cadence®

# Completing the Design

**Product Version 23.1**
**September 2023**

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

# Contents

1

# Design Completion Task Overview

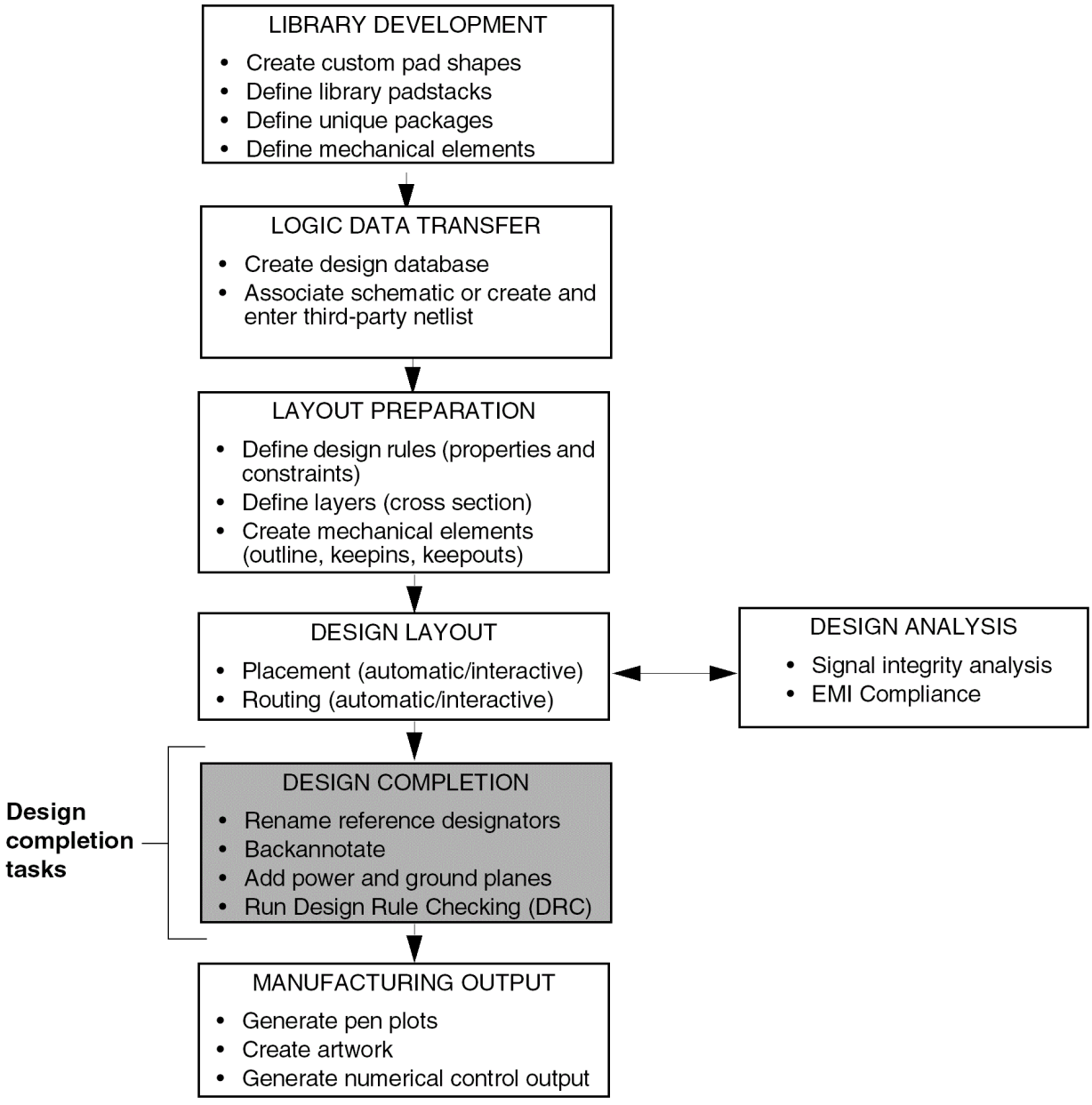
The design completion stage of a design flow may involve tasks related to renaming reference designators, running audits, extracting information from your design, and generating coupons. The following chapters cover these aspects of design completion.

Backannotating your design to a schematic capture tool is frequently done at this stage, but that information is covered in the Logic section of the documentation.

⚠ Many features are common to the layout editors, Allegro X PCB Editor and Allegro X Advanced Package Designer. When a feature is not common to all editors, it is noted in the heading. If an illustration shows only one of the editors, it is also noted.

The following figure illustrates the design completion stage in a design flow.

## Renaming Reference Designators in a PCB Editor Design Flow



2

# APD: Verifying the Design

Before you send a design out for fabrication, or pass the design footprint and electrical model along to the design, verify that your design meets all the physical and electrical design rules and requirements. APD  lets you check the physical design rules and electrical connectivity and provide an interface to electrical modeling and analysis tools.

Also the Component Compare feature lets you compare third-party die pin or component ball data (LEF/DEF, DIE text, OpenAccess (OA) or a BGA text file) to the corresponding footprint in the APD database (.mcm).

With Allegro Advanced Package Designer (APD), you can also produce documentation drawings and add various types of manufacturing output information to the design database that helps fabricate, manufacture, and assemble the die component.

The tasks include the following:

- Generating a Netlist with the Properties Output File
- Generating DIE/BGA (ASCII) Output
- Checking Physical Design Rules
- Checking Electrical Analysis
- EMI Analysis
- Using Component Compare
- Generating Outputs

# Generating a Netlist with the Properties Output File

You can generate a netlist output file that contains pin and net properties for the current design. For information, see the `netout` command.

# Generating DIE/BGA (ASCII) Output

You can create text files representing your die and BGA using File-Export-DIE Text-out Wizard (`die text out` command) and File-Export BGA Text-out Wizard (`bga text out` command) respectively. The format is organized in user-defined columns of data that can be used by spreadsheet software, for customization, or generating a variety of reports. Refer to the `die text out` and `bga text out` commands in the *Allegro PCB and Package Physical Layout Command Reference.*

# Checking Physical Design Rules

Online Design Rule Checking (DRC) provides immediate feedback when you violate a design rule. For details, see Running Online DRC in Chapter 1, "About Design Rule Checking," in the *Allegro User Guide: Creating Design Rules*.

## Comparing a Technology File with a Design File

If you start your design with a technology file, use the `techfile compare` command to compare the parameters, constraints, and user-defined properties in your current design to the corresponding values in an existing technology file. The APD/SiP tool writes any differences into the `tf_compare.log` in your current directory. You can use this command to check that your layout conforms to the specified constraints, parameters, and user-defined properties. The compare option reads the specified technology file, compares its values with those of the design drawing, and writes the comparison to a `tf_compare.log` file.

The log file contains any warnings or errors encountered when reading the specified technology file along with any differences found between the constraints in the technology file and the design drawing. The tool checks only the constraints specifically contained in the technology file against their counterparts in the layout.

# Checking Electrical Analysis

Pre-route signal integrity analysis lets you identify and correct potential problems in routing critical signals. Three-dimensional wire bonds are difficult to model; therefore, RLGC models are often calculated outside the APD/SiP tool and then you can make an association between the wire bond in the APD/SiP tool and the external model's text file in the *Bondwires* tab in the Signal Model Assignment dialog box. You can access this dialog box by choosing *Analyze-SI/EMI Sim-Model Assignment* (signal model command).

You can quickly scan the entire design and compare signals to the predefined electrical constraints. Using this quick scanning method and iterative "what-if" scenarios, you can identify and concentrate on problem signals. You can also modify circuit topologies to achieve optimum or acceptable results.

Pre-route signal integrity analysis reveals the following:

- How placement and net scheduling affect critical delays and reflection
- Where to terminate nets

## Simulation Overview

APSI (simulator tool) analyzes signal integrity and electromagnetic interference. The simulator helps you resolve the high-speed interconnect problems that often accompany higher-density designs, shorter cycle times, higher clock frequencies, shorter rise and fall times, and decreasing ratios of rise time to propagation delay. With the simulator, you can examine a design for delay, distortion, parasitic and crosstalk effects, electromagnetic interference (EMI), and design rule violations, and review the results in both waveform and text report formats.

When you analyze a design, you simulate the behavior of one or more extended nets (or Xnets). An Xnet is a set of connected and coupled nets.

Before performing a simulation, you need to properly prepare the design for simulation including properties, constraints, stackup definition, and device model assignment. If you set up the design correctly, it should pass the checks triggered by the `signal design audit` (*Analyze – SI/EMI Sim – Audit – Design Audit*) command in a satisfactory manner. A design audit:

- Checks the design.
- Checks the setup of referenced simulation model libraries.
- Displays the *Setup Report,* which verifies correct design setup and that all libraries and models it references are available and correct.

The Setup Report contains three sections describing:

- Errors pointing to problems with your design. You must address these deficiencies to avoid serious simulation problems.
- Warnings pointing to areas where you can enhance the accuracy of your simulations.
- Information about the design.

To find serious errors in setup, the simulator checks for the following situations:

- Zero thickness layers in the layerstack
- Nets with POWER or GROUND pins, but no VOLTAGE property
- Nets with POWER or GROUND pins or VOLTAGE property, but no shape or VOLTAGE_SOURCE pin
- No VOLTAGE property on any net
- Nets with no receivers and no pins attached to a component with an ESpiceDevice SIGNAL_MODEL reference
- Clines with a SIGNAL_MODEL reference that do not exist in any open interconnect library
- No working interconnect library
- Active DesignLink reference does not exist in any open device library
- Default IOCells that do not exist in any open device library
- Components with a SIGNAL_MODEL reference that does not exist in any open device library
- Model versions
- Referenced device models that do not pass `dmlcheck` (Audit Report lists problem models, but actual errors appear in APSI log window)
- Pin signal_model parameters in IBISDevice pinmap do not match APD/SiP layout pinuse
- APD/SiP layout component pins not found in IBISDevice pinmap (other than NC pins)
- Components with the TERMINATOR_PACK property, but are not assigned an ESpiceDevice SIGNAL_MODEL property

To find setup problems that might hinder accuracy (Warnings), the simulator checks for the following situations:

- Default settings in the layerstack
- Wire bond layers that do not have the SIGNAL_MODEL property attached to clines

- Components that have no SIGNAL_MODEL property

The simulator reports the following design information:

- Layerstack information
- Number of nets and components
- Assigned models, including the library file

If your design is set up for multi-design simulation, the *Design Audit* command also checks your project across the designs.

The simulator generates analysis results by developing and simulating circuit models of a design. When it develops circuit models, the simulator uses the device models you assign to components. It automatically creates the interconnect models during simulation.

You can use the simulator throughout the development of a design:

- During critical component placement
- After component placement and before you route any connections
- After you route the critical nets
- After you route the entire design

## Simulation Use Model

This use model describes a design process that employs the simulator to check for signal integrity and EMI in high-speed designs. The simulation tasks outlined include:

- Setting up the simulator and performing signal integrity analysis before routing
- Performing signal integrity analysis during routing of critical nets
- Performing signal integrity analysis after routing

## *Pre-Route Work Flow*

Pre-route signal integrity analysis comes after preliminary placement and before routing. In pre-route signal integrity analysis, you are looking for the following:

- How placement effects critical delays and reflections in the design
- How net scheduling effects delays and reflections

- The need for terminators on nets in the design
- An early evaluation of the power distribution system

## *Starting Pre-Route Signal Integrity Analysis*

The following steps describe the procedures for setting up the simulator and starting pre-route signal integrity analysis.

1. Initialize an analysis directory to tell the simulator where to write signal analysis data files (optional).
   After placement, the simulator can provide you with delay and distortion data that comes from hypothetical traces. The simulator develops these hypothetical traces based on a percent manhattan distance between pins and user-defined assumptions for the characteristic impedance and propagation velocity. You can specify this information on the *Interconnect Models* tab in the Analysis Preferences dialog box.
2. Load the device model libraries.
3. Assign the device models from these libraries to components in the design.
4. Set the simulation preferences and set up the layout cross-section.
   The preferences specify, for example, default IOCell models and the units of measurements for reports. When you set up the cross-section you define how the layers stack up and which materials and thicknesses you use for these layers.

## *Performing Pre-Route Signal Integrity Analysis*

After you set up your device models and device model libraries and make IOCell model assignments, you can perform simulations and generate analysis data. The following steps describe the procedures used to generate and interpret analysis data.

1. Select signals for simulation by choosing one of the following:
   - Click to select a ratsnest line or a pin in the Design Window.
   - Specify a net by name in the Signal Analysis dialog box.
   - Specify a netlist file by name in the Signal Analysis dialog box.
2. Select the type of analysis results to create by choosing one of the following:
   - Click *Reports* in the Signal Analysis dialog box to present the analysis results as text

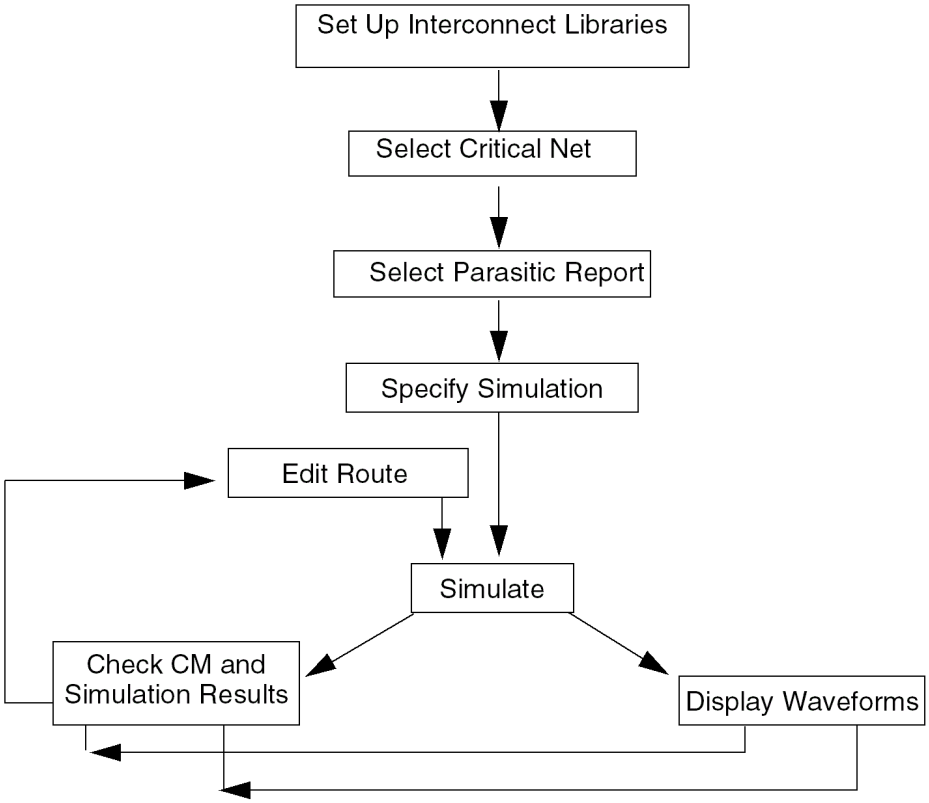
reports. This opens the *Report Generator*.

- Click *Waveforms* in the Signal Analysis dialog box to present the analysis results as waveform files. This opens the Waveform Simulation dialog box.

