fi	Fully isolated pads	Only fully isolated pads are suppressed. Active only if -sp is set. Without this switch, pads are considered to be isolated in these cases: • a single totally isolated pad • two pads touching or intersecting • a pad transversed by a trace not through its center • a pad touching a surface where its center is not inside the surface
-gui		Starts the GUI version of the translator.

Parameter	Equivalent GUI Parameter	Description
		To run the translator without displaying the user interface, provide these parameters: • -ijp *<full path to input brd files>* • -jp *<output path>* • -jn *<output product model name>* If any of these parameters are not provided, the GUI will open even if the -gui option has not been provided.
-gz -gzip	Create Archive	Compress the ODB++ folder structure of the product model to create a single *tgz* file.
-help		Lists the help switches in the console window.
-hg -help_gui		Displays online help.
-iff	Ignore FIXFLAG	Suppression ignores the Allegro FIXFLAG setting. Active only if -sp is set.
-ijp *<path>*	Input Path	The full path to the input brd files. Default = the current working directory. This parameter is required if you want to run the translator without displaying the user interface.
-jn *<product_model>*	Output product model name	Output ODB++ product model name. Default = odbjob. This parameter is required if you want to run the translator without displaying the user interface. See “ODB++Design Entity Naming Rules” on page 31.
-jp *<product_model_path>*	Output Path	Output path for the ODB++ product model. Default = the current working directory. This parameter is required if you want to run the translator without displaying the user interface.
-lp *<log_path>*		Log file path. Default is output product model path.
-m *<tolerance>* -match *<tolerance>*	Symbol tolerance	Where *<tolerance>* is the number of mils for symbol tolerance. Default = 0.2 inches (200 mils).
-matrix_file *<matrix_path>*	Matrix File	The full path to the matrix file. The matrix file can only be edited from the GUI.
-net_none_flag	Read $NONE$ net	Does not assign the $NONE$ net to features with no net.
-nn -neut_nets	Keep Net names	Nets are renamed to generated numeric values. Default = Net names are kept.

Parameter	Equivalent GUI Parameter	Description
-no_view	Open ODB++ Viewer	Runs the translator without opening ODB++ Viewer after the translation has completed.
-noDPW	-	Suppresses automatic launching of the wizard.
-o <*dist*> -outline <*dist*>	Outline size	Where <*dist*> is the number of mils to extend the outline on negative planes. This parameter corresponds to the “-o” option of the Cadence Allegro artwork program. The default value is 0.1 inches (100.0 mils)
-odb_version	ODB++Design version to export job	The ODB++Design version in which to export. • **v8** — ODB++Design Version 8 (default). • **v7** — ODB++ Version 7.
-p <*mode*> -padflash <*mode*>	Padflash	Controls whether Allegro padflash codes are used for padstacks. • **s** (substitute) — substitute padflash definitions using the thermal model file • **i** (ignore) — (default) use the pad size
-p_assem	Export Option = ASSY	Only assembly data is to be written to ODB++ output.
-p_fab	Export Option = FAB	Only fabrication data is to be written to ODB++ output.
-pst <*option*> -profile_symbol_type <*option*>	Symbol type for lines and arcs	Defines the symbol type of lines and arcs describing step profile: r(ound)/s(quare).
-r -read_drc	Import Keepin/out regions	Triggers the generation of multiple DRC layers beginning with the prefix “drc_”. • Allegro layers Route keepout, Route keepin and Via keepout are used to generate an ODB++ layer called drc_route. • Allegro layers Package keepout, Package keepin, Component keepout and Component keepin are used to create drc_comp_top and drc_comp_bottom. • Allegro layers No_Probe_Top and No_Probe_Bottom are used to create drc_tp_top and drc_tp_bottom.
-ral [<*layer name*>] -rout_artwork_layer [<*layer name*>]	Create Rout From Artwork Layer	Controls how the rout is created: • If <*layer name*> contains the name of a valid layer, as specified in the Allegro artwork, the features in that layer are used to create an ODB++ rout layer with the original name. • If <*layer name*> is empty, or the value is not found in the *.out* files, the translation creates a rout layer by

Parameter	Equivalent GUI Parameter	Description
		merging the features from the following Allegro artwork CLASS/SUBCLASS: BOARD GEOMETRY:OUTLINE“&”BOARD GEOMETRY:DESIGN_OUTLINE“&”BOARD GEOMETRY:CUTOUT
-rc -round_corners	Round Corners	(Corners of polygons (contours) will be rounded.
-re -remove_eda	Remove EDA Data	Removes EDA component and package data.
-read_sqa <*option*>	Read SQA Data	Controls creation of the signal quality layer. • **yes** — Read SQA data and create an SQA layer. • **no** — (default) Do not read SQA data and create an SQA layer.
-rr <*option*> -read_region <*option*>	Import Areas-Constraint region	**yes** — Generates an ODB++ layer named fab_drc from the Allegro layer group “Constraint region”.
-rrd -remove_dielectric	Remove redundant dielectric	Combines successive dielectric layers.
-sf	Turn eda_cadence_silk_fill on	Set configuration parameter eda_cadence_silk_fill.
-skip_refdes <*option*>	Skip RefDes with Asterisk	Skip components with a RefDes containing an asterisk (*). • **no** — All components are translated. (default) • **yes** — Components with names containing an asterisk are not translated. • **part** — The translation excludes components whose RefDes contains an asterisk but includes their pad and drill features.
-sp	Suppress Unconnected Pads	Sets configuration parameter eda_cadence_suppress.
-te	Turn eda_cadence_therm_err on	Sets configuration parameter eda_cadence_therm_err.
-kal -keep_aux_layers	Keep auxiliary layers name as in artwork	Sets configuration parameter eda_cadence_keep_auxiliary_layers_name to Yes.
-tf <*thermal_file*>	Use thermal model file	Where <*thermal_file*> is the full path name for the thermal model file. If the thermal_file is specified, the thermal_model must be specified in -tm. If no thermal file is specified, a

Parameter	Equivalent GUI Parameter	Description
		default model is used, which uses direct connect and no thermals.
-tm <*thermal_model*>	Select Model button	Where <*thermal_model*> is the name of the model to be used. The available model names are defined in the thermal model file. If no thermal_file is specified, the thermal_model is ignored.
-tr_sym		Controls the translation of symbols. • **yes** — Translate symbols as components. If there are multiple shapes with the same name, each will be translated as a separate component. • **no** (default) — Do not translate symbols.
-up -use_panel	Use Panel Outline as profile	Creates the step profile from data with CLASS = BOARD GEOMETRY and SUBCLASS = PANEL_OUTLINE/DESIGN_OUTLINE/OUTLINE in the *geoms_<pm>.out* file.
-v -ver		Displays version information about the translator. When this option is used, all other parameters are ignored.
-verify		Requests verification from the user before performing various actions such as Save and Translate.

Thermal Model Configuration

Cadence Allegro Designer (Version 13) does not explicitly define the shape of the thermal pads or the Padflash definitions. Typically, these definitions are deferred until the Gerber wheel apertures are defined. However, to generate accurate board data, ODB++ Inside for Cadence Allegro requires the use of a Thermal Model to explicitly define these shapes.

Structure of the Thermal Model File
Thermal Model Examples

Structure of the Thermal Model File

The thermal model file contains a units statement and one or more model definitions. Each model describes one type of behavior. The file can be built in such a way that each model is customized for a specific customer, a product type, or an EDA system.

See “Thermal Model Examples” on page 48.

This is the structure of the file:

```
.units [inch|mm]
.model <name>
... model info ....
.model <name>
... model info ....
.model <name>
... model info ....
```

Lines starting with the number sign (#) are comments and are ignored.

Units Statement

The units directive must be the first line in the file.

```
.units [inch|mm]
```

It specifies the measurement units that will be used for the models.

- **inch** — 0.001 inch (mil) units.
- **mm** — 0.001 mm (micron) units.

Model Statement

Each model definition begins with the model directive.

```
.model <name>
```

The name is limited to 64 characters that can include letters, digits, and these characters: dash (-), underscore (_), period (.), plus (+).

Rule Statements

Each model definition consists of the model directive followed by a set of rules expressed in Backus-Naur Form (BNF). The rules are used for substitution of clearances to thermal pads. In the EDA system, a padstack or padflash always defines the shape of the clearance in the Power & Ground layer. It is the electrical net of the pin that determines whether the clearance will be retained or will be substituted by a thermal pad.

When the translator processes the data, it determines whether a particular clearance must be converted to a thermal relief pad. It is at this point that the model, with the name provided as a translation parameter, is consulted.

Each rule consists of a condition and a derivation.

<rule> ::= *<condition>* : *<derivation>*

If the condition is met, then the thermal shape described in the derivation is used for the pad. The first match is used.

Condition

The condition of a rule consists of the type of padstack or the name of the padstack, optionally followed by equations defining the clearance size or drill size that match the rule.

<condition> ::= *<type>* {*<equation>*}

Type

<type> ::= PIN | VIA | *<geometry_name>* | '*<geometry_name>*'

- PIN or VIA — Keyword indicating the type of padstack that matches the rule.
- *<geometry_name>* — Name of the padstack that matches the rule. This is the name of the padstack geometry used in the EDA system. For Cadence Allegro, the padflash definition is used.

  This name can be surrounded by quotation marks, accommodating the rare case of a padstack called PIN or VIA.

A model can contain some rules defined with PIN or VIA types and some rules defined with *<geometry_name>*.

Equation

<equation> ::= [D|C] ['<' | '<=' | '=' | '>' | '>='] *<value>*

The equations represent numerical checks for the drill size (D) or clearance size (C) with which the padstack is to be compared. These are some examples:

PIN C>80 — This matches pin padstacks with clearances greater than 80 units.

VIA C<=40 D<=12 — This matches via padstacks with clearances less than or equal to 40 units and drill less than or equal to 12 units.

Derivation

The rule derivation specifies the shape to be used. It can be a standard symbol or it can be based on the clearance and drill size and shapes.

<derivation>::= NULL | '*<sym_name>*' | *<set_values>*

NULL

The NULL keyword can be used to create direct connect. The clearance will be deleted completely without a thermal pad.

Symbol Name

The symbol name can be any legal ODB++ standard name, semi-standard name, or special symbol (such as 003, thr80x50x0x4x10).

<*sym_name*>::= Standard, semi-standard, or special symbol.

Set Values

If a standard symbol name does not provide enough flexibility, the derivation can be defined based on the clearance and drill size and shapes.