3. Specify the type of simulation that you want the simulator to run.
   Select the appropriate options in the Report Generator or the Waveform Simulation dialog box.
   To use the Power Plane Designer to analyze you power and ground plane design, specify the *Do Plane Modeling* interconnect modeling preference and perform an SSN simulation.
4. Trigger the simulation by clicking *Create Report* or *Create Waveforms.*
   The simulator performs the necessary simulations based on your specifications.
5. Following simulation, review the delay and distortion data in text reports, view time domain waveform displays at receiver pins, or view animated movies of ground bounce between the power and ground planes.
   Based on the simulation results, you may edit the placement of components again, modify net schedules, or experiment with terminators to suppress distortion.

## *Critical Net Analysis Work Flow*

After pre-route analysis, you may want to interactively route critical nets and then analyze them for signal integrity. The following flow chart shows the procedures in Critical Net analysis during routing.




During pre-route analysis, the simulator built a simulation circuit model. It used the device models you specified and the hypothetical interconnect models that it approximated from the percent manhattan distance, the default impedance, and the default propagation velocity you specified. Now that the critical nets have been routed, you can analyze them more precisely, this time using the actual conductor instead of the manhattan-based estimates.

You can begin critical net analysis with interconnect library setup to specify where you want the simulator to save the interconnect models it creates. You might also create a *Parasitics* report for a critical net.

You can also scan the design for problem areas using the same steps you followed in pre-route analysis. You then select a net for simulation and look at the results as waveform displays and text reports. After you examine your results, you can edit the routing for that critical net and perform another analysis. The process of analysis and editing the traces is an iterative process that you can continue until you see satisfactory simulation results.

## *Post-Route Work Flow*

Once you complete the routing phase, you can re-scan the design to analyze routing effects on the design. If the layout is modified to meet pre-route signal noise analysis constraints, you can verify the design by substituting actual route characteristics for manhattan-based estimates.

Post-route differs from pre-route as follows:

- You begin with parasitic analysis.
- Most layout modifications involve editing conductors.
- You use the Conductor Cross Section window to view geometric displays of the models that APSI writes for segments of interconnects.

## During post-route signal integrity analysis you look for:

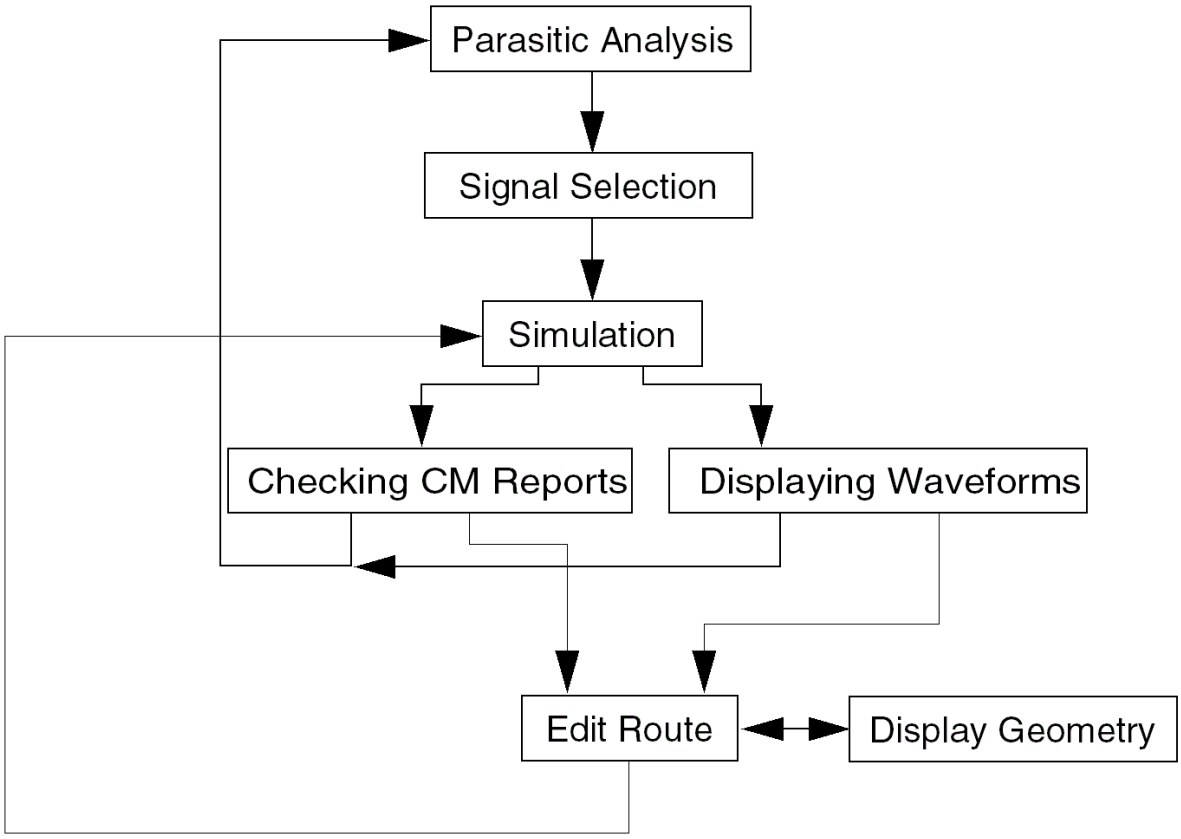
- The effect of the routed interconnect on signal integrity and EMI
- The effect that the routed interconnect have on each other
  You can now see the effect of couplings between interconnect segments and how they create crosstalk and affect signal integrity and EMI.
- The effect of neighboring interconnects that you have added since critical nets were routed

## Design Flow Tasks

APD/SiP tools perform the following post-route checks:

- Signal integrity on routed conductors
- Crosstalk introduced by neighboring nets

The following flow chart describes post-route signal analysis:




To perform post-route signal integrity analysis:

1. Begin with the parasitic analysis.
2. After parasitic analysis, you can scan the design for problem areas or proceed to detailed analysis of individual nets.
   You can run single or multi-line simulations depending on whether you want to take neighboring nets into account.
3. After simulation:
   - Review the *Delay, Ringing, Crosstalk, SSN*, and *EMI Single Net* reports.
   - Use the Conductor Cross Section window (`sigxsect`) to look at geometric displays of the models the simulator writes for interconnect segments.
   - Use the Emwave window to view ground bounce movies.

If two interconnect segments are within the distance specified in the *Geometry Window* parameter and if you are running multi-line simulations, the simulator writes a model that includes both interconnect segments. You see both segments in the Conductor Cross Section window. You can

also display equipotential field lines between interconnects in the Conductor Cross Section window. You can slide interconnect segments and see how it changes both the field lines and the RLGC matrix of the model.

Because of the high volume of simulations often performed for post-route analysis, you can run post-route analysis in batch mode.

# EMI Analysis

APSI provides EMI single net simulations, which enable you to compute differential mode radiated electric field emissions from traces. Simulation results include a graphical display of the emission spectrum and a text report summarizing emission details and compliance results.

## Using EMControl with APSI

Where EMControl is available, you can use APSI with EMControl to perform EMI analysis. Some of the signal routing and signal quality rules provided with EMControl employ APSI simulations and APSI device models during analysis for EMI. Using EMControl enables you to begin evaluating designs for EMI early in the design process and with increasing accuracy throughout design development.

Before running EMC rule-checking, perform the following APSI setup tasks:

- Specify any analysis preferences. If necessary, APSI creates a new simulation case directory.
- Specify which device and interconnect model libraries APSI should use.
- Assign the SIGNAL_MODEL property to components.

Also, when you initialize the EMControl run directory, you need to point EMControl to this APSI run directory.

See the EMControl online help for more information.

# Using Component Compare

During IC and Package co-design, changes are made by both IC and package designers. As part of an ECO-validation process, this feature can be used to view the changes in the update file before importing them into the current design. Alternately, at the end of the co-design process, it is imperative that both IC and package designers are viewing identical pin data. The package designer can run this tool to verify that the die pin pattern in the IC and package designs match.

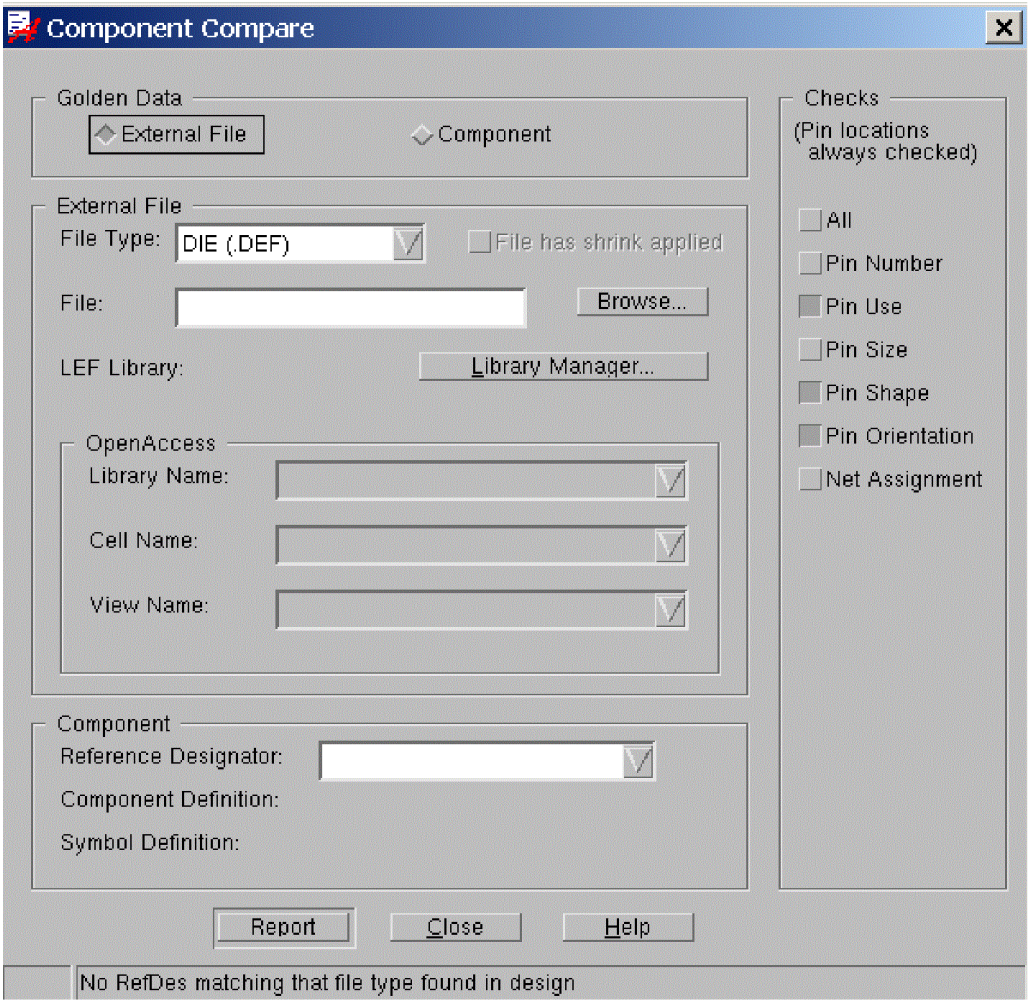
Using the Component Compare feature, you can compare third-party die pin or package ball data (LEF/DEF, DIE text file, die abstract, BGA text file, or OpenAccess (OA) data) to the corresponding footprint in the System-in-Package (SiP) or the `.mcm` database. Once you specify the golden data, which is the source against which differences are highlighted, the layout tool creates a report of the logical and physical differences such as pin location, number, use, size, shape, orientation, and net assignment.

You can use this feature:

- During Distributed Co-Design
  You can compare die pin patterns from a die abstract file and a component in a package database before updating a co-design die.
- During IC and package co-design
  Either the package designer or the IC designer makes changes to the die and the changes must be sent to the other designer for approval through an ECO-type process. The person receiving the modified die pin data wants to review the changes in the file before importing them into the design.
- At the end of the package and IC co-design
  Package designers want to verify that the die pin pattern in the IC design and package design match. This is important to ensure that both IC and packaging tools have identical die pin data.
- When package design is complete
  Package designers want to compare the ball patterns from a third-party tool (for example, a BGA text file) and the ball pattern in a package.

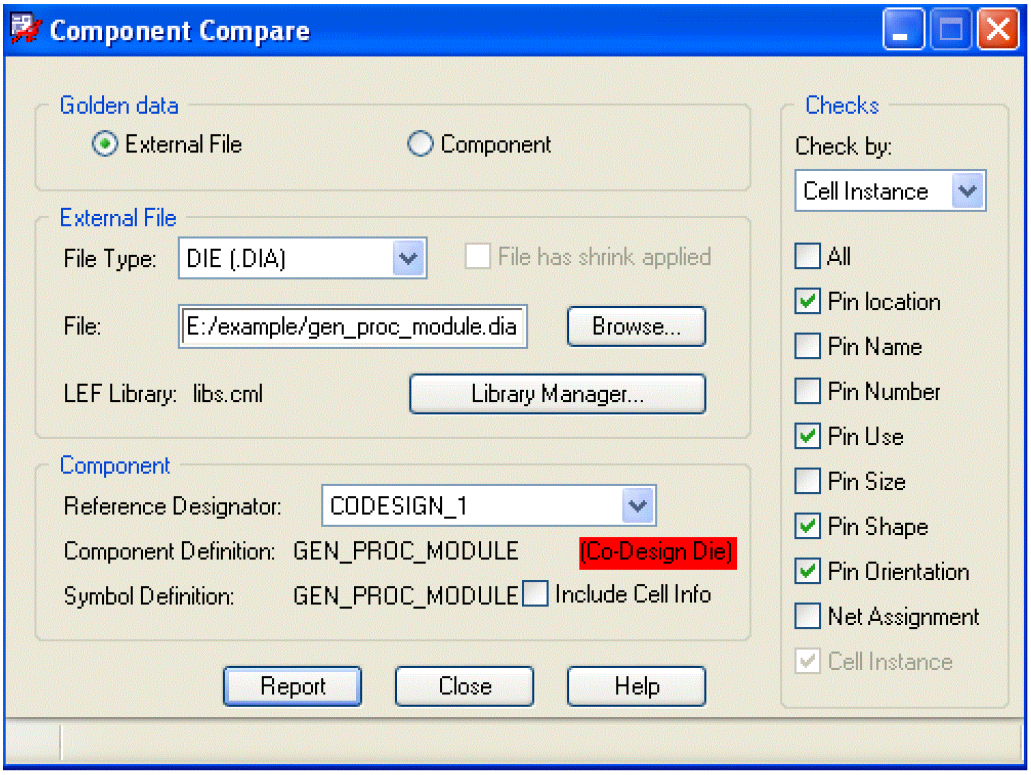
The following figure shows the Component Compare dialog box that appears on the UNIX platforms (OA is not supported on Windows).

## Component Compare Dialog Box (UNIX Platform)




The following figure shows the Component Compare dialog box that appears on Windows.

### Component Compare Dialog Box (Windows)




For information on using this feature, see the `compare comp` command in the *Allegro PCB and Package Physical Layout Command Reference*.

## Component Compare Report

The Component Compare report highlights the differences in pin pattern. The ASCII report contains information regarding data checks made on the pin location and the additional selections you make. Also, pins, which are new to or removed from the golden data, are listed.