<*set_values*> ::= <*od*> <*id*> <*tie*> <*num_ties*> <*angle*> <*oshape*> <*ishape*> <*style*>

- <*od*> ::= [C+value | C-value | D+value | D-value | value]

  The outer diameter can be specified as a fixed value or as a value added or subtracted from the Clearance (C) or Drill (D) sizes.

- <*id*> ::= [C+value | C-value | D+value | D-value | value]

  The inner diameter can be specified as a fixed value or as a value added or subtracted from the Clearance (C) or Drill (D) sizes.

- <*tie*> ::= <*value*>

  The size of the tie.

- <*num_ties*> ::= <*value*>

  The number of ties.

- <*angle*> ::= <*value*>

  The start angle for the first tie in degrees.

- <*oshape*> ::= R | S | C

  The shape of the outer ring can be round (R), square (S), or the same shape as the clearance (C).

- <*ishape*> ::= R | S | C

  The shape of the inner ring can be round (R), square (S), or the same shape as the clearance (C).

- <*style*> ::= R | S

  The style of the thermal near the tie can be rounded (R) or squared (S).

Thermal Model Examples

These examples show a thermal model file with two models, a thermal model file using Cadence Allegro padflashes, and an example of a typical derivation statement.

Thermal Model File With Two Models

```
.units inch
#
.model std
# All via clearances with drill size less than 40 mils to be cleared.
# Other vias will have a thermal that is a function of the drill size.
#
VIA D>=40  : D+40 D+20 10 4 0 R   R   S
VIA         : NULL
#
# All pin clearances with clearance size less than 45 mils to be cleared.
# Other clearances will have a thermal that is a function of the
# clearance size, in two groups - Clearances equal to and above 165 mils,
# and clearances equal to or above 45 mils.
#
PIN C>=165 : C    C-30 15 4 0 C   C   S
PIN C>=45  : C    C-20 10 4 0 C   C   S
PIN         : NULL
#
.model symbols
#
# This model matches specific padstack names with fixed thermals. This is
# useful when the EDA system used a limited set of fixed names
#
D73: 'ths85x65x45x4x12'
D74: 'ths62x42x45x4x12'
D75: 'ths100x80x45x4x12'
D76: 'ths120x100x45x4x12'
D77: 'ths160x140x45x4x12'
```

Thermal Model File Using Cadence Allegro Padflashes

```
.units inch
.model allegro_model
# Direct replacement of symbols
# Replace the padflash named "TH05" with a round clearance of 5 mils.
TH05: 'r5'
# Replace the padflash named "T165X145X20X45" with a square thermal with
# an outer diameter of 165 mils, inner diameter of 145 mils
# with four ties each of 20 mils, first starting at 45 degrees.
T165X145X20X45: 'ths165x145x45x4x20'
# Replace the padflash named "5MIL" with a direct connection
5MIL: 'null'
# calculated values
```

```
# Place a direct connect for all VIA pads
VIA: NULL
# For pins with a clearance less than or equal to 45 mils,
# place a rounded thermal with outer diameter the size of the
# clearance inner diameter 20 mils smaller, 4 ties of 20 mil
# starting at 45 degrees outer and inner diameters shaped as
# the clearance
PIN C<=45 : C   C-20 15 4 45 C C R
```

Typical Derivation Example

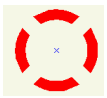
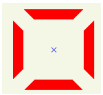
This is an example of typical derivations:

```
 C   C-20 15 4 45 C   C   S
```

The example specifies these values:

Parameter	Value	Description
od	C	The outer diameter is the same as the clearance.
id	C-20	The inner diameter is 20 units less than the clearance.
tie num_ties	15 4	There are 4 ties, 15 units each.
angle	45	Starting at 45 degrees.
oshape ishape	C C	Both rings are the same shape as the clearance.
style	S	The style is squared.

This specification will produce different thermals depending on the padstack:

Padstack	Symbol	Graphic
Padstack with round 80 mils clearance.	ths80x60x45x4x15	
Padstack with square 80 mils clearance.	s_ths80x60x45x4x15.	

Information Acquired from Cadence Allegro Data

Several types of Cadence Allegro design information stored in the *.out* files are read into the ODB++Design product model. In addition, you can configure the translation to extract specific data that was not imported automatically from a file stored in the base installation location or at an arbitrary path.

Generated Extract Files
Importing Allegro Geometry Properties
Importing Allegro Component Properties
Deriving Component Outline From Specific Subclasses
Cadence Allegro DFA Table

Generated Extract Files

These files, created by Cadence Allegro, contain information about the design.

If you are running ODB++ Inside from within Cadence Allegro, you can specify a *.brd* file as the input path. If you are running ODB++ Inside stand-alone, you must specify a directory containing the *.out* files that have been extracted from Cadence Allegro.

The extraction files are generated using *$ALLEGRO_BRD2ODB/valor_ext.il* skill code.

The variable *<pm>* in the file names represents the name of the current product model.

For CAD layers to be present in the generated ODB++, you must create the films for those layers.

File	Description

comps_<pm>.out	The components file contains the outline shape of all the components.
	The outline records are typically in subclass PLACE_BOUND_<*side*> but may also be in subclass ASSEMBLY_<*side*> or DFA_BOUND_<*side*>.
	The rows below define component CDN_220 that is based on package CAP-01005. The component is at 14.3160,35.9390, with no rotation. The LINE rows represent the outline of the component. The height information is read from PACKAGE_HEIGHT_MAX. The component is mounted body-up on layer ISL3.
	`S!ASSEMBLY_ISL3!3361 1!`
	`..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390!0.0`
	`00!GEOMETRY!LINE!14.1060!36.0490!`
	`14.5260!36.0490!0.0000!!!!!NOTCONNECT!!!YES!discrete1005`
	`!!!`
	`BODY_UP!!B2!ISL3!`
	`..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390!0.0`
	`00!GEOMETRY!LINE!14.5260!36.0490!`
	`14.5260!35.8290!0.0000!!!!!NOTCONNECT!!!YES!discrete1005`
	`!!!`
	`BODY_UP!!B2!ISL3!`

File	Description
	..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390!0.000!GEOMETRY!LINE!14.5260!35.8290! 14.1060!35.8290!0.0000!!!!!NOTCONNECT!!!YES!discrete1005!!! BODY_UP!!B2!ISL3! ..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390!0.000!GEOMETRY!LINE!14.1060!35.8290! 14.1060!36.0490!0.0000!!!!!NOTCONNECT!!!YES!discrete1005!!! BODY_UP!!B2!ISL3! ..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390!0.000!GEOMETRY!RECTANGLE!14.1060! 35.8290!14.5260!36.0490!1!!!!!!!0.22 MM!YES!discrete1005!!!BODY_UP!!B2!ISL3! ..CDN_220!CAP-01005!CDN_300!CAP-01005!14.3160!35.9390! 0.000!GEOMETRY!RECTANGLE!14.0460!35.8290!14.5860!36.0490!1! !!!!!!!YES!discrete1005!!!!!B2!ISL3! Components with BOM_IGNORE data are assigned attributes “Not Populated per BOM” (.no_pop) and “Ignore Graphically/Output” (.comp_ignore).
conn_<pm>.out	The connectivity file contains net connections. Net information can be read from this file or from the geometry file, depending on the setting of configuration parameter eda_cadence_add_nets_from_geometry: **eda_cadence_add_nets_from_geometry = Yes** — Work as before and get net information from the geometry file *geoms_<pm>.out*. (Default) **eda_cadence_add_nets_from_geometry = No** — Get net information from connectivity file.
crosssection_<pm>.out	The cross section file contains the impedance of the layer. To obtain this data, choose **show single impedance** ( ) in the Layout Cross section window of Cadence Allegro.
dfa_<pm>.out	For designs containing the old version of the DFA Spacing Table, this file stores the spacing requirements for various types of components. This information can be used during assembly analysis. See “Cadence Allegro DFA Table” on page 64
films_<pm>.out	The films file contains the artwork information from Allegro. This information describes the pieces of film to be output as artwork. Each piece of film is linked to an arbitrary number of subclasses and several parameters necessary to generate the correct physical layers. Typical films in the file are in this way: S!PIN!U0205!9!055C035!TOP!BOTTOM!EC_PRDB6!4925.00! 1050.00!!!4925.00!1050.00! ...PLATED!A!NULL!75.00!75.00!0.000!CIRCLE!4925.00! 1050.00!90.00!90.00!