This report is saved in the current working directory as `<refdes>_compare.rpt`. Revised copies of previous reports are also saved.

Each report has a title that indicates the date and time that the report was generated. The body of the report has these sections:

- Header
- Pin Differences
- Missing Pins
- Extra Pins

## Header

The header consists of the following:

- The list of checks you selected for comparison.
- The design names from both the external file and the `.sip/.mcm` database component as well as information specifying the golden data. The report also indicates if the design names do not match.
- A comparison of die extents in the external file and the component. The report indicates whether the die extents match. If they do not match, the layout tool lists both extents.

## Pin Differences

This section outlines the differences in pin attributes between the external file and the component, based on the checks selected. Included in this section are pins that match in location in both the external file and the component, but which have differences in either pin numbers, pin use, pad size, pad shape, orientation, or net assignment.

## Missing Pins

This section includes pins in the golden data for which there are no corresponding pins with the same locations in the other data.

## Extra Pins

This section includes all pins that are not listed in the golden data. For example, if the golden data is the DEF file and there is a pin in the component whose location does not match any pin in the golden data, it is included in this section.

## Report

The Component Compare feature generates an ASCII report highlighting the differences in pin pattern. This report is saved in the current working directory as *<refdes>*_compare.rpt. Revisioned copies of previous reports are saved. The report contains data checks made on the pin location as well as the selections in the Checks section of the dialog box.

In addition, pins which are new to and removed from the golden data are listed.

# Example

Below, you can view part of a Component Compare report.

```
Component Compare Report created on: Thu May 07 15:12:58 2009
-------------------------------------------------------------------------

Checks: [location][name][number][use][size][shape][orientation][net assignment][cell instance]

Design Component name: DIE
Design filename (golden data): dee1.sip
Input file component name: DIE
Input filename: R:/aastorage/16_3/distributed_codesign/NEW_CODESIGN/save_named_16-3-62.dia
Die sizes of component from design and input file match.
Check By: Cell Instance
The following differences were found:
-------------------------------------
Pin: Bump_676_25_25
 Pin location (Golden data - Component):(336415.000,337123.000)
 Pin location (Input file):(-6316115.000,337123.000)
Pin: Bump_675_24_25
 Pin location (Golden data - Component):(309415.000,337123.000)
 Pin location (Input file):(-6343115.000,337123.000)
Pin: Bump_674_23_25
 Pin location (Golden data - Component):(282415.000,337123.000)
```

Pin location (Input file):(-6370115.000,337123.000)

```
Pin: Bump_673_22_25
 Pin location (Golden data - Component):(255415.000,337123.000)
 Pin location (Input file):(-6397115.000,337123.000)
Pin: Bump_672_21_25
 Pin location (Golden data - Component):(228415.000,337123.000)
 Pin location (Input file):(-6424115.000,337123.000)
Pin: Bump_671_20_25
 Pin location (Golden data - Component):(201415.000,337123.000)
 Pin location (Input file):(-6451115.000,337123.000)
Pin: Bump_670_19_25
 Pin location (Golden data - Component):(174415.000,337123.000)
 Pin location (Input file):(-6478115.000,337123.000)
o
o
```

```
o
End - Component Compare Report
-------------------------------------------------------------------------
```

# Documenting the Design

You can produce documentation drawings and add various types of manufacturing output information to the design database that helps fabricate, manufacture, and assemble the die component. Design information includes dimensions and important features of the substrate, bond wire diagrams, plots, and reports for post-processing.

The APD/SiP tool lets you plot each layer in the design, including conductor, plane, die-level, and dimensioning layers. Using the tool's drafting capabilities, you can create drawing formats, add dimensioning, and create exploded views of specified areas of the design, including notes. If you need to create three-dimensional documentation or graphics, the APD/SiP tool can transfer data back to an MCAD system such as AutoCAD™, using the DXF interface.

You can tailor manufacturing output such as device type, scale factor, output units, and so on. You can also automatically generate an aperture table using data directly from your design database.

Gerber format can be either vector- or raster-based. Manufacturing output also supports the stream format.

The plot command sends the current design to the printer. You can control the scale and number of dots per inch for the output. On UNIX you can create penplots.

For detailed information on creating plot and intermediate plot files, see "Plotting," in the *Allegro User* Guide: Preparing Manufacturing Data.

# Generating Outputs

You can generate board-level component data, exporting the data to files for input into Allegro X PCB Editor. Also, you can generate the files to fabricate and manufacture your new package design. Allegro Advanced Package Designer (APD) tool includes several manufacturing outputs and interfaces to accomplish these tasks.

To obtain via coordinates for the manufacturing process, use the NC Drill feature. Normally, this file is created as input to a numerical control machine for drilling holes in a substrate. However, this feature may be used to generate the X,Y coordinates for vias. The assumption for NC Drill is that every padstack (pins and vias) with a Drill Hole definition is included in the output. Those padstacks without a Drill Hole definition (`smd`) are not included in the output. To generate numerical control data, use the general process that follows and refer to Creating Numerical Control Data for additional details and procedures.

- Create different drill drawings
- Generate a drill legend table
- Generate the paper tape output file
- Create an APD profile

## Outputting DXF Data

Mechanical data consists of component size and shape, cavity location, and I/O pin size and location. You can export mechanical data directly to a mechanical design package such as AutoCAD™. The APD/SiP tool recognizes the AutoCAD release 12-14 bi-directional mechanical interface—**D**rawing e**X**change **F**ormat (DXF). The AutoCAD™ DXF bi-directional mechanical Interface lets you exchange APD/SiP design data with other computer aided design (CAD) systems. Refer to DXF Bi-Directional Interface in the *Allegro User Guide: Transferring Logic Design Data* for details and procedures.

## Outputting Stream Data

You can convert an APD design to GDSII stream format. The stream full-geometry view extracts all geometric information from the `.mcm`/`.sip` database and converts only those classsubclasses included in the Layer Filter table. Arcs and circles are converted to line segments before conversion to stream because stream does not allow arcs and circles.

You can use the stream out command. Refer to Stream-Based Artwork in the *Allegro User Guide: Preparing for Manufacturing* for details and procedures.

3

# Running the Assembly Rules Checker

The Assembly Rules Checker feature lets you perform specific design rule checking of several different rules in a package design. These rules allow you to gauge whether the package, as designed, meets the physical and spacing requirements necessary for the part to be successfully manufactured and assembled. These design rule checks are checks on the size (length) of an object, the position or location of an object, or some measure of the distance between two objects. The Assembly Design Rule Check (DRC) is different from the online DRC that you use in package design tools, which helps you find design flaws that do not meet the original system specifications or process technology limitations.

The Assembly DRC errors may range from something as simple as the percentage of a bond wire's length that is over the extents of the die, or they may be as complex as assessing the interaction between the bond wires and adjacent dies, spacers, and interposers above and below the wires in the same die stack.

⚠ The Assembly Rules Checker feature is available in the SiP Layout tool and APD (using SiP Layout XL).

# How the Assembly Rules Checker Works

The Assembly Rules Checker process is a batch process that you initiate after you complete the constraint settings. To run assembly rules checker on your designs, you need to complete the following tasks.

- Capture constraints in Constraint Manager
  Constraints for assembly rules are captured using the workbooks and worksheets in the Assembly domain in Constraint Manager. To launch Constraint Manager, choose *Setup – Constraints – Assembly*.
- Specify the rules to be verified
  Run the Assembly Rules Checker by choosing *Manufacture – Assembly Rules Checker* (`assemrules standard` command). In the Assembly Design Rule Checks dialog box, manually select the rules to be verified. If required, you can edit the constraint values for the selected tool, by launching Constraint Manager from the Assembly Design Rule Checks dialog box.

  ⚠ For cross-probing between Constrain Manager and the layout editor to work, the Assembly Design Rule Checks dialog box should not be open.

Once you set up the rules, they are processed and checked in a batch mode.

The Assembly Rules Checker batch process may take considerable time to complete, depending on the number of active rules and the number of variants that are to be checked for each rule. A larger design takes longer to check, and the check takes longer if you run more rules. You should take this into account before invoking the Assembly Rules Checker process.

By default, the results of the rule checks are saved in a tab-delimited report file (`adrc_report.txt`), and DRC markers are placed in the design where the violations occur. Each DRC marker lists the item in violation, along with the required and actual values for the constraint. A log file (`adrc.log`) is also generated that lists which rules were checked, along with the values and settings for each rule.

Typically, you should run the Assembly Rules Checker process before final manufacturing sign-off and before you generate artwork and masks. However, you should run the checks as soon as possible during the design cycle, so that you find potential problems while they are easier to correct.

You should run Assembly Rules Checker:

- After you place all die components, but before you bond them.
- After you create the bond finger pattern and all wire bonds, but before you perform package

routing (If desired, you can perform a true 3D DRC check at this time, as well).

- After you complete final package routing and you are running the design through the final analysis and manufacturing prep stages.

# Capturing Assembly Design Rule Constraints

You set the constraints for the rules either at design level or by completing the Assembly Constraint Sets (ACSets) in the Constraint Manager. Usually, constraint values for rules that do not have any parameters dependencies, is specified at the design level. However, if the constraint values for a assembly rule is influenced by physical characteristics, or properties, of the items, use of ACSets is recommended.

## Assembly Constraint Set (ACSet)

An ACSet is a named, reusable collection of constraint values for assembly rules. In a design created in the 16.3 release, there are no predefined default constraints or ACSets. ACSets can be added for the rules related to wires and dies, and also for design-level and layer-based constraints.

Using ACSets is recommended in situations where the constraint values are influenced by different physical parameters. For example, in case of wirebond designs, there are assembly rule constraints based on wire profiles (a similar path in space followed by a group of similar wires) as well as on the physical properties, such as wire diameter and wire material. In these situations you can define different ACSets for different wire profiles.

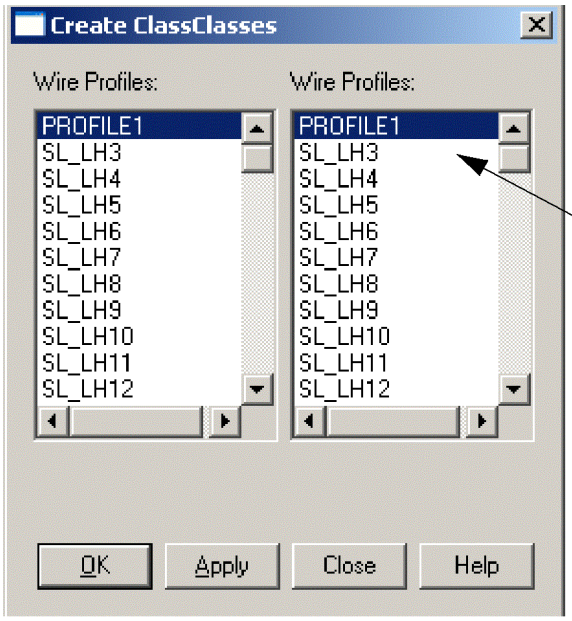
To know more about the difference between constraints and the dependency parameters, see Constraints Vs Properties.

## Class-Class Objects

For capturing Assembly Rule Constraints, you can use Die class-class and Wire Profile class-class objects.

The Die (Wire Profile) Class-Class objects are used to capture constraints between dies (or wires) belonging to the same class (profile) or between dies (or wires) belonging to the different classes (profiles).

A Class-Class object is created when you group two wire profile or two Die classes together.



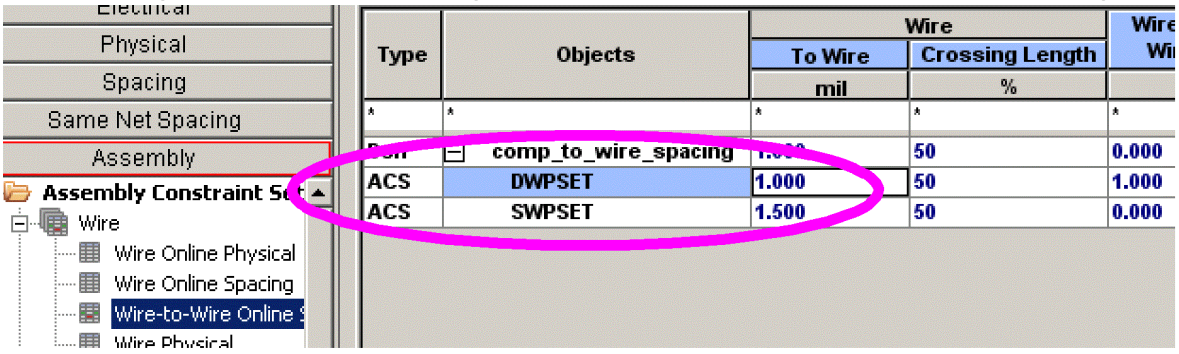

Creating a Class-Class object to capture constraints between wires belonging to same profile

## *Example 1*

Consider a design in which the value specified for the wire-to-wire spacing constraint is dependent of the wire profile. The constraint value is different if the two wires belong to same profile than the value if wires are from different profiles.

In this scenario, you use ACSets and Wire Profile Class-Class objects to capture constraint values. The sequence of tasks to be performed in this scenario are:

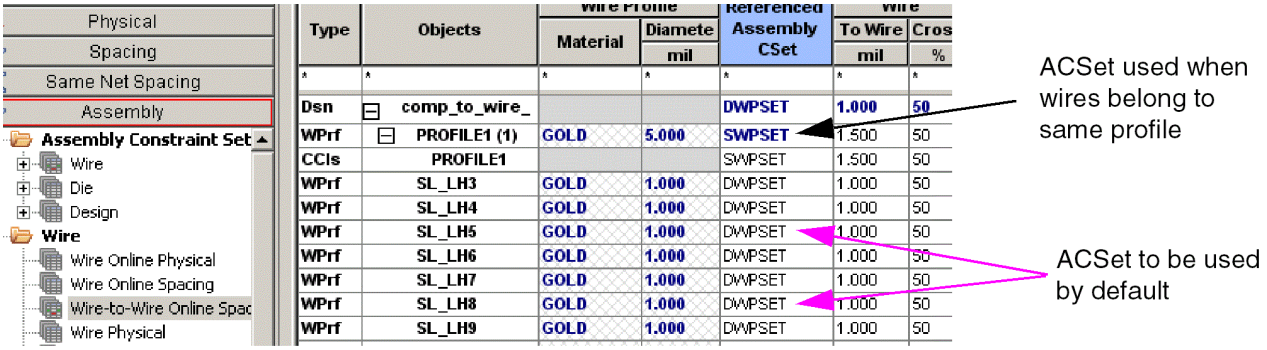
a. In Constraint Manager, create two ACSets, one to capture constraints between wires with same profile and other to capture constraints between wires of different profiles.




b. Apply these constraints in the relevant worksheets in Wire workgroup.
At the design level, specify the ACSet that is to be used by default, as the Reference ACSet.