File	Description
	``` S!VIA CLASS!!!VIA!TOP!BOTTOM!GND!!!4800.00!-100.00!4800.00!-10 0.00! ...PLATED!!CROSS!50.00!50.00!0.000!CIRCLE!4800.00!- 100.00!54.00!54.00! ``` Each line includes information about the film, and the location of the film. Pin lines contain component and pin number as well. Based on these lines, toeprints are added to components and drill holes are added to the appropriate drill layers. Lines with CLASS = BOARD GEOMETRY or EMBEDDED GEOMETRY, and the SUBCLASS value consisting of two parts separated by an underscore (“_”) are used to create solder mask, solder paste, and silk screen layers. The layer name is derived from the FILM_NAME field and its type is taken from the first part of the SUBCLASS field. The second part of the SUBCLASS field provides the name of the reference copper layer. As an example, the following line would be processed by creating an ODB++ layer named inner_2_soldermask with Type = solder_mask and Reference = isl2: ``` S!inner_2_soldermask!EMBEDDED GEOMETRY!SOLDERMASK_ISL2! 0!0.0000!0.0000!0.0100!0.2540!positive!no!no!no!yes!yes! no!yes! ``` Because an empty films file causes the translation to fail, the *valor_ext.il* import script checks whether the films file is empty and prompts you to check the artworks file and extract again.
geoms_<pm>.out	The geometry file contains graphical data describing feature placement. It is the largest file extracted. Each line of the file contains graphic data that describes the feature. The file also contains the class and subclass that can be mapped to the physical layer to which the feature is added. If the additional parameter “Create Rout From Artwork Layer” is not set with the name of a valid layer, a rout layer named “profile” is created from DESIGN_OUTLINE/OUTLINE data: • If DESIGN_OUTLINE data exists (Allegro 17.2 or later): BOARD GEOMETRY:DESIGN_OUTLINE“&”BOARD GEOMETRY:CUTOUT • If DESIGN_OUTLINE data does not exist (older versions): BOARD GEOMETRY:OUTLINE“&”BOARD GEOMETRY:CUTOUT Features defined in the following fields are added to the rout layer: • NCROUTE_PATH. • NCROUTE_PLATED. These features receive the attribute .rout_plated. The step profile is generated from lines with CLASS = BOARD GEOMETRY and SUBCLASS = PANEL_OUTLINE, DESIGN_OUTLINE, or OUTLINE, according to the setting of parameter Use Panel Outline, using the following order:

File	Description

	• If Use Panel Outline = Yes:
	a. PANEL_OUTLINE
	b. DESIGN_OUTLINE
	c. OUTLINE
	• If Use Panel Outline = No:
	a. DESIGN_OUTLINE
	b. OUTLINE
	c. PANEL_OUTLINE
	Data with SUBCLASS = CAVITY provides the layer profile. If no CAVITY exists, the step profile is used to define the layer profile.
	Data with SUBCLASS = CAVITY and GRAPHIC_DATA_10 = VOID is a layer profile hole. If configuration parameter eda_cadence_add_boundary_layer = yes, a documentation layer named boundary_<*layer_name*> is created from each line with CLASS = BOUNDARY:
	`S!BOUNDARY!L3!14442 1 0!!!!!!!!!LINE!495.5!1516.5!495.5!1651.0!0.0!!!!!SHAPE!!!!!!!!!!!!`
	Based on the above line, the layer boundary_l3 is created to which features are added as specified.
	Several lines are used to describe a polygon. These four lines represent a closed shape that is translated into one surface:
	`S!ETCH!GND!2261 1 0!!!!N_GND!!!!!!`
	`... LINE!8450.00!3250.00!8450.00!4650.00!0.00!!!!!SHAPE!`
	`S!ETCH!GND!2261 2 0!!!!N_GND!!!!!!`
	`...`
	`LINE!8450.00!4650.00!12350.00!4650.00!0.00!!!!!SHAPE!`
	`S!ETCH!GND!2261 3 0!!!!N_GND!!!!!!`
	`...LINE!12350.00!4650.00!12350.00!3250.00!0.00!!!!!SHAPE!`
	`S!ETCH!GND!2261 4 0!!!!N_GND!!!!!!`
	`...LINE!12350.00!3250.00!8450.00!3250.00!0.00!!!!!SHAPE!`
	A rectangle shape with CLASS = EMBEDDED GEOMETRY, SUBCLASS = SOLDERMASK_* or PASTEMASK_*, and GRAPHIC_DATA_10 = POLYGON is used to create a filled rectangular shape in the layer specified.
	The fields NET_PHYSICAL_TYPE and NET_SPACING_TYPE correspond to attributes of the same name.
	The shorted nets information for SMD pads is taken from the NET_SHORT field.
	This is a typical line in the file. This line adds a 6-mil line between (2.000,0.625) and (1.95,0.575) to a signal layer (SIG_2). The net of the line is CPUD9:
	`S!ETCH!SIG_2!7550 1!!!!CPUD9!!!!!!`
	`...LINE!2000.00!625.00!1950.00!575.00!6.00!!!!!CONNECT!`
	To translate additional data stored in the geometry file, see the following tasks:

File	Description
	• “Importing Allegro Component Properties” on page 60 • “Importing Allegro Geometry Properties” on page 58

layers_<pm>.out	The layers file describes the order of the physical layers in the design.
	The list includes the dielectric layers and the electric layers, but it does not include such layers as solder mask and silk screen.
	This is a typical line in the layers file:
	`S!5!SIG_1!POSITIVE!!YES!!595900 mho/cm!COPPER!NO!`
	`3.98 w/cm-degC!1.2 mil!`
	The system uses fields 2, 3, 4, and 12 which provide the relative order (5), name (SIG_1), polarity (POSITIVE), and layer dielectric thickness of the layer (1.2 mil).
	If a board is described, the system also relates to the board thickness. In this example of such a string (usually located at the beginning of the file) board thickness is 26.4 mil.
	`J!D:\home\allegro\hitachi.brd!Tue Oct 15 14:53:39`
	`2014!-100.000!-100.000!1100.000!800.000!0.001!`
	`millimeters!!26.4 mil!22!OUT OF DATE!`
	The copper layers below and above the dielectric layer whose LAYER_MATERIAL definition matches one of the values of the configuration parameter eda_flex_material, are assigned the appropriate flex subtypes. See “Subtypes to Support Flex/Rigid Flex Manufacturing” in the *Getting Started With ODB++Design*.

nets_<pm>.out	The nets file contains information about net classes and properties. This file is optional. • **Classes** — Allegro declares three types of class: spacing, physical, and electrical. Every net may connect or have any combination of triplet of spacing, physical, and electrical classes, if any. The classes are defined in the technology file. • **Properties** — The file contains net properties, such as impedance. A net with property NO_TEST = Yes will have attribute testpoint_count = 0.

pads_<pm>.out	The pads file contains information about the padstacks used in the product model.
	Padstack information is required to derive the drill size at any given pin or via. It is also necessary to know which thermal is required when the pin or via has the same net as the containing surface.
	This is a typical line in the pads file:
	`S!C55N067!00014!~DRILL!o!!67.00!`
	`125.00!125.00!0.00!0.00!CIRCLE!N!J!....`
	This line specifies that padstack C55N067 has a 67 mil drill at the center (0,0) of the padstack.

padstacks_<pm>.out	The padstacks file contains additional padstack information.

File	Description
	Each line of the file specifies the padstack. The value in the Usage field is stored in the appropriate Valor NPI attribute: • DIE_PAD — .bump_pad • MOUNTING_HOLE — .mount_hole • BOND_FINGER — .pad_usage=bond_finger • FIDUCIAL — .pad_usage=g_fidutial • TOOLING_HOLE — .pad_usage=tooling_hole, .tooling_hole (each used by different analysis actions) Features with the value of LASER in the drillNonStandard field receive the attribute .via_type=laser.

pins_<pm>.out	The pins file contains information about pins (toeprints) and vias.
	Each line contains information about the padstack, net, drill figure, and character added to the legend document. It also includes the location of the pin or via. Pin lines contain component and pin number as well.
	`S!PIN!U0205!9!055C035!TOP!BOTTOM!EC_PRDB6!4925.00!`
	`1050.00!!!4925.00!1050.00!`
	`...PLATED!A!NULL!75.00!75.00!0.000!CIRCLE!4925.00!`
	`1050.00!90.00!90.00!`
	`S!VIA CLASS!!!VIA!TOP!BOTTOM!GND!!!4800.00!-`
	`100.00!4800.00!-100.00!`
	`...PLATED!!CROSS!50.00!50.00!0.000!CIRCLE!4800.00!-`
	`100.00!54.00!54.00!`
	Based on these lines, Valor NPI can add toeprints to components and add drill holes to the appropriate drill layers.
	This additional information is included:
	• Information on slots — Supports functionality added in V15.2.
	• DRILL_HOLE_POSTOL and DRILL_HOLE_NEGTOL — Maximum and minimum drill tolerance values support drill tolerances added in V.15.2.
	• DRILL_ARRAY_LOCATION — Supports the Multiple/Plural Drill function added in V14.1.
	• BACKDRILL_SIZE — If no value exists, the drill padstack definition is the backdrill size.
	• BACKDRILL_BOTTOM_FROM, BACKDRILL_BOTTOM_LAYER, BACKDRILL_TOP_FROM, BACKDRILL_TOP_LAYER — Start layer number from the bottom or top of the design and the ending layer number from the bottom or top of the design (cut layer). The backdrill span is created from BACKDRILL_TOP_FROM to BACKDRILL_TOP_LAYER or from BACKDRILL_BOTTOM_LAYER to BACKDRILL_BOTTOM_FROM.
	• BACKDRILL_TOP_MAX_DEPTH and BACKDRILL_BOTTOM_MAX_DEPTH — The maximum allowable drill depth, as stored in the layer attribute .backdrill_max_depth.
	• BACKDRILL_TOP_MNCLAYER and BACKDRILL_BOTTOM_MNCLAYER — The index number of the "Must Not Cut" layer, counted from top down

File	Description
	starting from 0, regardless of the drill direction. The name of "Must Not Cut" layer is stored in the layer attribute .backdrill_penetrate_stop_layer. • BACKDRILL_TOP_MAX_STUB and BACKDRILL_BOTTOM_MAX_STUB — The maximum allowable PTH stub length. Features with this data are assigend attribute .backdrill_max_stub_drill. • EMBEDDED_LAYER and EMBEDDED_STATUS — The value of EMBEDDED_LAYER is stored in NPI attribute .placement_layer. EMBEDDED_STATUS = BODY_UP (top) or BODY_DOWN (bottom). • DRILL_TOP_NAME!DRILL_BOTTOM_NAME — These fields provide the drill span. If no values exist, the drill span is taken from START_LAYER_NAME!END_LAYER_NAME in the pinsside file. If pins in the pins file refer to a component not found in *comps_<pm>.out*, the translator performs one of these actions: • If an existing package name is found, the value in COMP_PACKAGE in the pins file is used as the package name. • If a package is not found, a bounding box around the pins is regarded as the outline of the package. The value in COMP_PACKAGE in the pins file is used as the package name. • If the reference in the pins file has no package, the pins are ignored. The NET_SHORT field contains intentional short data for pins and vias. The value is a colon (:) separated list of shorted nets. If a pin or via location is also a test point location, the TEST_POINT field contains a value of TOP or BOTTOM to reference the documentation layer on which the test point shape is placed. The blank field indicates that the location is not a test point. The PROBE_FIGURE field defines the test point shape as TRIANGLE, SQUARE, HEXAGON X, HEXAGON Y, OCTAGON, DIAMOND, OBLONG X, OBLONG Y, or RECTANGLE. If the value is “RECTANGE” or contains “X” or “Y”, the fields GHAPHIC_DATA_3 and GHAPHIC_DATA_4 contain the width and height of the shape. Otherwise, the upper boundary for drawing the shape is determined by adding half of the smaller in width or height (GHAPHIC_DATA_3 and GHAPHIC_DATA_4) to the Y location (GHAPHIC_DATA_2).

pinsside_<pm>.out	The pinsside file is used to establish the side on which the component is placed. This is the syntax of a line of the file:
	`A!CLASS!REFDES!START_LAYER_NAME!END_LAYER_NAME!`
	`SYM_MIRROR!EMBEDDED_STATUS!`
	If the component is an embedded component, the component side is taken from the value of EMBEDDED_STATUS. BODY_UP indicates top, and BODY_DOWN indicates bottom.
	If all pins are thru-hole, the component side is determined by the values of START_LAYER_NAME and END_LAYER_NAME, and the SYM_MIRROR flag is checked.
	If there is even one SMT pin, the layer on which it is located determines the side.
	If the START_LAYER is not the top or bottom layer, but if it is the top or bottom layer of any zone, the start layer is treated as an outer layer.

File	Description
props_<pm>.out	The properties file contains additional component property information. This file is optional. This allows users to read additional component properties directly into ODB++. Users requiring the extraction of additional properties can add them manually to the view file.
regions_<pm>.out	The regions file contains information about regions.
tech_<pm>.out	The technology file is an ASCII file containing Allegro or APD parameter and constraint data. This file is optional. You can use this file to apply a uniform set of design rules and constraints to multiple designs: • User Units • Drawing Parameters • Layout Cross Section Parameters • Spacing Constraints (including clearance rules) • Net Type Clearances (to extend the scope of Signal Quality Analysis) • Physical Constraints • Electrical Constraints • User Property Definitions From Cadence Allegro version 16.0, tech files are generated in XML format. ODB++ Inside can read either format. When the new version of the DFA Spacing Table is used, the technology file stores the spacing requirements for various types of components. This information can be used during assembly analysis. See "Cadence Allegro DFA Table" on page 64
zone_<pm>.out	The zone file contains Cadence Allegro zone information. Contours are taken from the geoms file, from the sub class ZONE_OUTLINE, according to the zone name in the geoms file. The translation creates these layers: • **zone_outline** — Document layer of subtype misc that contains the zone data in the form of lines, arcs, and text, as defined in Cadence Allegro. • **flex_area** — Mask layer of subtype flex_area that contains pattern-filled surfaces representing the zones spanning only copper layers of subtype signal_flex. The signal_flex subtype is assigned automatically if the values of configuration parameter eda_flex_material match the dielectric LAYER_MATERIAL definitions in the layers file. • **rigid_area** — Mask layer of subtype rigid_area that contains solid surfaces representing the zones spanning only copper layers of subtypes other than signal_flex.

Importing Allegro Geometry Properties

You can import geometry properties from the Cadence Allegro design, and store their values in ODB++ user-defined attributes. To do so, you need to create a file that lists the subclasses to extract, and then create a user attribute for each of those subclasses.

Prerequisites

The subclasses associated with the geometry properties that you want to import are stored in the *geoms_<pm>.out* file. See “Generated Extract Files” on page 50.

Procedure

1. Create a file from which to extract the names of the geometry property subclasses:

   a. In a text editor, create a list of the desired subclasses, specifying each subclass on a separate line.

      For example, to import the “Alert” subclass, include this line:

      ```
      Alert
      ```

   b. Do one of the following:

      - Save the list to this file:

        $ALLEGRO_BRD2ODB/added_comp_properties.txt

        Where *$ALLEGRO_BRD2ODB* is the directory where the Siemens product integrated with Cadence Allegro is installed, typically:

        ◦ Valor NPI:

          C:\MentorGraphics\Valor\vNPI_<ver>\edir\get

        ◦ ODB++ Inside:

          C:\MentorGraphics\ODB++_INSIDE_CADENCE_ALLEGRO/brd2odb_<ver>

      - Save the list as a text file under the name and in the location of your choice.

   c. If you saved the geometry properties file to an arbitrary path, set the system environment variable ALLEGRO_GEOM_PROP_BRD2ODB with the full path to this file, including the file name: the subclasses listed here will be extracted during translation.

   Note:
   When the environment variable ALLEGRO_GEOM_PROP_BRD2ODB is set with an explicit path to the component properties file, the *$ALLEGRO_BRD2ODB/added_comp_properties.txt* file is ignored, if it exists.

2. Map the specified geometry property subclasses to ODB++ user attributes:

a. In a text editor, open the user attributes file as appropriate for your product:

- Valor NPI:

  $VALOR_DIR/fw/lib/misc/userattr

- ODB++ Inside:

  $ALLEGRO_BRD2ODB/fw/lib/misc/userattr

b. For each subclass listed in the geometry properties file, create a user attribute with the appropriate definitions:

- The Data Type must make logical sense:
  - Text — TEXT
  - True/False — BOOLEAN
  - Integer — INTEGER
  - Float — FLOAT
- The NAME must be identical to the subclass name, only in lowercase and with an underscore character (_) as a prefix.

  For example, if the subclass name in the geometry file is “Alert,” the NAME definition should be “_alert.”

- The ENTITY must be “feature.”

The attribute definitions in the following example capture the “Alert” subclass:

```
TEXT {
  NAME=_alert
  PROMPT=Alert
  MIN_LEN=0
  MAX_LEN=100
  ENTITY=feature
  DEF=
  GROUP=Allegro
  OPTIONS=
  DEF_OPT=
}
```

If the attribute definitions are correct for the subclasses specified, a connection is established during translation and the appropriate ODB++ user attributes are assigned to features with the values as defined in Allegro.

Importing Allegro Component Properties

You can import component properties from the Cadence Allegro design as ODB++Design component properties or as user-defined attributes. In each case, you need to create a file that lists the subclasses to be extracted during translation.

Prerequisites

The subclasses associated with the component properties that you want to import are stored in the *geoms_<pm>.out* file. See "Generated Extract Files" on page 50.

Procedure

1. Create a file from which to extract the names of the component property subclasses:

   a. In a text editor, create a list of the desired subclasses, specifying each subclass on a separate line.

      For example, to import the "Description" subclass, include this line:

      ```
      Description
      ```

   b. Do one of the following:

      - Save the list to this file:

        $ALLEGRO_BRD2ODB/added_comp_properties.txt

        Where *$ALLEGRO_BRD2ODB* is the directory where the Siemens product integrated with Cadence Allegro is installed, typically:

        - Valor NPI:

          C:\MentorGraphics\Valor\vNPI_<ver>\edir\get

        - ODB++ Inside:

          C:\MentorGraphics\ODB++_INSIDE_CADENCE_ALLEGRO/brd2odb_<ver>

      - Save the list as a text file under the name and in the location of your choice.

   c. If you saved the component properties file to an arbitrary path, set the system environment variable ALLEGRO_COMP_PROP_BRD2ODB with the full path to this file, including the file name: the subclasses listed here will be extracted during translation.

Note:
When the environment variable ALLEGRO_COMP_PROP_BRD2ODB is set with an explicit path to the component properties file, the *$ALLEGRO_BRD2ODB/added_comp_properties.txt* file is ignored, if it exists.

Tip
If you want the data in the specified subclasses to be imported to ODB++Design as component properties rather than component attributes, you are done and do not need to continue with the rest of this procedure.

2. (Optional) Map the specified component property subclasses to ODB++ user attributes:

   a. In a text editor, open the user attributes file as appropriate for your product:

   - Valor NPI:

     $VALOR_DIR/fw/lib/misc/userattr

   - ODB++ Inside:

     $ALLEGRO_BRD2ODB/fw/lib/misc/userattr

   b. For each subclass listed in the component properties file, create a user attribute with the appropriate definitions:

   - The Data Type must make logical sense:
     - Text — TEXT
     - True/False — BOOLEAN
     - Integer — INTEGER
     - Float — FLOAT
   - The NAME must be identical to the subclass name, only in lowercase and with an underscore character (_) as a prefix.

     For example, if the subclass name in the geometry file is “Description,” the NAME definition should be “_description.”

   - The ENTITY must be “component.”

   The attribute definitions in the following example capture the “Description” subclass:

```
TEXT {
  NAME=_description
  PROMPT=Description
  MIN_LEN=0
  MAX_LEN=100
  ENTITY=component
  DEF=
```