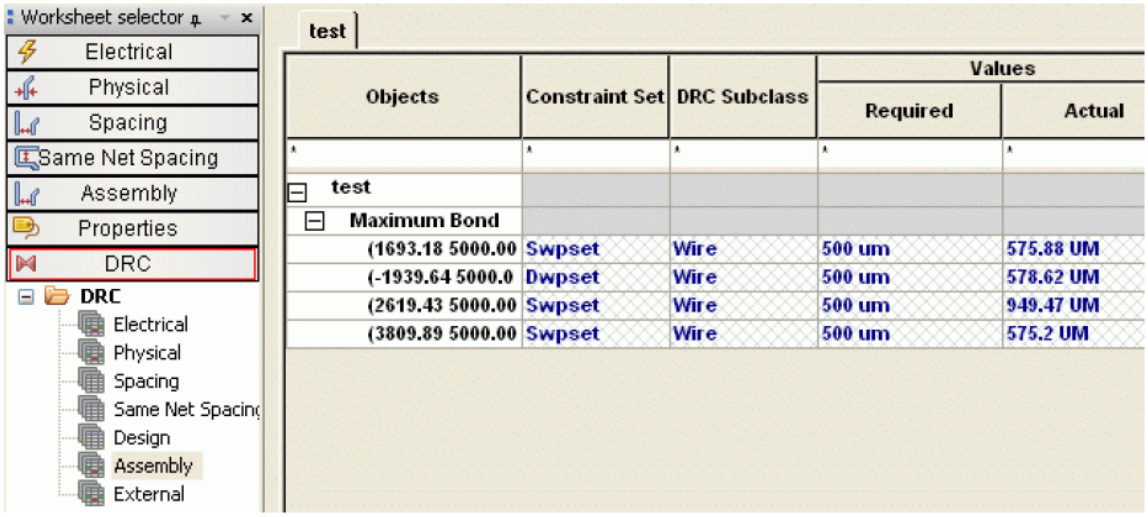
c. Create a class-class object with same wire profile as member.

d. In the wire-to-wire Online Spacing Worksheet, specify the ACSet to be used for the Class-class object




## DRC Markers

When assembly rule checks are run as a batch process, results are displayed as DRC markers. These markers are placed in the design, in the output reports file, and in Constraint Manager — in the Assembly worksheet of the DRC workbook.




By default, the output reports file is named as `adrc_report.txt`. If required, you can specify a different name in the Assembly Design Rule Checks dialog box. A section of the report file is shown in the following figure.

```
Rule:   Wire to Component Spacing
Wire Profile:   PROFILE1
Constraint:     Required Component to Wire Minimum Spacing:     75 MIL
--------------------------------------------------------------------------------

Die     Wire Start Location     Wire End Location       Net Name        Wire Profile    Component       Actual Dis
ConstraintValue
===     ===================     =================       ========        ============    =========       ==========
==================
DIE     ( 312.042 501.694 )     ( 189.803 419.939 )     POWER   PROFILE1        C1      30.034 MIL      0
DIE     ( 312.042 501.694 )     ( 189.803 419.939 )     POWER   PROFILE1        C2      38.013 MIL      0
DIE     ( 498.306 532.738 )     ( 651.225 491.281 )     GROUND  PROFILE1        R1      44.78 MIL       0
DIE     ( 312.042 698.306 )     ( 213.58 800 )  DUMMY NET       PROFILE1        C3      62.11 MIL       0
DIE     ( 498.306 636.218 )     ( 638.568 678.841 )     DUMMY NET       PROFILE1        C4      51.029 MIL      0
DIE     ( 498.306 584.478 )     ( 643.171 587.938 )     DUMMY NET       PROFILE1        C4      47.006 MIL      0
```

## Constraints Versus Properties

You may want to apply a certain check to similar items, but using different constraints depending on physical characteristics, or properties, of the items. For example, there is a rule that checks the length of a wire. When using this rule, it finds all wires in the design that are less than some minimum value ‘X’ units in length, or greater than some maximum value ‘Y’ units in length. It may also be true that the minimum ‘X’ and maximum ‘Y’ values are different for wires of different diameter or material. (A thicker wire may be allowed to be longer than a thinner wire, for instance.) Similarly, the ‘X’ and ‘Y’ pairs might be different if the wire is made of gold than they would be if the wire is made of some other metal. For different wire materials or diameters (or ranges of diameters), a wire’s length would be checked against different ‘X’ and ‘Y’ pairs.

You may consider ‘X’, ‘Y’, ‘Diameter’ and ‘Material’ as parameters (things that should be used as input) of the rule. However, for Assembly Rules Checker, ‘X’ and ‘Y’ are treated as the constraints because you want to check that the wire meets those ‘X’ and ‘Y’ conditions. If they do not meet the conditions, they are considered a violation of the rule. ‘Diameter’ and ‘Materials’ are treated as technology properties because they influence what ‘X’ and ‘Y’ values will be applied in the check of a given wire. A wire that does not have that ‘Diameter’ or that is not made of that ‘Material’ is not flagged as a violation of the rule.

# Reusing Constraints

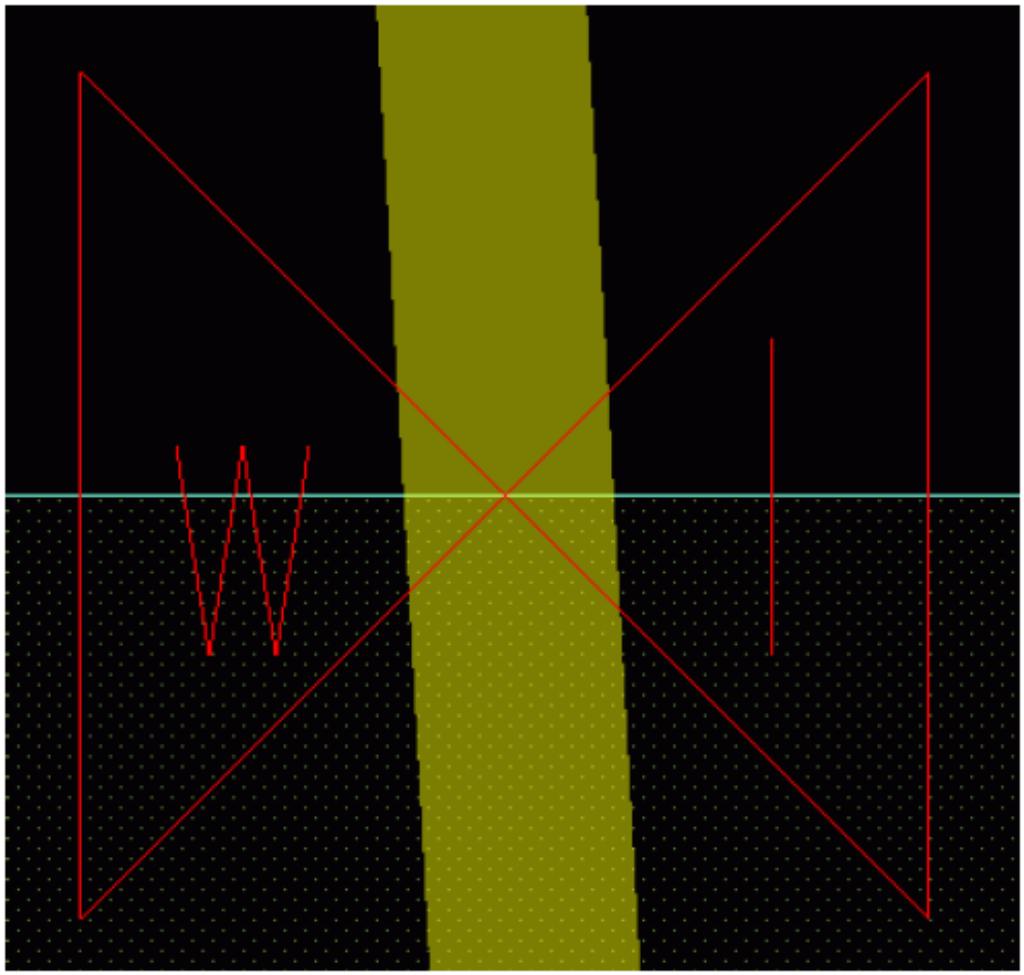
ADR constraints are saved in the Allegro technology file. Reusing the technology file with a new design, makes the constraints available for use in the new design. To reuse the assembly rules constraints, one of the following methods can be used.

- Reusing Technology File
  Assembly rule constraints are saved in the technology file. To export technology file with constraints data, choose *File – Export – Technology.* In the Export a technology file (`.tcf`x) dialog box, ensure that the Assembly Constraints check box is selected.
  To update a new design with technology file data, launch Constraint Manager and choose *File – Import – Technology File.* In the Import a technology file (`.tcf`x) dialog box, ensure that the Assembly Constraints check box is selected.
- Reusing Constraints Data
  To export assembly rules constraints data, choose *File – Export – Constraints*, and ensure that the Assembly Constraints check box is selected in the Export Constraints dialog box.
  To import these constraints in a new design, open the database file in layout editor and launch Constraint Manager. To import constraint data, choose *File – Import – Constraints.*

# Working with Pre16.5 Designs

Prior to 16.5 ADRC markers were listed in the External worksheet. However, the ADRC markers are listed in the Assembly worksheet of the DRC domain in Constraint manager. Also, ADRC marker will be deleted when object with marker is moved.

The two letter combination for the ADRC bow-tie marker will appear different for various rule changes. For example, for violation of Wire Length over Parent Die the letter combination is *wl* as follows.



The following table lists some of the combinations.

| Group | Rule | Combination |
|---|---|---|
| Wire Physical | Wire Length over Parent Die | wl |
| Wire Physical | Wire Length over Lower Die | wl |

| | | |
|---|---|---|
| Wire Physical | Wire Maximum Angle to Finger | wa |
| Wire Spacing | Wire Substrate End Distance inside soldermask | ws |
| Wire Spacing | Wire to Component Spacing | wc |
| Die Physical | Die Overhang | oh |
| Die Physical | Die Pad Pitch | pp |
| Die Physical | Die Pad To Lower Die Overhang | oh |
| Die Physical | Die Pad to Upper Die Spacing | pd |
| Die Spacing | Die to Connected Finger Spacing | df |
| Die Spacing | Die To Finger Spacing | df |
| Die Spacing | Die to Package Edge Spacing | de |
| Die Spacing | Die To Die Spacing, Connected Dies | dd |
| Die Spacing | Die To Die Spacing, Unconnected Dies | dd |
| Optical | Wire to Die Pad Optical Short | os |
| Optical | Wire to Finger Optical Short | os |
| Optical | Wire To Wire Optical Short, Die to Die | os |
| Optical | Wire to Wire Optical Short, Die to Substrate | os |
| Die Stack | Center to Center Delta, Extends Based | cc |
| Die Stack | Center To Center Delta, Pin Based | cc |
| Die Stack | Die Stack Height | sh |
| Die Stack | Die Stack to Die Stack Spacing | dd |
| Die Flag | Die Flag to Die Flag Spacing | xx |
| Die Flag | Die Flag to Discrete Component Spacing | xc |
| Die Flag | Die Flag to Finger Spacing | xf |
| Die Flag | Die Flag to Package Edge Spacing | xe |

| | | |
|---|---|---|
| Solder Mask | Continuous Solder Mask Coverage | sm |
| Solder Mask | Minimum Solder Mask Shape | sm |
| Solder Mask | Minimum Solder Mask Void | sm |
| Solder Mask | Solder Mask To Die Edge Spacing | sd |
| Solder Mask | Solder Mask to Package Edge Spacing | se |
| Solder Mask | Solder Mask To Solder Mask Spacing | ss |
| Package Substrate | Any Metal To Any Metal Spacing | mm |
| Package Substrate | Cline to Via Overlap | vo |
| Package Substrate | Conductor to Package Edge Spacing | me |
| Package Substrate | Discrete Component Pad to Finger Spacing | pf |
| Package Substrate | Discrete Component Pad to Package Edge | pe |
| Package Substrate | Exposed Metal to Exposed Metal Spacing | mm |
| Package Substrate | Finger to Package Substrate Spacing | fe |
| Package Substrate | Minimum cline segment | cs |
| Package Substrate | Trace Extension from Finger | te |
| Package Substrate | Via to Package Substrate Edge Spacing | ve |
| Shape | Acute Angle Shape Boundary | aa |
| Shape | Minimum Shape Check | ms |
| Shape | Minimum Void Check | ms |
| Acute Angle Metal | Acute Angle Routing | aa |
| Acute Angle Metal | Merged Metal Minimum Angle | aa |
| Acute Angle Metal | Trace Minimum Angle to Pad | aa |
| Acute Angle Metal | Trace Minimum Angle to Shape | aa |
| Acute Angle Metal | Trace Minimum Angle to Trace | aa |

| | | |
|---|---|---|
| Miscellaneous | Conductor Shape Void Overlap | vo |
| Miscellaneous | Degassing Void Overlap | vo |
| Miscellaneous | Tombstone Check | ts |

4

# Renaming Standard Components

This chapter describes how to rename standard components using the automatic renaming feature in the layout editor, *Logic – Auto Rename RefDes*.

After you place and route a new design or rearranged components on an existing design, reference designators must normally be renamed to aid accurate testing and assembly on the actual design.

The automatic renaming process lets you rename *every* component on a design in a single operation without having to attach the AUTO_RENAME property to each component. Renaming occurs on both sides of the design in a single operation.

⚠ When you choose this method, the prefixes stored in the symbol definition are used in the new reference designators. In addition, separate counters run for components with different prefixes.

You can also elect to rename individual components by attaching the AUTO_RENAME property to them or by renaming components on one side of the design only.

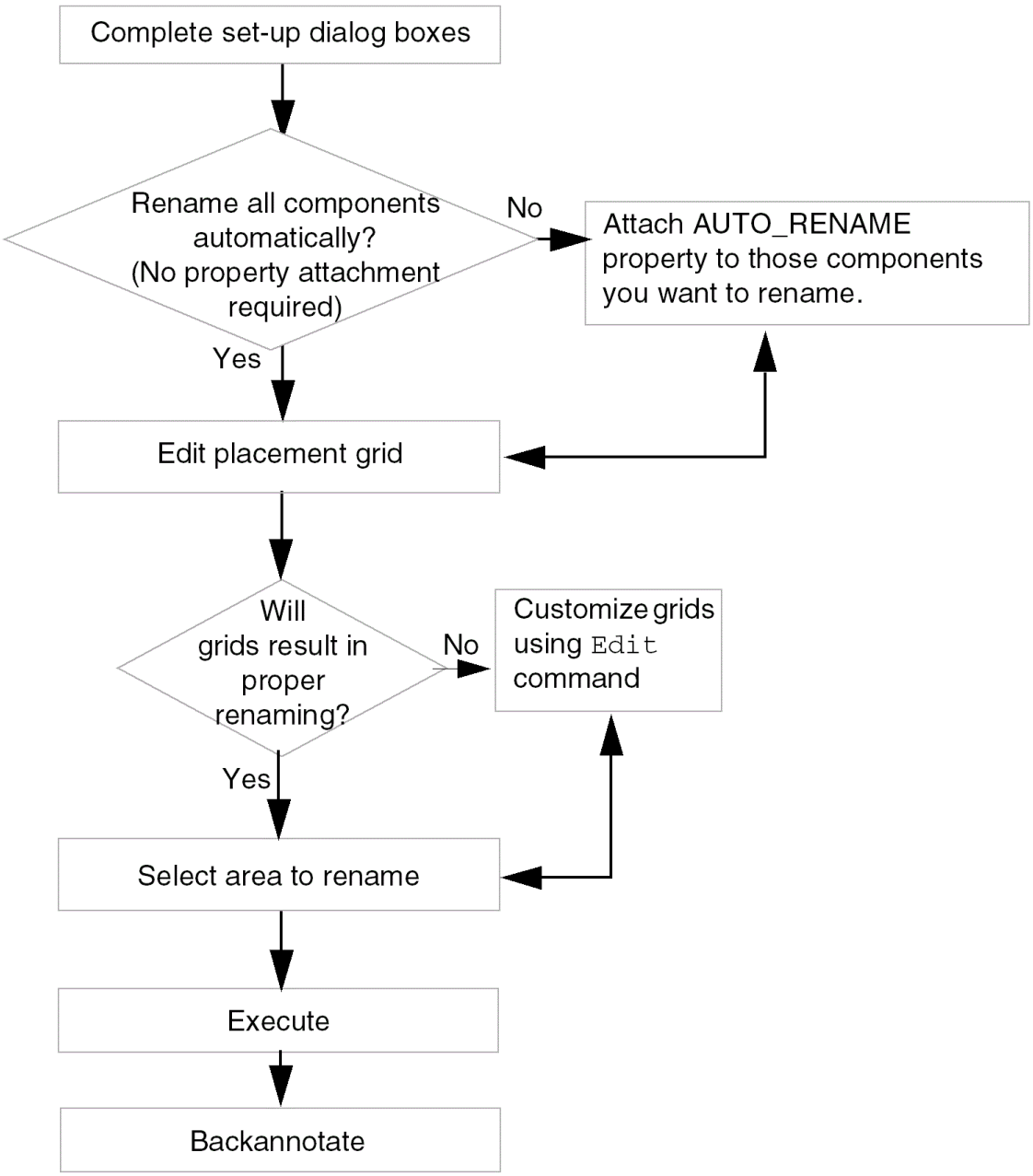
Renaming is controlled by placement grid line locations only (user-defined or default selection) or by sequential renaming within grid blocks. With grid-based renaming, you can designate the direction (horizontal or vertical) and order (left-right, right-left, upwards-downwards) of the renaming process. Additionally, you can define grid descriptions by alpha characters and/or integers.

All of these options are accessed by selections on a graphical user interface, defined in detail in this chapter.

## The Auto Ref Des Rename Flow

The automatic renaming function is a straight-forward, easy to implement feature. The following figure illustrates the steps necessary to perform this operation.

## The Automatic Renaming Flow



# Setting Automatic Rename Options

The options available in the automatic reference designator (auto refdes) rename feature are available only through the Rename Ref Des dialog boxes. There are two dialog boxes:

- Rename Ref Des
  This dialog box is accessed through the menu bar. It lets you choose whether to rename all the design components or to choose specific components for renaming. It also lets you choose a placement grid configuration.
- Rename Ref Des Set Up
  The Set Up dialog box is accessed through the Rename Ref Des dialog box and contains the balance of the options available in automatic rename, including layer options, reference designator format, and renaming method.

Options can be set in any order; that is, the options, format, renaming methods, and so on in the Set Up dialog box.

For procedural information, see *Logic – Auto Rename RefDes – Rename* (rename param command) in the *Allegro PCB and Package Physical Layout Command Reference*.

# Executing the Rename Function

Once you set up all options in the rename dialog boxes, you execute the rename process in a particular area.

1. Select the portion of the design to be renamed.
    - *Logic – Auto Rename RefDes – Design* (rename area design command) that is, the entire design.
    - *Logic – Auto Rename RefDes – Room* (rename area room command) opens a fill-in that requires you to enter a room name.
    - *Logic – Auto Rename RefDes – Window* (rename area window command) lets you choose a defined area of the design.
    - *Logic – Auto Rename RefDes – List* (rename area list command) opens a window that displays what area is currently active.
2. From the Rename menu, click *OK*.
   The status line in the console window prompt displays the message

   Auto Rename of Reference Designators IN PROGRESS.

   As the operation proceeds, various warning or error messages may appear at the console window prompt. When the operation finishes, the status line reads:

   Auto Rename of Refdes COMPLETE. x components renamed.

The layout editor running on a UNIX workstation provides a batch program called reftxt that uses a text file to rename reference designators. In the text file, you can indicate changes anywhere on the design and allow the reference designator to have any number of characters. You run this program after components are placed.

## Creating a Rename Text File

The rename text file that you create with a text editor is a "was/is" list of reference designators. Each line describes one reference designator to be changed, followed by at least one space or tab, then the reference designator to be substituted. Reference designators can be listed in any order, and previous ones do not affect those further down the list. A sample list might appear as follows:

```
U1    U12
```

```
R3    R45

U41   R45

U10   U9
```

The `reftxt` command can also be used to accommodate reference designators that might otherwise be too long for the automatic rename function. For example, reftxt can be used to change reference designators to include part numbers or other company-defined information. Below is a sample text file for this type of situation:

```
Z1    A007421

Z2    A007422

C1    A443011

C2    A443012
```

If you plan to change to lengthy designators, make sure that the symbols have been built with a reference designator placement that accommodates them.

# Attaching the AUTO_RENAME Property

When you run the rename process on an individual component or on one group of components at a time, you must define the components to be renamed by attaching the AUTO_RENAME property to them individually.

⚠ When you are renaming groups of reference designators, you do not want to include certain components in the renaming process. To prevent these components from being renamed, attach the HARD_LOCATION property to them.

# Editing the Placement Grid

The layout editor defines the placement of components within rows and columns based on their relationship to placement grid lines. As you rename a group of components, you might need to edit the grid to a size appropriate for those components.

Before executing the `rename` command, examine the grids in the assigned area to ensure appropriate results. The established placement grid may work, but you might have to manually alter the grid, or define a new grid. If you are renaming various differently sized components, you may need to resize the grid for each set of components.

To create a grid:

1. Create a package keepin area large enough to contain the area of the grid.
2. From the menu bar in the layout editor, choose *Place – Autoplace – Top Grids* (place set topgrid command) or *Place – Autoplace – Bottom Grids* (place set bottomgrid command).
   The place set grid dialog box appears for the selected grid type.
3. In the dialog box field, enter an x coordinate, then click *OK*. (See Creating a Non-Etch Grid for Interactive Placement in Preparation for Placing Elements for details.)
   The Place Set Grid dialog box for the y coordinate appears.
4. Enter a y coordinate, then click *OK*.
5. Enter the grid point where you want the grid to originate.
   You can indicate this by a mouse pick or by entering explicit x and y coordinates inside the placement keepin.

You can customize grid lines by choosing *Edit – Move* (move command), *Edit – Copy* (copy command), and *Edit – Delete* (delete command).

For additional information on working with grids, see Setting Placement Grids in Preparation for Placing Elements.

5

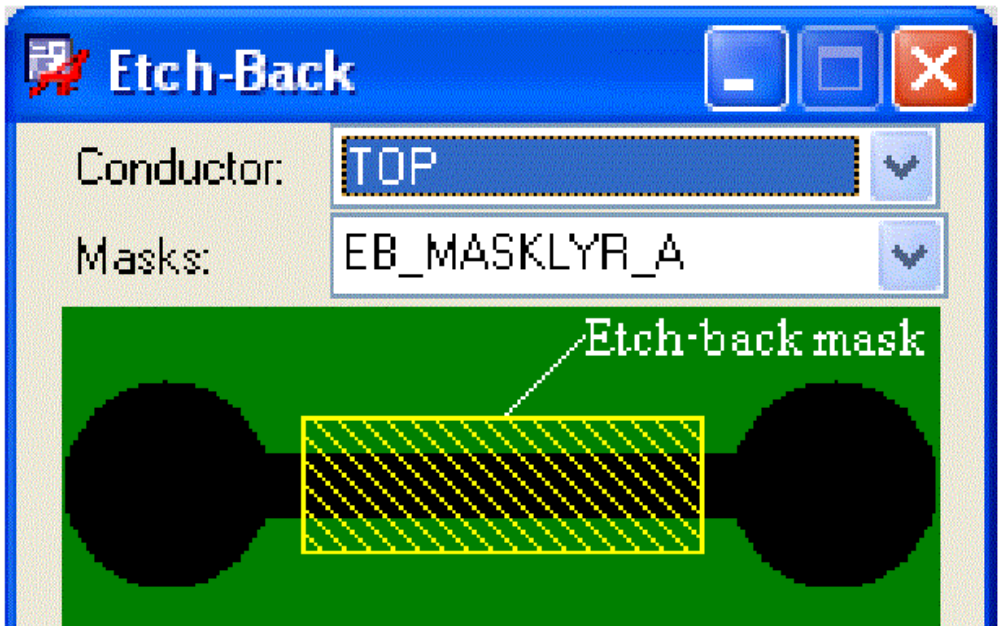
# APD: Etch-Back

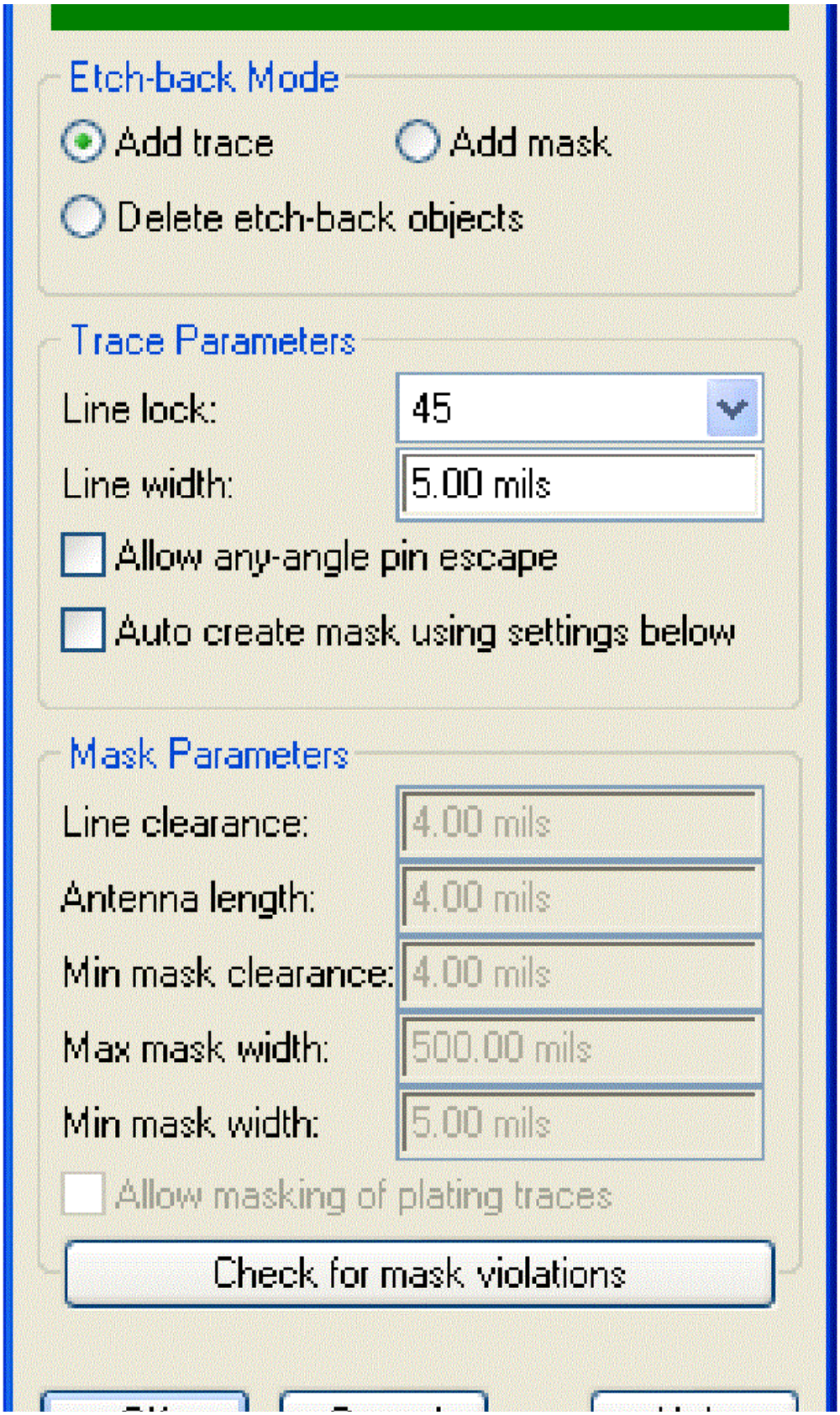
## Etch-Back Support

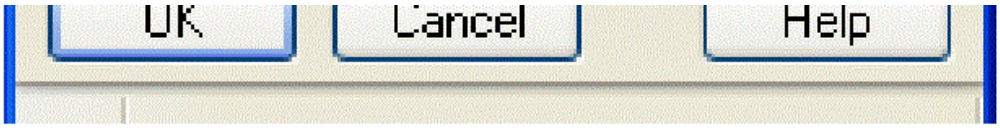
The APD/SiP tools now support an etch-back tool that simplifies the process of creating the required net-to-net connections to ensure plating bar connectivity and generating the masks that define where to remove the lines to break the shorts. After you complete the final substrate routing, you run the `pbar check` command, which identifies any nets that are not yet connected to the plating bar. If there are unplated nets, you run the `etchback` command to generate the necessary connections. You create all the etch-back traces (shorting clines and lines) and masks (used to expose etch-back routing that will be etched off the substrate). The shorting element consists of two cline objects at each end of the etch-back trace where the short is not covered by the mask and a non-conductive line segment that lies underneath the mask, therefore, not impeding signal integrity analysis.

If you make any changes to the component routing, you need to delete the affected etch-back masks and traces and then re-run the tool to create traces and masks that match the changes.

**Etch-Back Mask Parameters Dialog Box**




Etch-back Mode
Add trace
Add mask
Delete etch-back objects
Trace Parameters
Line lock:
45
Line width:
5.00 mils
Allow any-angle pin escape
Auto create mask using settings below
Mask Parameters
Line clearance:
4.00 mils
Antenna length:
4.00 mils
Min mask clearance:
4.00 mils
Max mask width:
500.00 mils
Min mask width:
5.00 mils
Allow masking of plating traces
Check for mask violations

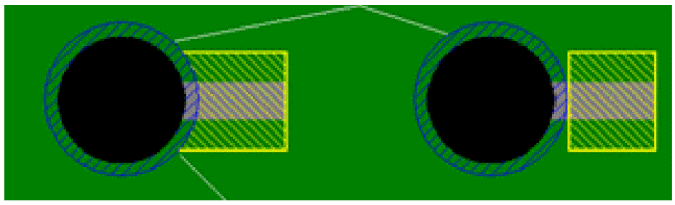



For information on using this tool, see the `etchback` command in the *Allegro PCB Package and Physical Layout Command Reference.*

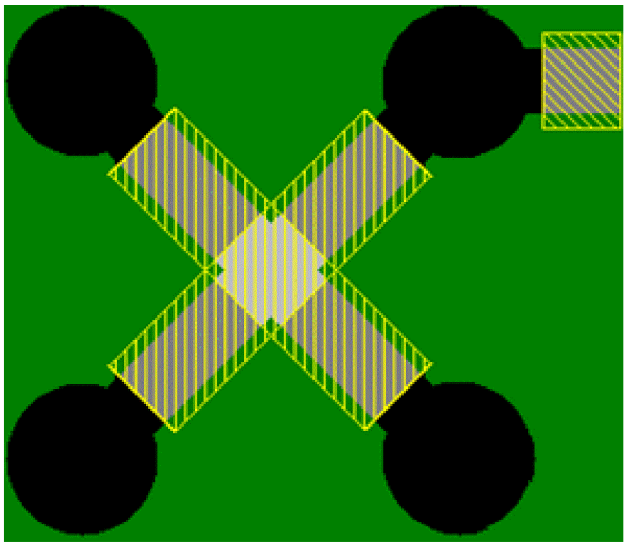
## Etch-Back Masks

When you create the etch-back masks, the etch-back tool avoids using acute angles as these can cause acid traps.