```
  GROUP=Allegro
  OPTIONS=
  DEF_OPT=
}
```

If the attribute definitions are correct for the subclasses specified, a connection is established during translation and the appropriate ODB++ user attributes are assigned to components with the values as defined in Allegro.

Deriving Component Outline From Specific Subclasses

You can import data in specific component subclasses as the component outline. To do so, you need to create a file that lists the subclasses to extract, and then specify those subclasses in the Additional Parameters dialog box during translation.

Prerequisites

To be properly associated with the packages on the board, the component subclasses must be part of the Package Geometry class and must be added at the library level.

Procedure

1. Using a text editor, specify the two subclasses in which the top and bottom component outlines are stored, as defined in Allegro. Put each name on a separate line, for example:

   ```
   DISPLAY_TOP
   DISPLAY_BOTTOM
   ```

2. Do one of the following:

   - Save the list to this file:

     $ALLEGRO_ BRD2ODB/added_comp_subclasses.txt

     Where *$ALLEGRO_ BRD2ODB* is the directory where the Siemens product integrated with Cadence Allegro is installed, typically:

     - Valor NPI:

       C:\MentorGraphics\Valor\vNPI_<ver>\edir\get

     - ODB++ Inside:

       C:\MentorGraphics\ODB++_INSIDE_CADENCE_ALLEGRO/brd2odb_<ver>

   - Save the list as a text file under the name and in the location of your choice.

3. If in the previous step you saved the component subclasses file to an arbitrary path, set the system environment variable ALLEGRO_COMP_SUBCLASSES_BRD2ODB with the full path to this file, including the file name: the subclasses listed here will be used during translation.

Note:
When the environment variable ALLEGRO_COMP_SUBCLASSES_BRD2ODB is set with an explicit path to the component subclasses file, the *$ALLEGRO_ BRD2ODB/added_comp_subclasses.txt* file is ignored, if it exists.

Results

Running the translation with the additional parameter Component Outline = User Defined and the names of the subclasses in the component subclasses file specified in the User Defined Top and User Defined Bottom fields sets the component outline with the data in those subclasses.

Examples

Figure 6. Setting Component Outline to DISPLAY_TOP and DISPLAY_BOTTOM (Subclasses) in Valor NPI

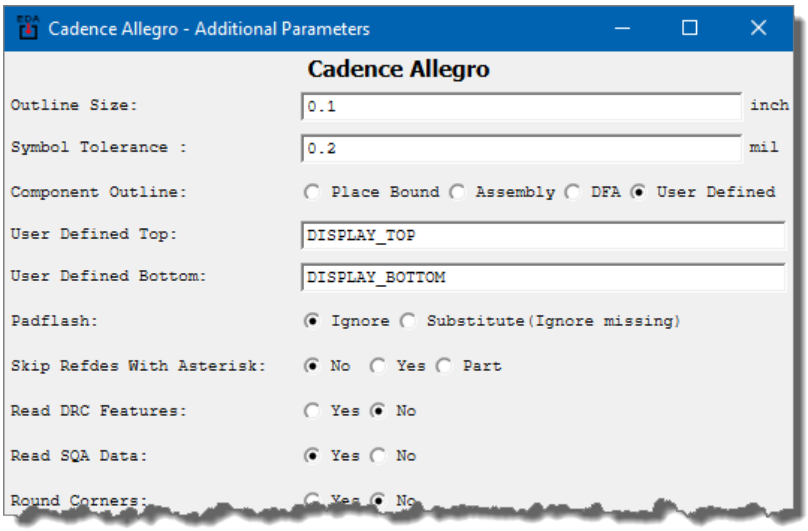
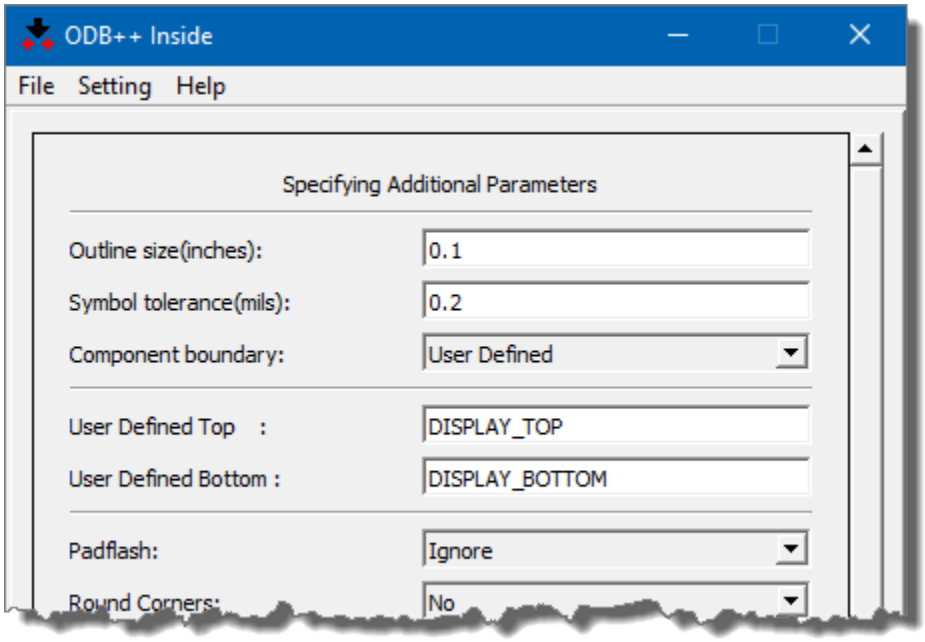


Figure 7. Setting Component Outline to DISPLAY_TOP and DISPLAY_BOTTOM (Subclasses) in ODB++ Inside



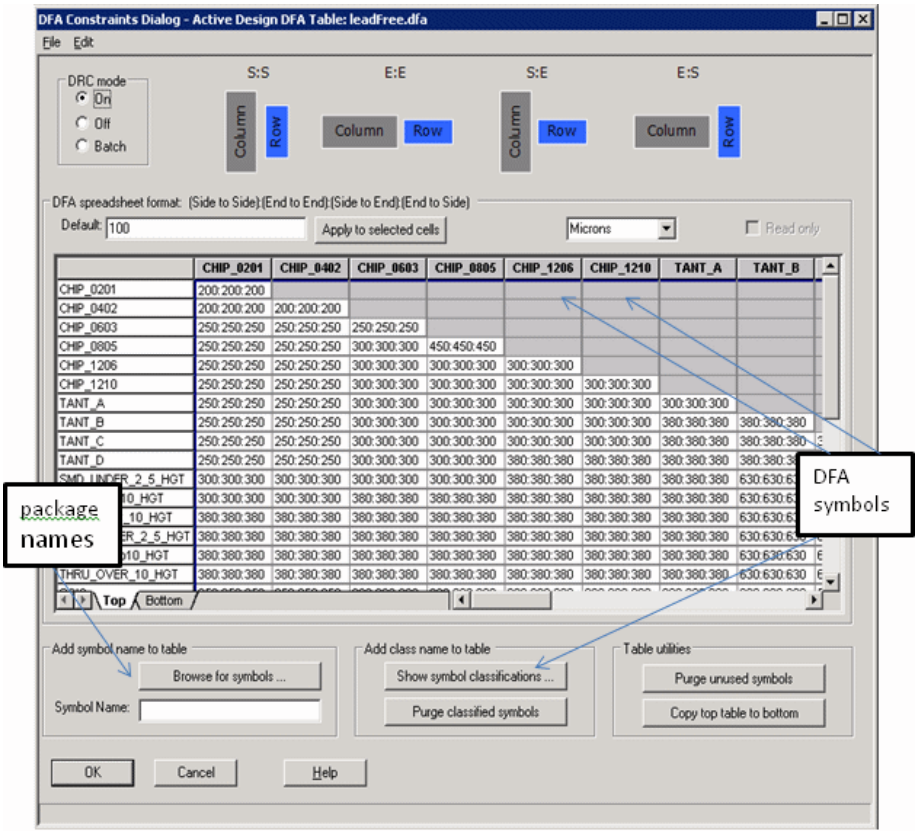
Cadence Allegro DFA Table

The DFA table defines the distance required between different types of components. If the design contains this table, the information is extracted according to the Cadence Allegro version, and then imported into the product model.

DFA Table Versions Prior to 17.4

Access: Click **DFA** ( ) in the toolbar.

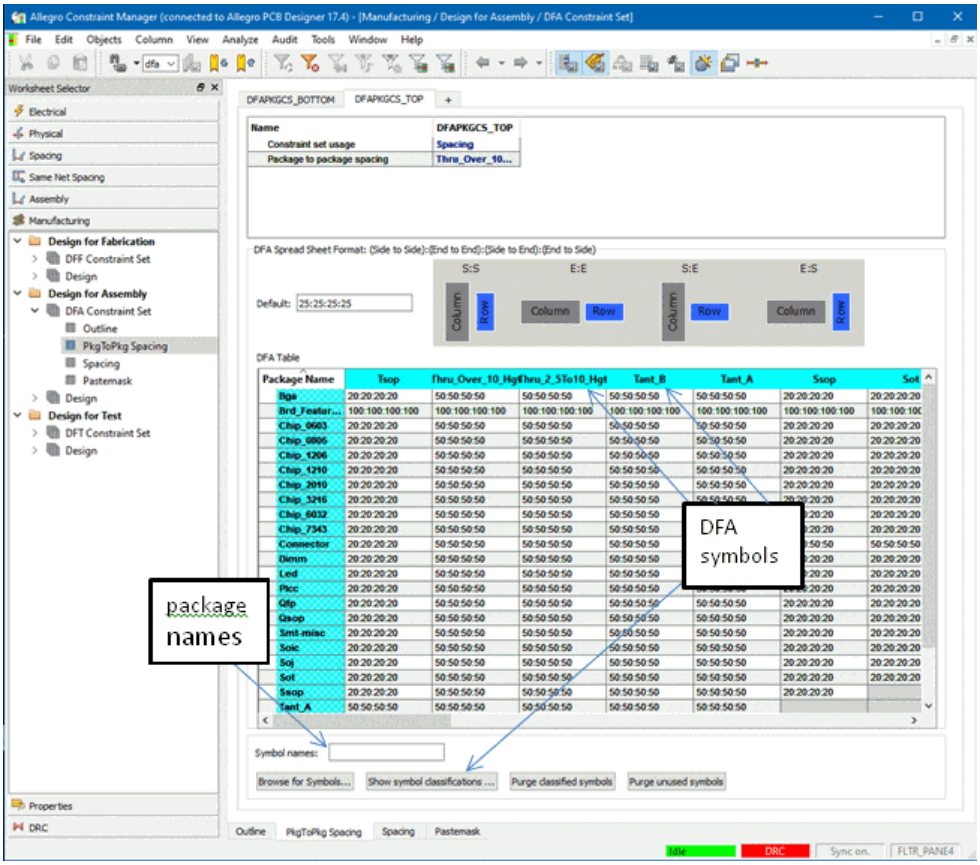
Data extracted to: *dfa_<pm>.out*.



DFA Table Versions 17.4 and Newer

Access: In the Worksheet Selector pane, choose **Manufacturing > Design for Assembly > DFA Constraint Set > PkgToPkg Spacing**.

Data extracted to: *tech_<pm>.out*.



Related Topics

Generated Extract Files

Supported Features

These features are included in the translator.

Support for Backdrill Depth, Stub Length, and Must Not Cut Layer
Support for Improved Component Placement
Support for Mask Layers Associated With Inner Copper Layers
Support for Padstack Types and Usage
Support for Test Point Shapes
Support for Intentional Shorts
Support for Components Excluded From BOM
Support for DFA Boundaries
Support for Partial ODB++Design Output
Support for Flex Subtypes
Support for Boundary Elements
Support for Backdrill Size
Support for Dielectric Layer Subtypes
Support for Bend Areas
Support for Skipping Extraction of Net Impedance Average
Package Height Properties
Support for CLASS_CONSTRAINT_REGION