### Preferred Etch-Back Mask



### Removal of Etch-Back Traces



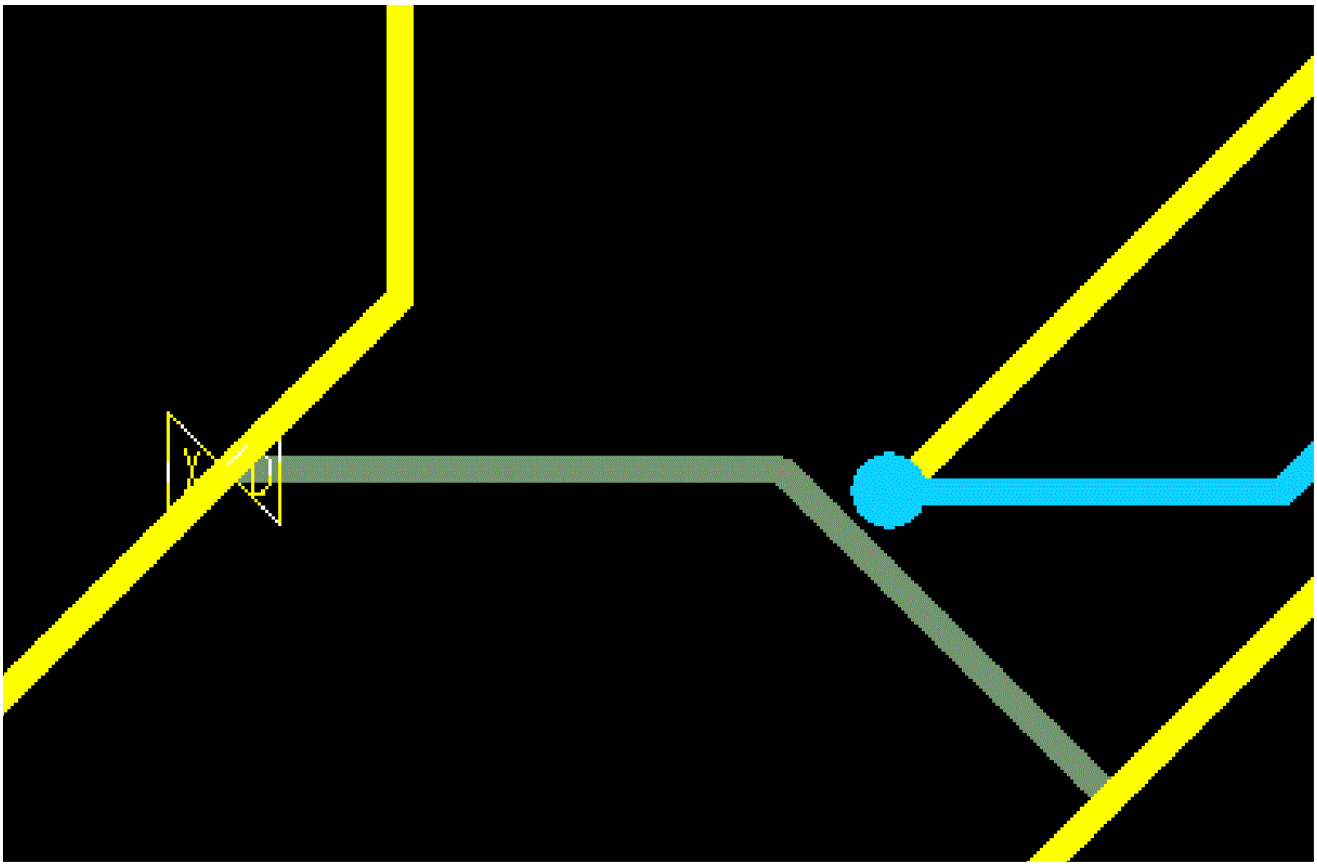
## Related DRC Errors

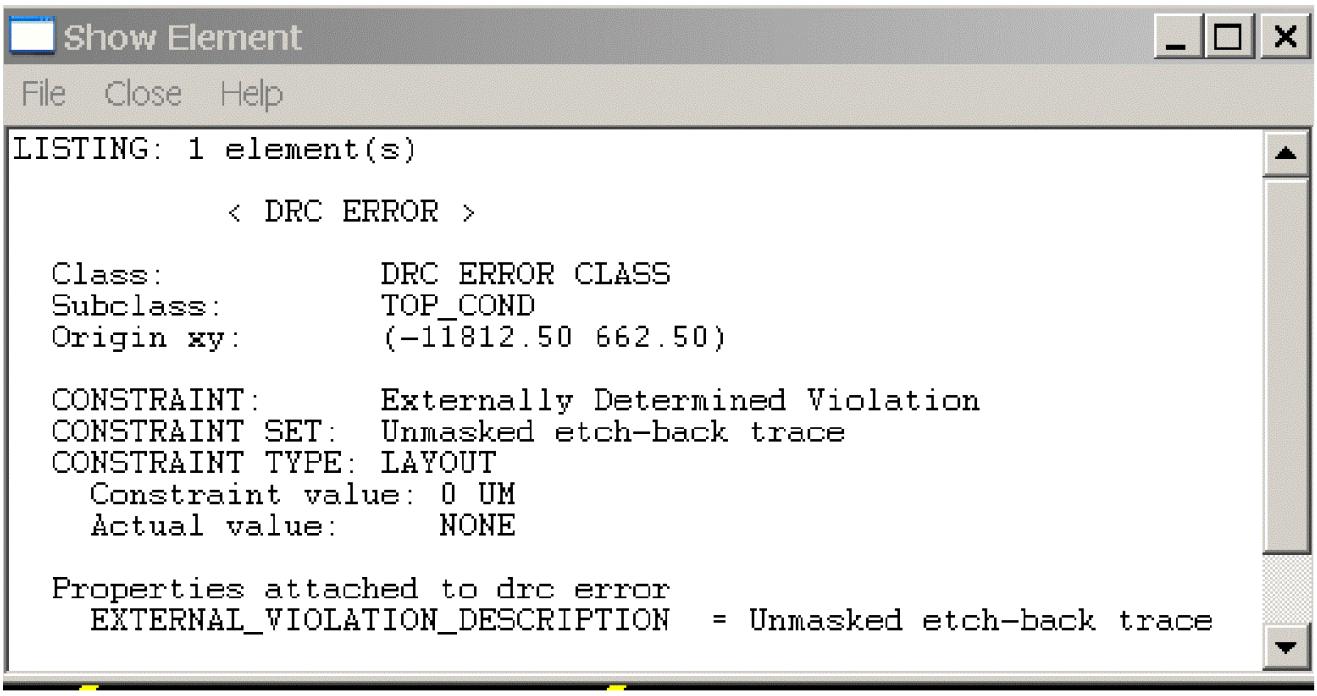
The following describes the two DRC errors that can occur when running `etchback` and related commands.

| DRC Error | Description |
|---|---|
| Unmasked Etch-Back Trace | This DRC marker attaches to any etch-back trace missing its respective mask. |
| Minimum Etch-Back Registration Parameter Violation | Indicates the distance between the etch-back mask and any (regular) conductors. This DRC marker attaches to an etch-back trace when the tool cannot create the etch-back mask without violating the user-specified parameter value. |

The following figure shows an unmasked etch-back trace. The Show Element window indicates that an Unmasked Etch-Back trace DRC error has occurred.

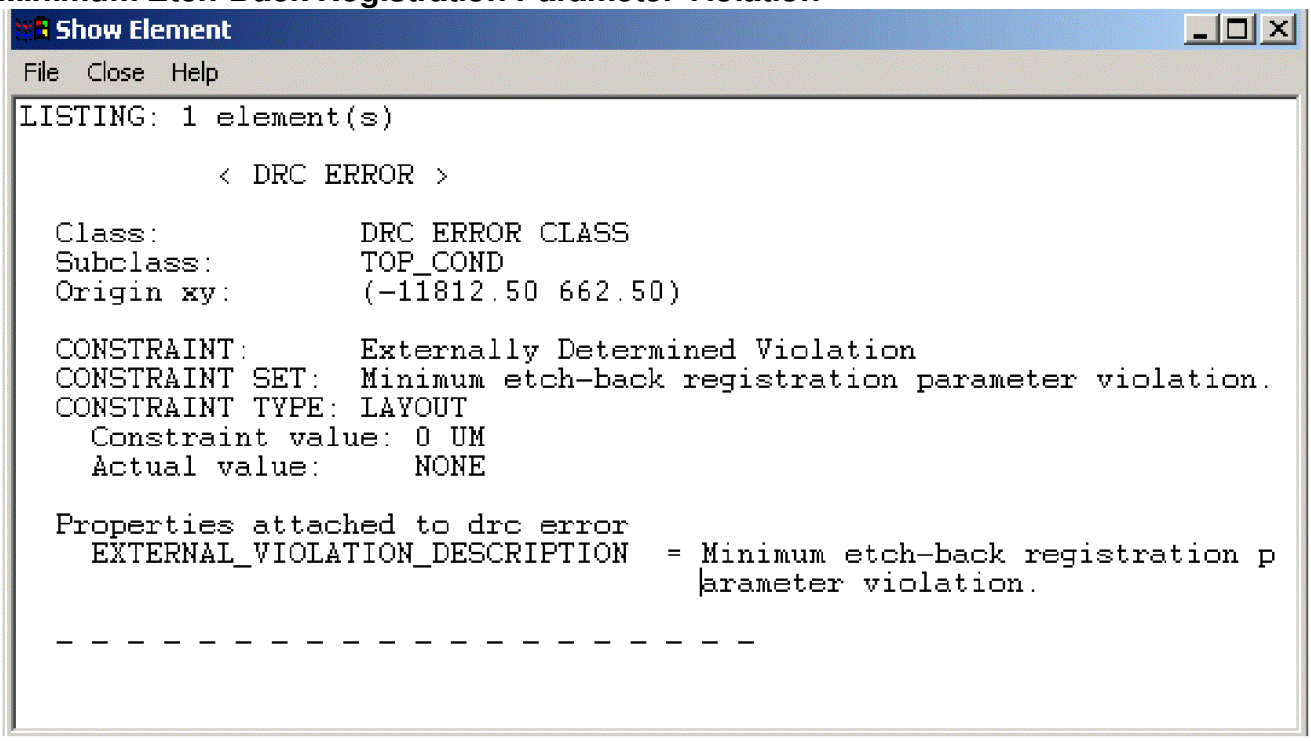
## Unmasked Etch-Back Trace DRC Error

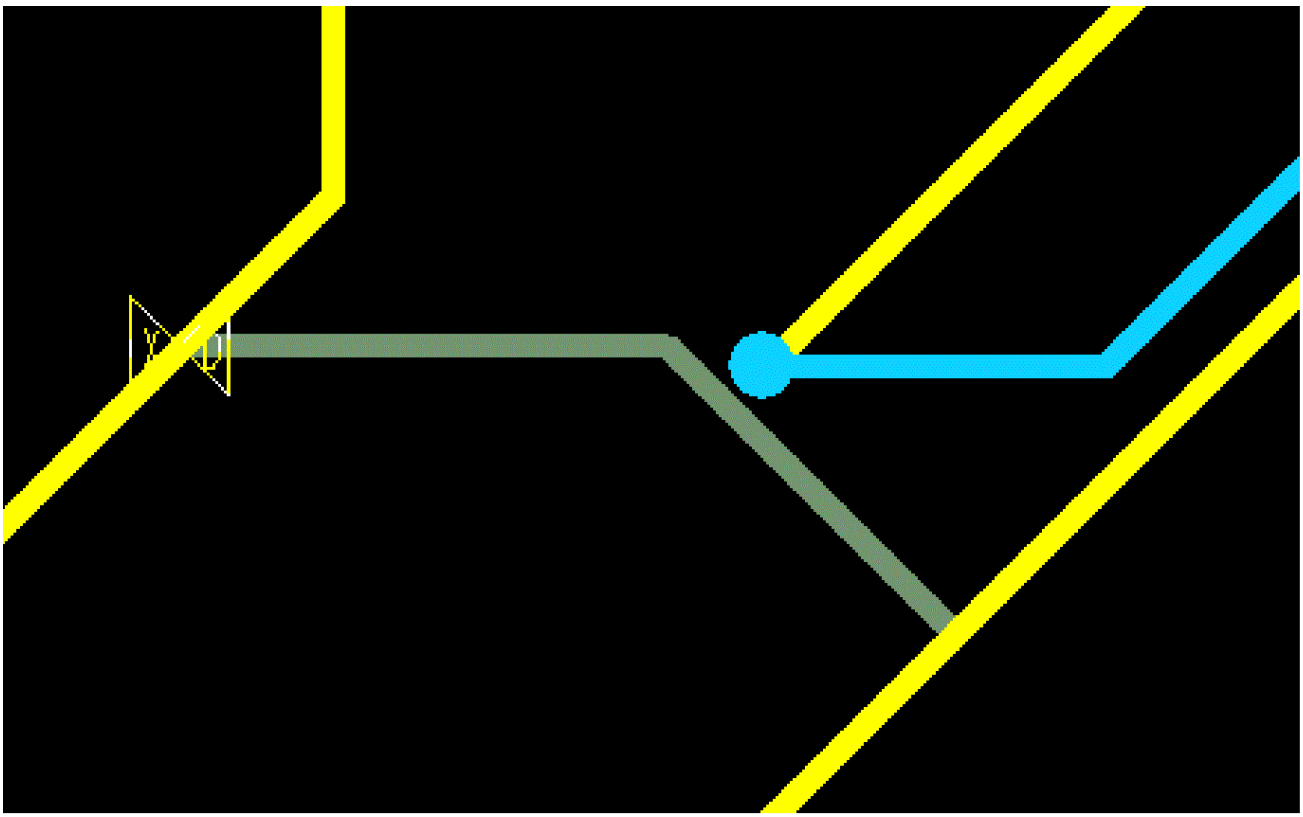





The design rule checking marker shown in the following figure indicates that the tool cannot create an etch-back mask without violating the value between the etch-back mask and an adjacent conductor.

## Minimum Etch-Back Registration Parameter Violation



6

# Extracting Views from the Layout Editor

The `extracta` command, available in the layout editor, reads binary representations of the Allegro editor designs, then translates that data into ASCII text files. These files contain database information you can sort and format for reporting and analyzing.

The `extracta` command is specific to the Allegro database; however, it produces data in a form that standard processing programs can use. The following figure illustrates the flow of data in the extract process.

You control `extracta` by creating an extract command file that `extracta` uses as input. In the extract command file, you list those data fields you want extracted from the Allegro database and establish selection criteria for certain data values. The information you provide is called a *view*.
The `extracta` command provides predefined views from which you can choose (baseview files). You can also customize views, and extract multiple views.

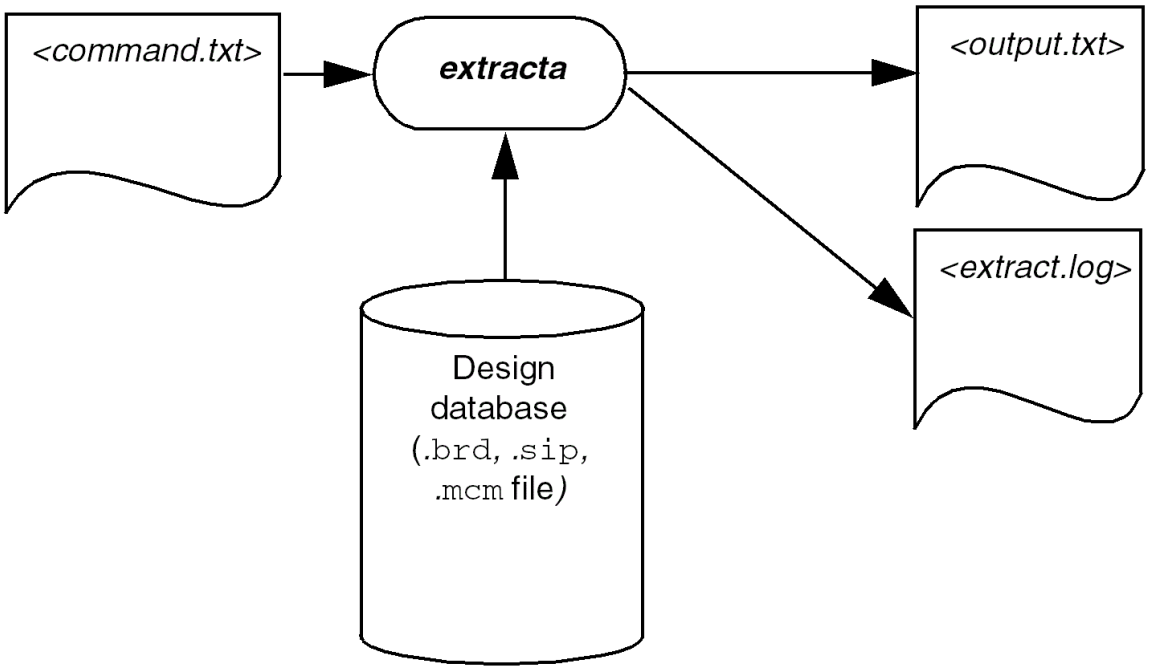
The `extracta` command creates an output file of the information you specified in the command file. The output file is composed of many text record representations of the database that you specified to `extracta`. You can rerun `extracta`, specifying other database elements to obtain a different view of the database.

In addition to the output file, `extracta` creates a log file named `extracta.log` , which contains a copy of the control file, as well as any error messages.

Although the extract command can be run any time during the design process, normally you run it on a completed design. Extracting data views involves the following tasks:

- Creating an extract command file
- Running `extracta`
- Formatting the output file

### Data Flow for the `extracta` Command




In versions prior to 14.2, the `extracta` program was called `extract`. If you are operating on a UNIX platform, Cadence provides a link to the old extract name so you do not need to change your scripts. This is not the case on Windows platforms. If you are operating on a Windows system, you must create new scripts for use with `extracta`.

## Related Topics

- extracta

# Extract Command File Format

The extract command file contains a number of *records*. Each record is a line of the text file (terminated by an end-of-line character). The following rules apply:

- `extracta` ignores blank lines.
- `extracta` ignores preceding or trailing white space, as well as white space around the selection record operator (= or !).
- You can enclose field names or values in single or double quotes.
- Field names, view names, and the OR and END keywords can be in uppercase or lowercase. The case of field values is left unchanged.
- Two successive single (or double) quotes means a null string.
  A null string is needed to choose records that have a data field with a null (non-existent) value.
  For example, the x and y coordinates of a component's symbol are written as a null value if the component is not placed.

### Record Types

| | |
|---|---|
| **Comment Records** | Lines that start with a pound sign (#). Comments are not processed but are written to the log file. |
| **Viewname Record** | The first line in the file—or following an END record—that is not a comment record or blank line. The record contains the name of the view that defines the type of element to be extracted. For example: COMPONENT The view name is edit checked. Descriptions of the view are provided later in this chapter. |
| **Data Field Record** | Each line that is not a comment, viewname, selection, or OR and END record contains the name o f a data field in the database. For example: COMP_CLASS SYM_NAME |
| | Each occurrence of a data field record causes a field to be written to the text file for each entity. The names of the data fields specified are included in the first header record of the output file. The data field name is checked for validity. |
| **Selection Records** | Any record that contains an equal sign (=) is a selection record. The format of the selection record is <data_field> <operator> <data_value>, where <operator> is either = (equal) or != (not equal). Do not include a space between the ! and =. Multiple selection records are implicitly treated as an AND, that is, a record is chosen only if all selection statements are TRUE. |
| | For example, to specify legal class and subclass values for a filter, type the following: CLASS = "DRAWING FORMAT" SUBCLASS = "TITLE_BLOCK" |
| **OR Record** | A record with only the word OR. The OR record is used to make choose statements connected with an OR rather than the implicit AND. If an entity does not satisfy the selection criteria of the choose statement(s) before the OR, then those after the OR–and before the next OR–are evaluated. |

## Property Names

Property names can be added to command files in the form of data field records when you want to extract a property from any element that might have that property, or from a specific element type in the view. In your command file, you can specify a property to be extracted by

- Specifying the property name alone (Example 1)
- Combining the property with an object type (Example 2)

# Example 1: Property Extract

The following example command file requests `extracta` to write out the MIN_LINE_WIDTH property value, a property that may be attached either to clines or nets. With only this information, `extracta` looks for the property on both the cline and the net that owns the cline, searching in that order. The cline property supersedes the net property in the output.

```
#ExtractMIN_LINE_WIDTH,
#wherever it may be:
GEOMETRY
  CLASS
  SUBCLASS
  RECORD_TAG
  GRAPHIC_DATA_NAME
  GRAPHIC_DATA_NUMBER
  GRAPHIC_DATA_1
  GRAPHIC_DATA_2
  MIN_LINE_WIDTH
END
```

Property extract ———

# Example 2: Object Type Property Extract

Specifying *<object_type>* and *<property_name>* causes `extracta` to look for the property only on that object type, where *<object_type>* is BOARD, SYM, COMP, FUNC, GEO, NET, PIN, or VIA and *<property_name>* is the name of the property to be extracted.

This example asks for the net MIN_LINE_WIDTH property. In this case, `extracta` extracts the property only if it is attached to the net:

```
                    # Extract MIN_LINE_WIDTH if attached to the net
                    GEOMETRY
                      CLASS
                      SUBCLASS
                      RECORD_TAG
                      GRAPHIC_DATA_NAME
                      GRAPHIC_DATA_NUMBER
                      GRAPHIC_DATA_1
                      GRAPHIC_DATA_2
Net property extract ——— NET_MIN_LINE_WIDTH
                    END
```

## Property Search Hierarchy

When looking for a property that does not have an `<object_type>` qualifier in the command (and therefore may be found on any object in the view), `extracta` searches the view objects from the bottom up. It first looks for the property on objects owned by the view object, then on the view object itself.

The following table shows the hierarchical order `extracta` uses when searching for a property. It also shows the prefix you attach to a property name in the command file when you want the property to be found only on a particular element types. The `extracta` program searches for the property from the top of the list.

### Hierarchical Order the `extract` Command Uses

| Element Type | Prefix Name |
|---|---|
| Standalone Pin | PIN_ |
| Symbol Pin | PIN_ |
| Function Pin Instance | PIN_ |
| Function Pin Definition | PIN_ |
| Component Pin Definition | PIN_ |
| Via | VIA_ |

| | |
|---|---|
| Line | GEO_ |
| Cline | GEO_ |
| Void | GEO_ |
| Shape | GEO_ |
| Rectangle | GEO_ |
| Filled Rectangle | GEO_ |
| Figure | GEO_ |
| Text | GEO_ |
| Net | NET_ |
| Group | GRP_ |
| Symbol Instance | SYM_ |
| Symbol Definition | SYM_ |
| Function Instance | FUNC_ |
| Function Definition | FUNC_ |
| Component Instance | COMP_ |
| Component Definition | COMP_ |
| Board | BOARD_ |

## Group Properties

You can extract the properties attached to groups by adding a GRP_ prefix to the property name and including the field in one of the base views. For example:

```
NET

GRP_TOTAL_ETCH_LENGTH

END
```

This extracts a record for every net in the design and indicate which nets belong to a group that has a TOTAL_ETCH_LENGTH property attached.

## Multiple Views

You can extract multiple views by using multiple END records in the `extract` command file. The END record delimits multiple extracts from a single command file. An END causes the `extracta` program to be started with the current view and fields. After completion, it reads the next line in the command file.

> ⚠ Succeeding data acquired from the database with the `extracta` command is written to the corresponding output file from the command line. Therefore, when extracting multiple views, you must specify multiple output file names.

The following is a sample extract file using multiple views:

```
# Note:
#   This command file contains two views:
#   component and component_pin
#
COMPONENT
#
COMP_PACKAGE
COMP_DEVICE_TYPE
COMP_VALUE
COMP_TOL
REFDES_SORT
REFDES
#
END
#
#   component pin view for the pin data
#
COMPONENT_PIN
#
REFDES  != ''
NET_NAME != ''
#
NET_NAME_SORT
NET_NAME
REFDES_SORT
REFDES
PIN_NUMBER_SORT
PIN_NUMBER
#
END
```

## SORT Data Fields

You can sort extracted data files with these data fields:

- FUNC_DES_SORT
- NET_NAME_SORT
- PIN_NUMBER_SORT
- REFDES_SORT

The `extracta` command derives each of these data fields from its corresponding original data field (for example, FUNC_DES_ SORT from FUNC_DES) by separating the field into an alphabetic and

numeric subfield, and expanding the numeric subfield to a wide, right-justified field. This assures that a standard sort of the file results in a reasonable sort order. For example, the function designators FUNC1, FUNC2, FUNC3, and FUNC10 sort to:

- FUNC1
- FUNC10
- FUNC2
- FUNC3

⚠ When using the standard ASCII `sort` command, all 1s in each column must precede all 2s, and so on. The *_SORT* data fields overcome this problem by transforming the function designators: FUNC1 to FUNC 00000001, FUNC2 to FUNC 00000002, FUNC3 to FUNC 00000003, FUNC10 to FUNC 00000010, and so on. This results in a sort to:

- FUNC 00000001
- FUNC 00000002
- FUNC 00000003
- FUNC 00000010

Use the *_SORT* fields by putting both the data fields *_SORT* and the data field name itself in your command file. Make sure to put the *_SORT* data field name first, so that it controls the sort ordering. You can then read the corresponding (untransformed) *FUNC_DES* data field itself for display and printing.

## Baseview Files

The layout editor provides a number of text files that contain chosen data fields, called `baseview` files. Baseview (`_bv`) files are located at `\cds\share\pcb\text\views` in the install directory of the software.

Baseview files can assist you in writing a command file since, typically, only one type of data is extracted from your design. For example, all component reference designators or all symbol pin x-y coordinates. As baseview files contain all the basic data fields associated with a particular view, you can include any number of data fields related to an element by simply copying part or all of a `.bv` file and pasting it into your command file. You can then enter specific values for the data fields (pins with pin number = 1, drill size =.029, and so on). You can also combine selection records.

You can extract data using any one of these predefined views:

| VIEW | BASEVIEW FILE |
|---|---|
| BOARD | `board_bv.txt` |
| COMPONENT | `comp_bv.txt` |
| CONNECTIVITY | `conn_bv.txt` |
| COMPOSITE_PAD | `cpad_bv.txt` |
| COMPONENT_PIN | `cpin_bv.txt` |
| FULL_GEOMETRY | `fgeo_bv.txt` |
| FUNCTION | `func_bv.txt` |
| GEOMETRY | `geo_bv.txt` |
| LAYER | `layer_bv.txt` |
| LOGICAL_PIN | `lpin_bv.txt` |
| NET | `net_bv.txt` |
| RAT_PIN | `rat_bv.txt` |
| SYMBOL | `sym_bv.txt` |

Extract Data Dictionary contains all the data fields that you can extract and the views in which they are available.

## Baseview File Contents

The following sections show the contents of the baseview files in the layout editor.

⚠ The property name of any property that is legal for the object being extracted by a view file can be added to the extracta command file and will correspondingly be added to the list of fields output when the property exists on an object.

# BOARD

This view extracts the basic data about the design extents and scaling. `extracta` supplies the data automatically as the second J record of every extraction, so you usually do not need to extract the data with a separate view file.

```
#
# board_baseview - basic fields for the BOARD view
#
BOARD
 BOARD_NAME
 BOARD_ACCURACY
 BOARD_UNITS
 BOARD_EXTENTS_X1
 BOARD_EXTENTS_Y1
 BOARD_EXTENTS_X2
 BOARD_EXTENTS_Y2
 BOARD_LAYERS
 BOARD_THICKNESS
 BOARD_DRC_STATUS
 BOARD_SCHEMATIC_NAME
 BOARD_BOARD_THICKNESS
END
```

# COMPONENT

This view extracts components in the design. The basic identifier of each record is the reference designator of the component.

```
COMPONENT
  REFDES_SORT

  REFDES
  COMP_CLASS
  COMP_PACKAGE
  COMP_DEVICE_TYPE
  COMP_VALUE
  COMP_TOL

  COMP_MAX_POWER_DISS_DEVICE
  COMP_MAX_POWER_DISS_INSTANCE
```

```
  COMP_ALT_SYMBOLS
  COMP_AUTO_RENAME
  COMP_COMPONENT_WEIGHT
  COMP_DENSE_COMPONENT
  COMP_DEVICE_LABEL
  COMP_FIX_ALL
  COMP_HARD_LOCATION
  COMP_HEIGHT
  COMP_INSERTION_CODE
COMP_MAX_POWER_DISS
  COMP_NO_MOVE
  COMP_NO_PIN_ESCAPE
  COMP_NO_ROUTE
  COMP_NO_SWAP_COMP
  COMP_NO_SWAP_GATE
  COMP_NO_SWAP_GATE_EXT
  COMP_NO_SWAP_PIN
  COMP_PART_NUMBER
  COMP_PIN_ESCAPE
  COMP_PLACE_TAG
  COMP_ROOM
  COMP_TERMINATOR_PACK
 COMP_VOLTAGE
  COMP_VOLT_TEMP_MODEL
  COMP_WIRE_BOND

END
```

# COMPONENT_PIN

This view extracts pins of components in the drawing. The basic identifier of each record is the reference designator and pin number of the component pin. The difference between the COMPONENT_PIN view and the LOGICAL_PIN view (described later in this chapter) is that the COMPONENT_PIN view does not include pins of functions not yet assigned to components. Also, common pins (shared between functions) only occur once in the COMPONENT_PIN view, whereas they will occur once per function in the LOGICAL_PIN view.