Support for Translating Back-Drill Information
Support for Mirrored Padstacks
Support for the COMPONENT KEEPOUT Class

Support for Backdrill Depth, Stub Length, and Must Not Cut Layer

Backdrill data is read in from the *pins_<pm>.out* file and the associated attributes are added to the design.

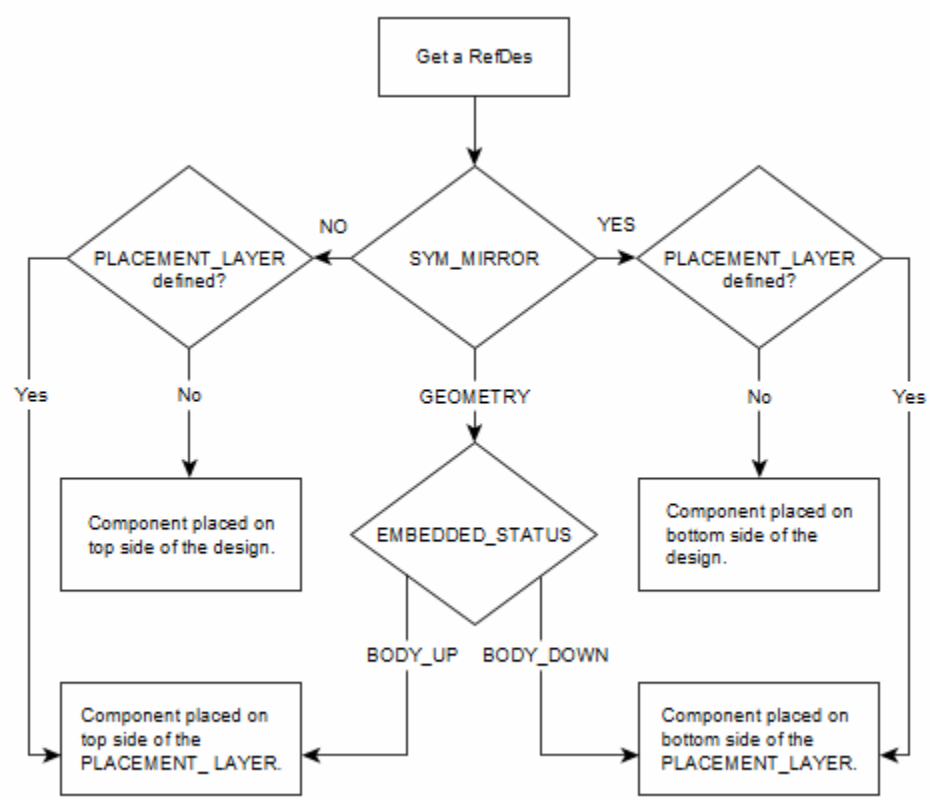
See “Generated Extract Files” on page 50.

Support for Improved Component Placement

The PLACEMENT_LAYER field has been added to the *comps_<pm>.out* file, with the value specifying the name of the copper layer on which the component should be placed.

For each value in the REFDES field, the translator uses the data in the SYM_MIRROR, PLACEMENT_LAYER, and EMBEDDED_STATUS fields to determine the layer on which a component should be mounted and the orientation of the package relative to the placement layer. If the REFDES field is empty, the value is generated automatically.

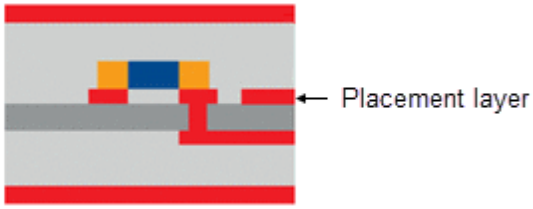
Figure 8. Component Placement Logic



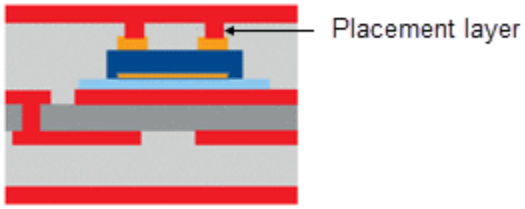
- SYM_MIRROR = NO and PLACEMENT_LAYER is undefined — The component is placed on the outer copper layer on the top side of the design.

- SYM_MIRROR = YES and PLACEMENT_LAYER is undefined — The component is placed mirrored on the outer copper layer on the bottom side of the design.

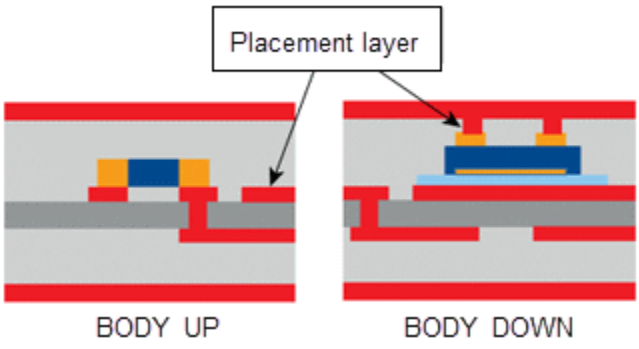
- SYM_MIRROR = NO and PLACEMENT_LAYER is defined — The component is placed on the top side of the PLACEMENT_LAYER.



- SYM_MIRROR = YES and PLACEMENT_LAYER is defined — The component is placed on the bottom side of the PLACEMENT LAYER.



- SYM_MIRROR = GEOMETRY and PLACEMENT_LAYER is defined — The component is placed on the inner copper layer specified in PLACEMENT_LAYER, in the orientation specified in the EMBEDDED_STATUS field as BODY_UP or BODY_DOWN so that the package is located on the top or bottom side of the placement layer, respectively.



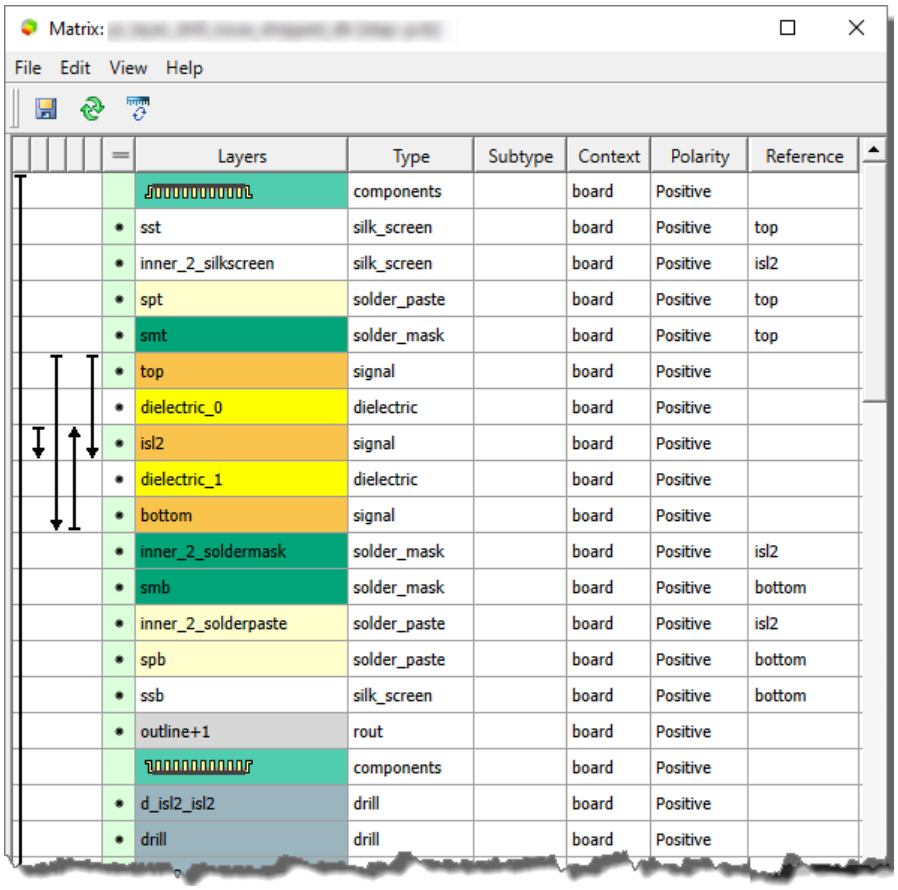
See “Generated Extract Files” on page 50.

Support for Mask Layers Associated With Inner Copper Layers

The Type, Context, and Reference parameters of the solder mask, solder paste, and silk screen layers associated with inner copper layers are now set in a post-process function, based on the definitions in the films file.

In the resulting ODB++Design matrix, these layers are grouped by type and positioned above or below the copper layers, according to the side of their associated components. The order of layers in a group reflects the order of their reference copper layers.

Figure 9. ODB++Design Matrix With Inner Solder Paste, Silk Screen, and Solder Mask Layers



Inner solder mask and solder paste layers connected to components on both sides are placed on the top side of the board.

The side of the inner silk screen layers for which no component reference exists is determined by the Mirrored flag in the EDA data.

See “Generated Extract Files” on page 50.

Support for Padstack Types and Usage

Hole type and padstack usage data is read in from the *padstacks_<pm>.out* file and the associated attributes are added to the design.

See “Generated Extract Files” on page 50.

Support for Test Point Shapes

The translator now places test point shapes on dedicated top and bottom documentation layers that have the same name as in Cadence Allegro.

The location and graphical representation of a test point shape are read into the TEST_POINT and PROBE_FIGURE fields of the *pins_<pm>.out* file.

See “Generated Extract Files” on page 50.

Support for Intentional Shorts

The translator now reads in intentional shorts data from Cadence. Known shorts are not reported as violations in Valor NPI Netlist Analyzer.

Shorted nets information is extracted to the NET_SHORT field of the *pins_<pm>.out* and *geoms_<pm>.out* files, where the former provides the values for pins and vias, and the latter for SMD pads. During translation, these values are used to define the intentional short instances in the *<step_name>/eda/shortf* file.

See “Generated Extract Files” on page 50.

Support for Components Excluded From BOM

The translator supports components marked as “BOM ignored” in Cadence Allegro.

Components with BOM_IGNORE data are assigned the following attributes during translation:

- **Not Populated per BOM** (.no_pop) — Designates the component as being not populated for the current version of the BOM.
- **Ignore Graphically/Output** (.comp_ignore) — Designates the component as to be ignored when calculating statistics, or during certain operations, such as analysis.

See “*comps_<pm>.out*” file “Generated Extract Files” on page 50.

Support for DFA Boundaries

The translator supports the DFA_BOUND_TOP and DFA_BOUND_BOTTOM sub-classes for the PART GEOMETRY class.

You can configure the translation to derive the component outline from the respective fields of the components file by setting the “Component Outline” parameter as described in “Specifying Additional Parameters Pages” on page 18.

Support for Partial ODB++Design Output

The translator supports partial ODB++ output.

ODB++ Inside for Cadence Allegro now supports partial output based on a layer groups list. See “Specifying Partial Export Parameters Page” on page 16.

Support for Flex Subtypes

The translator supports flex subtypes.

The copper layers below and above the dielectric layer whose LAYER_MATERIAL definition in the *layers_<pm>.out* file matches one of the flexible dielectric material names listed for the configuration parameter eda_flex_material, are assigned the appropriate flex subtypes. See “Subtypes to Support Flex/ Rigid Flex Manufacturing” in *Getting Started With ODB++Design*.

See “Generated Extract Files” on page 50.

Support for Boundary Elements

The translator reads in boundary elements to a documentation layer.

Boundary elements are surface areas used to create copper etch automatically within Allegro. If configuration parameter eda_cadence_add_boundary_layer = yes, a documentation layer is created for each record with in the *geoms_<pm>.out* file with CLASS = BOUNDARY, using this naming convention: *boundary_<layer_name>*.

Support for Backdrill Size

The translator supports backdrill size.

Backdrill size is read from the corresponding field in the *pins_<pm>.out* file. If this field does not exist, the drill padstack definition is the backdrill size. See “Generated Extract Files” on page 50.

Support for Dielectric Layer Subtypes

The translator supports Layer Subtypes for prepreg and core dielectric layers.

Layer Subtype for dielectric layers is read from the LAYER_FUNCTION field of the *layers_<pm>out* file:

- **LAYER_FUNCTION = DIELECTRIC_CORE** — Layer Subtype = core
- **LAYER_FUNCTION = DIELECTRIC_PREPEG** — Layer Subtype = prepreg
- **LAYER_FUNCTION ≠ DIELECTRIC_CORE, DIELECTRIC_PREPEG** — No Layer Subtype is provided

Support for Bend Areas

The translator supports bend areas.

If the Cadence Allegro design contains bend areas, this information is stored in the ODB++ product model in a layer named bend_area. This is a positive board layer of type mask and subtype bend_area. Bend area information is used in rigid flex analysis.

Support for Skipping Extraction of Net Impedance Average

During export from Allegro, the attribute NET_IMPEDANCE_AVERAGE is calculated for each net.

The *valor_ext.il* import script prompts for permission to skip this time consuming calculation, if the information is not required. As a result, extraction time is reduced.

Package Height Properties

Cadence Allegro properties package_height_min and package_height_max are interpreted to match their meaning in Cadence Allegro.

The usage of each of the properties depends on whether the other property is defined.

- **package_height_max** — The height of the component.

  This is stored in the ODB++ component attribute .comp_height (Height).

  The property package_height_max is not considered when package_height_min is specified.

  If package_height_min is not specified, package_height_max is used to indicate whether all components or no components can be placed in the area, regardless of their height:

  - **package_height_max = 0** — Any components can be placed in this area.
  - **package_height_max0** — No components can be placed in this area.

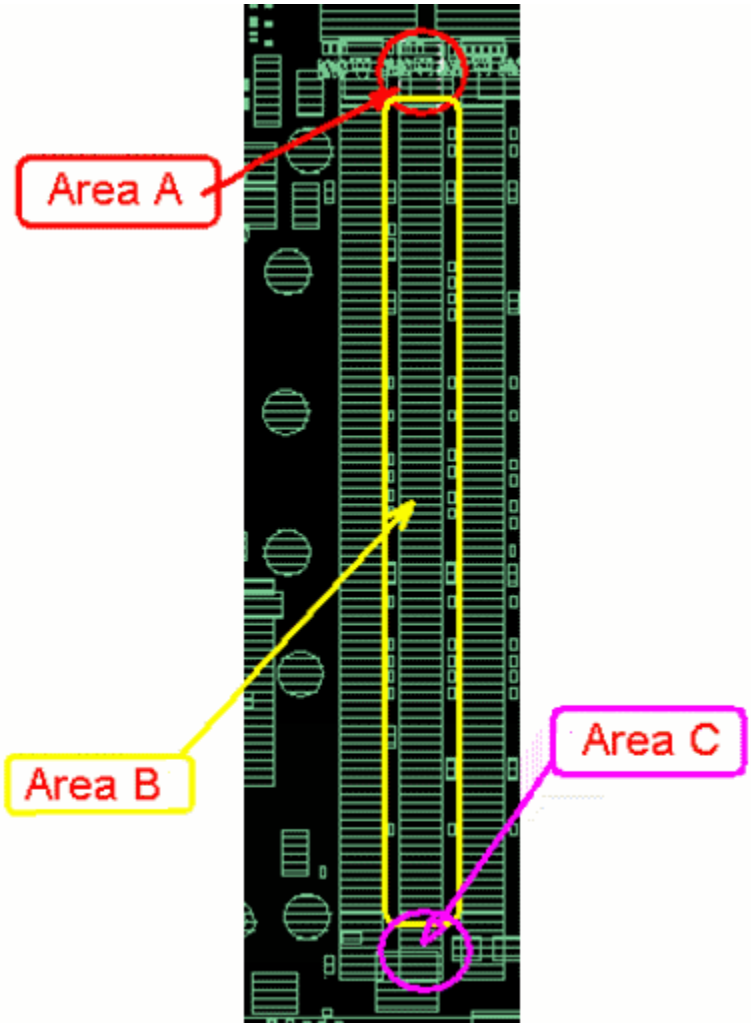
- **package_height_min** — The amount of space under the component. This is the lower limit of the height of the keepout area. If this value is specified for a keepout area, only components with height less than this value can be placed in this area.

  If a component is defined in Allegro as having a value for property package_height_min or if there are areas of the component with values for package_height_min, this information is stored with the product model, and can be used during component analysis.

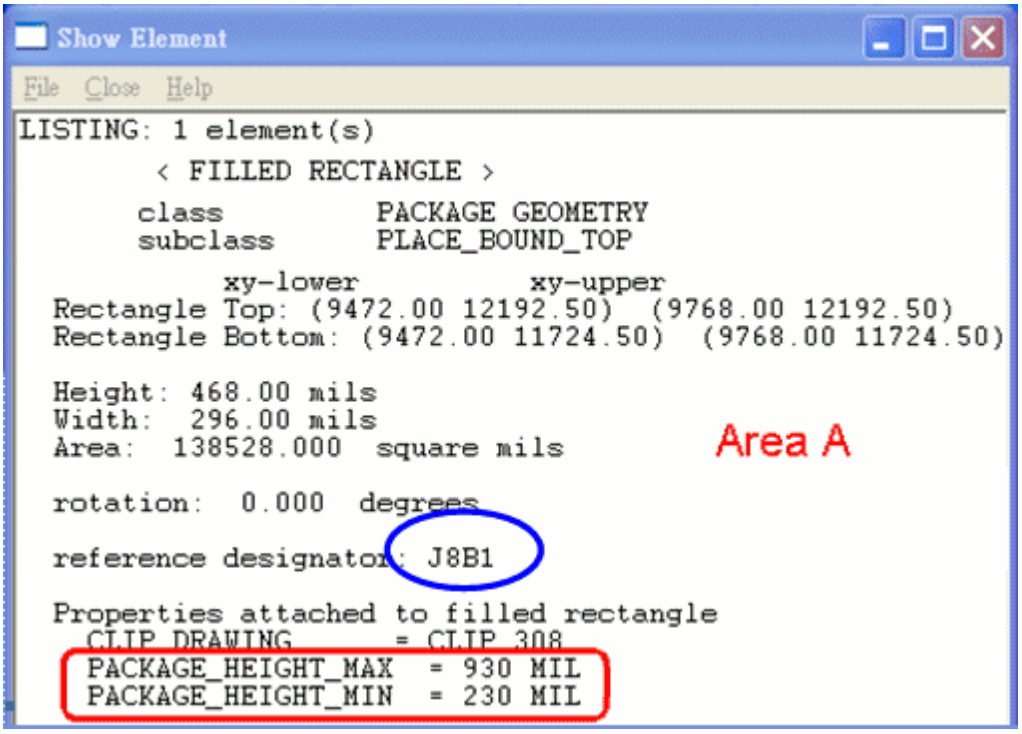
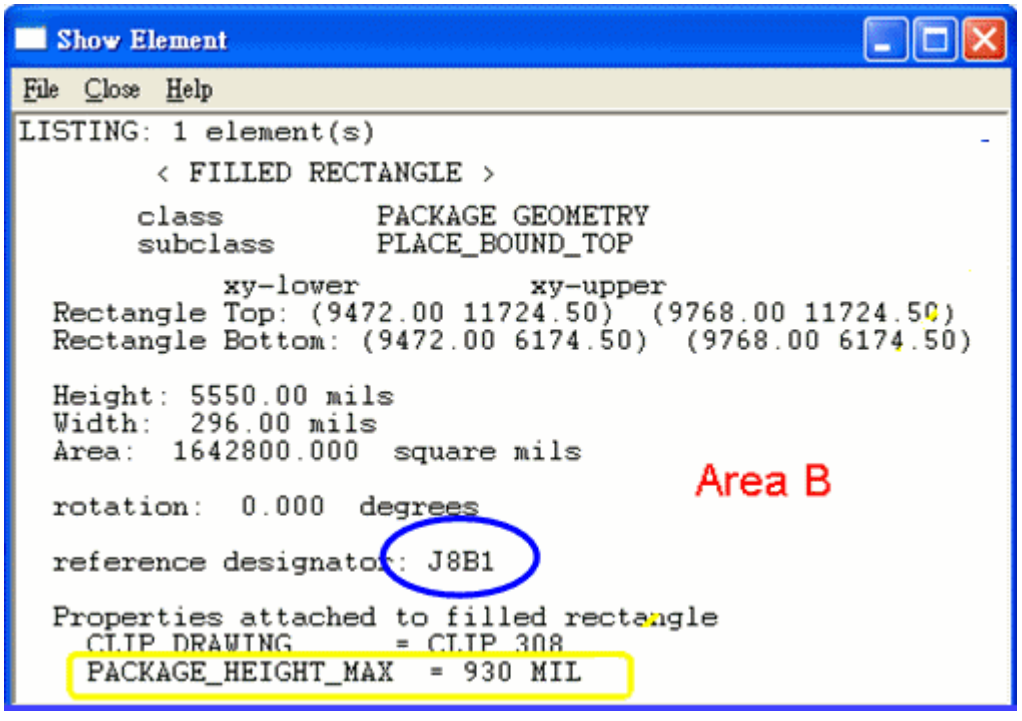
  A layer (height_top) is created in ODB++ to store height information for areas where there is space underneath components. In this layer, the maximum height of components that can be placed in a particular area is defined in the ODB++ feature attribute .drc_max_height (Maximum Height for Component). This attribute is set to the value of package_height_min for components, or for areas of components, where package_height_min is specified.

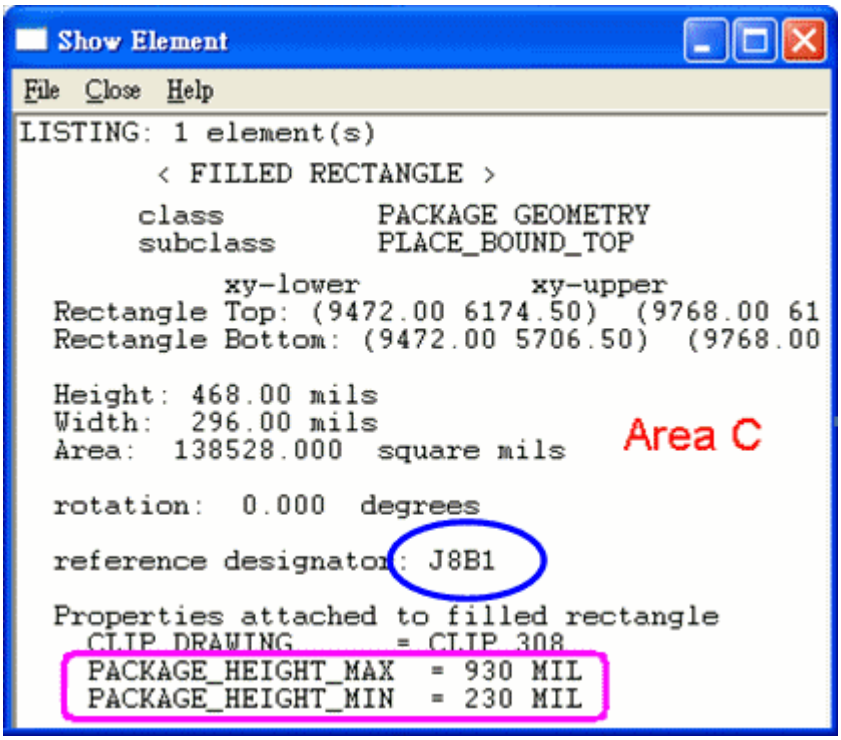
Example of a Component With Multiple Areas

In the example, RefDes J8B1 is a component with three areas defined.



These are the areas as defined in Cadence Allegro:

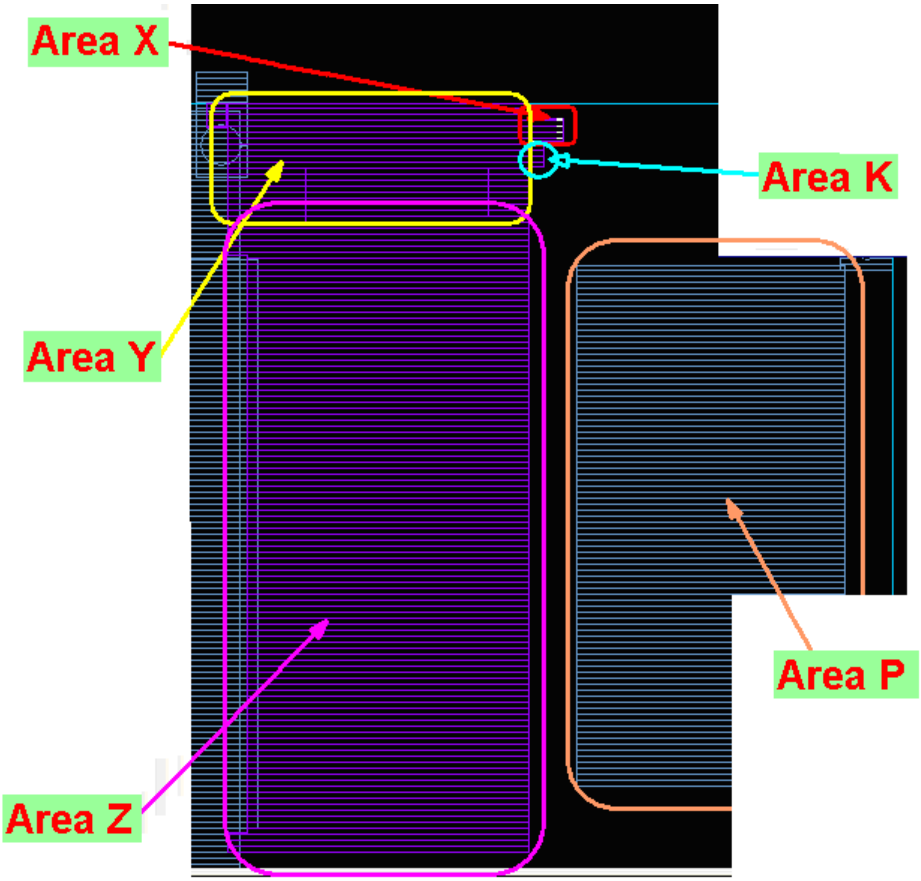
Area	Definition
Area A	
Area B	

Area	Definition
Area C	

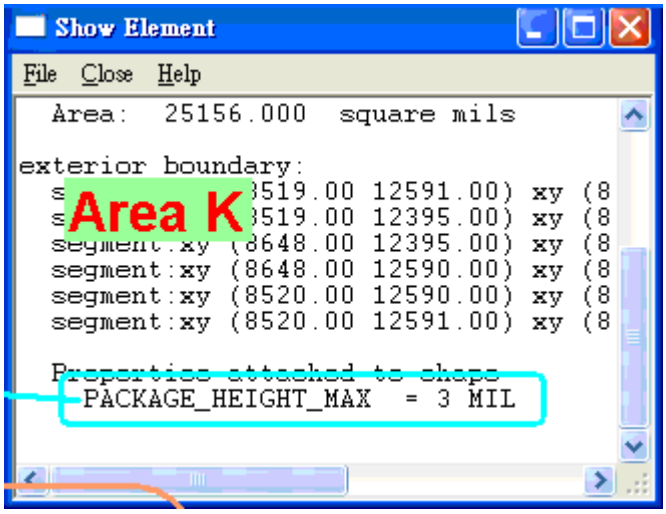
- The main part of the component (Area B in the example) is resting on the board, so it has no value for package_height_min.
- At the two ends of the component (Area A and Area C in the example), there is a space of height 230 mil underneath. A component that is placed under an end area of this component is not reported as an error if its height is less than 230 mil.

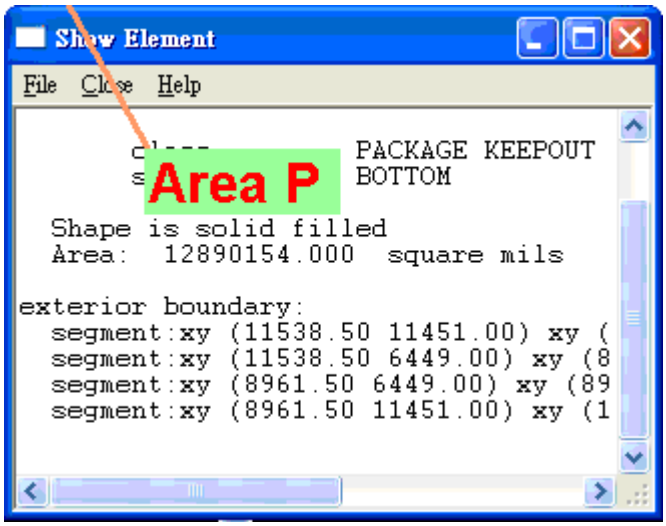
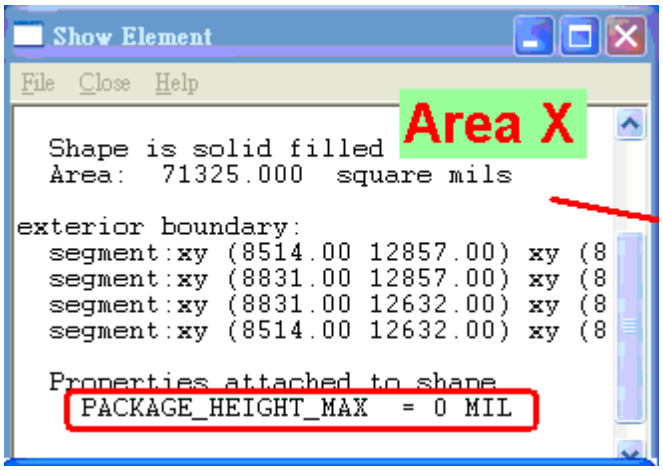
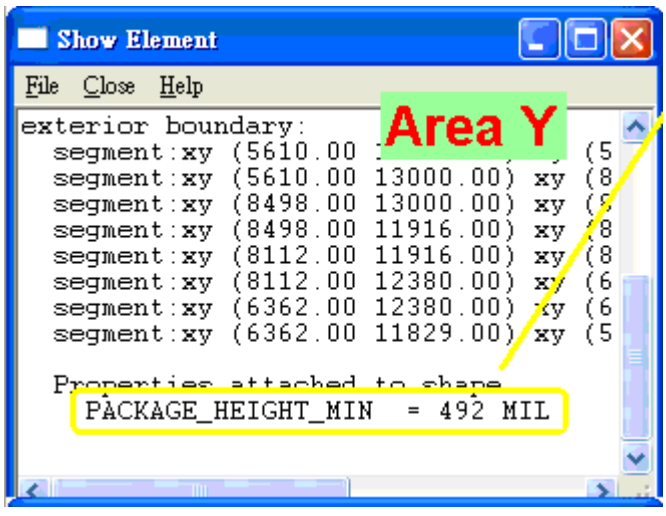
Example of Keepout Areas Based on Package Height Properties

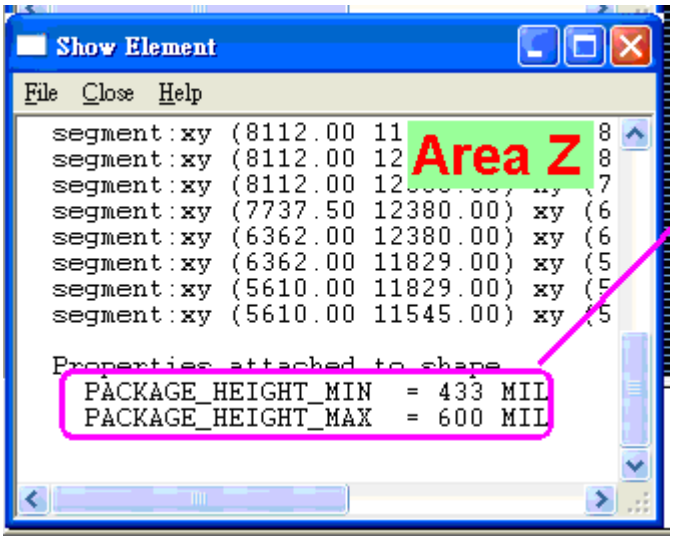
In the example, several areas are defined.



These are the areas as defined in Cadence Allegro:

Area	Definition
Area K	

Area	Definition
Area P	
Area X	
Area Y	

Area	Definition
Area Z	

The example shows these areas:

Area	package_height_min	package_height_max	Components Allowed	Description
X	not specified	0	all	package_height_max = 0
Y	492 mil	not specified	height < 492 mil	
Z	433 mil	600 mil	height < 433 mil	package_height_max is not considered when package_height_min is specified.
K	not specified	3 mil	none	package_height_max > 0
P	not specified	not specified	none	Neither property is specified.

Support for CLASS_CONSTRAINT_REGION

ODB++ Inside for Cadence Allegro supports class type CLASS_CONSTRAINT_REGION that was added to Cadence Allegro version 16.

Support for Translating Back-Drill Information

If the Cadence Allegro design contains back-drill information, new drill layers are created, for each drill span, to include this information.

Recent versions of Cadence Allegro implement back-drilling via the net property BACKDRILL_MAX_PTH_STUB, with the value denoting the maximum depth of the back-drill.

During translation, backdrills are added for pins/via holes and to existing drills. The span cannot be from top to bottom but must start or end with the top/bottom. A new layer is added for each backdrill span.

Support for Mirrored Padstacks

If a via is mirrored, or if a pin is used for a component on the bottom of the board, padstack information is taken from the mirrored layer.

Support for the COMPONENT KEEPOUT Class

The COMPONENT KEEPOUT class works like the PACKAGE KEEPOUT class.