This view differs from the COMPOSITE_PAD view (described next) in that there are no vias (or stand-alone pins) included in this view. Also, pins of unplaced components are included in this view and not in the COMPOSITE_PAD view.

```
COMPONENT_PIN
```

```
REFDES_SORT

PIN_NUMBER_SORT

REFDES

PIN_NUMBER

PIN_X

PIN_Y

PIN_EDITED

PIN_COMMON_CODE

PIN_SWAP_CODE

PIN_TYPE

PAD_STACK_NAME

NET_NAME

PIN_FLOATING_PIN

PIN_GROUND

PIN_NC

PIN_NO_PIN_ESCAPE

PIN_NO_SHAPE_CONNECT

PIN_NO_SWAP_PIN

PIN_PAD_STACK_NAME

PIN_PINUSE

PIN_PIN_ESCAPE

PIN_POWER

END
```

# COMPOSITE_PAD

This view extracts pad data from symbol pins and vias in the design. You can use this view to get padstack and drill hole data.

```
COMPOSITE_PAD
```

```
CLASS

PAD_STACK_NAME

PAD_STACK_INNER_LAYER

PAD_STACK_TYPE

GRAPHIC_DATA_NAME

GRAPHIC_DATA_NUMBER

GRAPHIC_DATA_1

GRAPHIC_DATA_2

GRAPHIC_DATA_3

GRAPHIC_DATA_4

START_LAYER_NAME

START_LAYER_NUMBER

END_LAYER_NAME

END_LAYER_NUMBER

REFDES

PIN_NUMBER

PIN_X

PIN_Y

TEST_POINT

VIA_MIRROR

VIA_X

VIA_Y

NET_NAME

DRILL_HOLE_NAME

DRILL_HOLE_X

DRILL_HOLE_Y

DRILL_HOLE_PLATING

DRILL_ARRAY_ROWS
```

```
DRILL_ARRAY_COLUMNS

DRILL_ARRAY_CLEARANCE

DRILL_ARRAY_LOCATIONS

DRILL_FIGURE_CHAR

DRILL_FIGURE_SHAPE

DRILL_FIGURE_WIDTH

DRILL_FIGURE_HEIGHT

DRILL_FIGURE_ROTATION

VIA_NO_SHAPE_CONNECT

VIA_PAD_STACK_NAME

END
```

# CONNECTIVITY

This view extracts data about electrical connections. The nets in the design are used to create records that represent a node or a connection. A node refers to the pins, vias, and Ts that are part of a logical net. A connection refers to ratsnest or etch geometry. The CONNECTIVITY baseview contains the following fields:

⚠ You can edit this baseview to add the `WIREBOND_PROFILE_NAME` property if, for example, you need to separate wires into their respective groups, or map them to some curvature.

```
CONNECTIVITY

#

NET_NAME != ''

RAT_CONNECTED != 'YES'

#

NET_NAME_SORT

NODE_SORT

NODE_1_NUMBER

NODE_2_NUMBER
```

```
RECORD_TAG

CLASS

SUBCLASS

NET_NAME

GRAPHIC_DATA_NAME

GRAPHIC_DATA_NUMBER

GRAPHIC_DATA_1

GRAPHIC_DATA_2

GRAPHIC_DATA_3

GRAPHIC_DATA_4

GRAPHIC_DATA_5

GRAPHIC_DATA_6

GRAPHIC_DATA_7

GRAPHIC_DATA_8

GRAPHIC_DATA_9

GRAPHIC_DATA_10

NODE_CONNECTS

RAT_CONNECTED

REFDES

PIN_NUMBER

PIN_TYPE

PIN_X

PIN_Y

VIA_X

VIA_Y

VIA_MIRROR

PAD_STACK_NAME

START_LAYER_NAME
```

```
END_LAYER_NAME

COMP_DEV_TYPE

COMP_TERMINATOR_PACK

COMP_VALUE

SEG_CAPACITANCE

SEG_INDUCTANCE

SEG_IMPEDANCE

SEG_PROPAGATION_DELAY

SEG_RESISTANCE

END
```

# FULL_GEOMETRY

This view contains the data in the GEOMETRY view plus detailed pad data. Pad data is defined as the actual pad in use for each subclass of a pin or via.

Standard geometries have the appropriate geometry (for example, CIRCLE) as the GRAPHIC_DATA_NAME, as well as their PAD_SHAPE_ NAME.

Pads that are arbitrary shapes are presented as a shape. The GRAPHIC_DATA_NAME is LINE (or ARC), and GRAPHIC_DATA_10 is SHAPE. Additionally the PAD_SHAPE_NAME field contains the shape symbol name (preceded by FIG_SHAPE and space). The SYM_TYPE and SYM_NAME fields still reflect information about the parent symbol for pins and are empty strings for vias.

```
FULL_GEOMETRY

   CLASS

   SUBCLASS

   RECORD_TAG

   GRAPHIC_DATA_NAME

   GRAPHIC_DATA_NUMBER

   GRAPHIC_DATA_1

   GRAPHIC_DATA_2

   GRAPHIC_DATA_3
```

```
GRAPHIC_DATA_4

GRAPHIC_DATA_5

GRAPHIC_DATA_6

GRAPHIC_DATA_7

GRAPHIC_DATA_8

GRAPHIC_DATA_9

GRAPHIC_DATA_10

REFDES


PIN_NUMBER

PAD_STACK_NAME

PAD_SHAPE_NAME

PAD_TYPE

PAD_FLASH

DRILL_HOLE_X

DRILL_HOLE_Y

SYM_NAME

SYM_TYPE

NET_NAME

PIN_X

PIN_Y

VIA_X

VIA_Y

SEG_CAPACITANCE

SEG_IMPEDANCE

SEG_INDUCTANCE

SEG_PROPAGATION_DELAY

SEG_RESISTANCE
```

```
END
```

# FUNCTION

This view extracts functions in the drawing. The basic identifier in this view is the function designator. If the function is assigned to a component, you can extract any of the component's data fields. If the component, in turn, is placed, you can extract any of the symbol's data fields.

```
FUNCTION

   FUNC_DES_SORT

   FUNC_DES

   COMP_DEVICE_TYPE

   FUNC_TYPE

   REFDES

   FUNC_SLOT_NAME

   FUNC_SPARE_FLAG

   FUNC_GROUP

   FUNC_HARD_LOCATION

   FUNC_LOGICAL_PATH

END
```

# GEOMETRY

This view extracts geometric elements of the design—the absolute coordinates of each element, formatted into data fields named GRAPHIC_DATA_n fields. Each n field has a different meaning depending on the type of geometric element it describes. See Extract Data Dictionary for a complete listing.

You can also extract any of the properties attached to any geometric element. For example, you can use GEOMETRY view data to extract the following:

- Design outline data for N/C router programming
- Types of geometry on an etch layer for translation to other systems, such as mechanical analysis systems

```
GEOMETRY
```

```
CLASS

SUBCLASS

   RECORD_TAG

   GRAPHIC_DATA_NAME

   GRAPHIC_DATA_NUMBER

   GRAPHIC_DATA_1

   GRAPHIC_DATA_2

   GRAPHIC_DATA_3

   GRAPHIC_DATA_4

   GRAPHIC_DATA_5

   GRAPHIC_DATA_6

   GRAPHIC_DATA_7

   GRAPHIC_DATA_8

   GRAPHIC_DATA_9

   GRAPHIC_DATA_10

   REFDES


   NET_NAME

   SYM_NAME

   COMP_DEVICE_TYPE

   SEG_CAPACITANCE

   SEG_INDUCTANCE

   SEG_IMPEDANCE

   SEG_PROPAGATION_DELAY

   SEG_RESISTANCE

   GEO_FILLET

   GEO_SYMBOL_ETCH

END
```

# LAYER

This view extracts data about the physical layers (for example, etch, multiwire, or dielectric) in the design. Layer information is obtained from the Cross Section parameter form.

```
LAYER

   LAYER_SORT

   LAYER_SUBCLASS

   LAYER_ARTWORK

   LAYER_USE

   LAYER_CONDUCTOR

   LAYER_DIELECTRIC_CONSTANT

   LAYER_ELECTRICAL_CONDUCTIVITY

   LAYER_LOSS_TANGENT

   LAYER_MATERIAL

   LAYER_SHIELD_LAYER

   LAYER_THERMAL_CONDUCTIVITY

   LAYER_THICKNESS

   LAYER_TYPE

END
```

# LOGICAL_PIN

This view extracts function pins in the drawing. The difference between the LOGICAL_PIN view and the COMPONENT_PIN view (previously described) is that the COMPONENT_PIN view does not include pins of unassigned functions (functions not assigned to components). Also, common pins (shared between functions) occur only once in the component pin view, whereas they occur once per function in the logical pin view. The LOGICAL_PIN view does not include "non-function" pins, such as power, ground, and no-connect pins.

```
LOGICAL_PIN

   FUNC_DES_SORT

   PIN_NAME
```

```
    FUNC_DES

    REFDES

    PIN_NUMBER

    PIN_X

    PIN_Y

    PIN_EDITED

    PIN_COMMON_CODE

    PIN_SWAP_CODE

    PIN_TYPE

    PAD_STACK_NAME

    NET_NAME

    PIN_FLOATING_PIN

    PIN_GROUND

    PIN_NC

    PIN_NO_PIN_ESCAPE

    PIN_NO_SHAPE_CONNECT

    PIN_NO_SWAP_PIN

    PIN_PAD_STACK_NAME

    PIN_PINUSE

    PIN_ESCAPE

    PIN_POWER

END
```

# NET

This view extracts net information from the design. The identifier in this view is the net name. You can use this view to extract all properties related to the nets (but not pins of nets; use LOGICAL_PIN or COMPONENT_PIN views for that).

```
NET
```

```
NET_NAME_SORT

NET_NAME

NET_STATUS

NET_CAPACITANCE

NET_ETCH_LENGTH

NET_ETCH_WIDTH_AVERAGE

NET_IMPEDANCE_AVERAGE

NET_IMPEDANCE_MAXIMUM

NET_IMPEDANCE_MINIMUM

NET_INDUCTANCE

NET_MANHATTAN_LENGTH

NET_MANHATTEN_LENGTH

NET_PATH_LENGTH

NET_PROPAGATION_DELAY

NET_RESISTANCE

NET_VIA_COUNT

NET_BUS_NAME

NET_PROPAGATION_DELAY

NET_DIFFERENTIAL_PAIR

NET_DIFFP_2ND_LENGTH

NET_DIFFP_LENGTH_TOL

NET_DRIVER_TERM_VAL

NET_ECL

NET_ECL_TEMP

NET_EXTERNAL_NOISE

NET_FIXED

NET_LOAD_TERM_VAL

NET_RELATIVE_PROPAGATION_DELAY
```

```
NET_MAX_BOND_LENGTH

NET_MAX_BVIA_STAGGER

NET_MAX_EXT_NPOSE

NET_MAX_FINAL_SETTLE

NET_MAX_FIRST_SWITCH

NET_MAX_OHM_LOSS

NET_MAX_OVERSHOOT

NET_MAX_PARALLEL

NET_MAX_PEAK_BXTALK

NET_MAX_PEAK_FXTALK

NET_MAX_PROP_DELAY

MET_MAX_SUM_BXTALK

NET_MAX_SUM_FXTALK

NET_MAX_THERM_SHIFT

NET_MAX_UNDERSHOOT

NET_MAX_VIA_COUNT

NET_MIN_BOND_LENGTH

NET_MIN_BVIA_GAP

NET_MIN_BVIA_STAGGER

NET_MIN_LINE_WIDTH

NET_MIN_NECH_WIDTH

NET_MIN_NOISE_MARGIN

NET_MIN_PROP_DELAYT

NET_NO_GLOSS

NET_NO_PIN_ESCAPE

NET_NO_RAT

NET_NO_RIPUP

NET_NO_ROUTE
```

```
   NET_NO_TEST

   NET_PROBE_NUMBER

   NET_RATSNEST_SCHEDULE

   NET_ROUTE_PRIORITY

   NET_ROUTE_TO_SHAPE

   NET_SAME_NET

   NET_STUB_LENGTH

   NET_TS_ALLOWED

   NET_VIA_LIST

   NET_VOLTAGE

   NET_WEIGHT

END
```

# RAT_PIN

This view extracts the COMPONENT_PIN view with NET_RAT_ SCHEDULE. This view is useful for retrieving the ratsnesting for a net.

```
COMPONENT_PIN

   NET_NAME_SORT

   NET_RAT_SCHEDULE

   NET_NAME

   REFDES

   PIN_NUMBER

   PIN_X

   PIN_Y

END
```

# SYMBOL

The basic identifier in this view is the symbol name (for example, DIP14). Use this view to extract symbol data from the design, whether or not the symbol has a component assigned to it.

```
SYMBOL

   SYM_TYPE

   SYM_NAME

   REFDES

   SYM_BOX_X1

   SYM_BOX_X2

   SYM_BOX_Y1

   SYM_BOX_Y2

   SYM_CENTER_X

   SYM_CENTER_Y

   SYM_EXTENTS_X1

   SYM_EXTENTS_X2

   SYM_EXTENTS_Y1

   SYM_EXTENTS_Y2

   SYM_HAS_PIN_EDIT

   SYM_MIRROR

   SYM_ROTATE

   SYM_X

   SYM_Y

   SYM_LIBRARY_PATH

   SYM_SHAPE_X_OFF

   SYM_SHAPE_Y_OFF

END
```

## Related Topics

- Extract Data Dictionary
- *Allegro Platform Properties Reference*.

# The extracta Output File

`The extracta` command generates a text file that contains two header records followed by one data record per extracted element. Each record in the file starts with a record-type character.

- The first record starts with the letter A
- The second record starts with the letter J
- Subsequent records start with the letter S

The reason for these record-type characters is that many processes require you to sort the file before further processing. Assuming you use a simple ascending, whole-record sort, the A–record always sorts to the leading record and the J–record next, followed by all the S–records. These characters also allow for the future definition of other types of data records.

The standard separator character between every field in all records is the exclamation point (!). Three types of records exist: A, J, and S records.

## A Records

The first record starts with the letter A. This record contains the labels for each data field `extracta` writes in each of the data records of the file. The labels are the same as the data field names in the `extracta` command file. (Some exceptions include cases where the view writes a fixed set of data fields and their corresponding names to the A record.)

## J Records

The second record starts with the letter J. This sample record contains the following global data about the Allegro design as described in the table.

```
A!REFDES!COMP_DEVICE_TYPE!

J!name.brd!Mon May 6 15:21:43 2003!0!0!11000!8500!1!mils!MIKEY!35.2876 mil!4!OUT OF
DATE!
```

```
S!U3!7400!

S!U5!CAPA!
```

| Field Position | Description | Syntax of Example |
|---|---|---|
| 1 | Design name | name.brd |
| 2 | Extracta execution date | Mon May 6 15:21:43 2003 |
| 3 | Drawing extents of the right x coordinate: left X | 0 |
| 4 | Drawing extents of the left x coordinate: lower Y | 0 |
| 5 | Drawing extents of the upper y coordinate: right X | 11000 |
| 6 | Drawing extents of the lower y coordinate: upper Y | 8500 |
| 7 | Database accuracy (the user units of the minimum incremental distance in the drawing) | 1 |
| 8 | Units of measurement used in the database as a text string (mils, millimeters, etc.) | mils |
| 9 | Value, if any, of the BOARD_SCHEMATIC_NAME property | MIKEY |
| 10 | Value, if any, of the BOARD_THICKNESS property | 35.2876 mil |
| 11 | Number of etch/wire layers in the design | 4 |
| 12 | BOARD_DRC_STATUS field | OUT OF DATE |

## S Records

Subsequent records in the file start with the letter S and contain the data values for the fields named in the A-record, in the same order as it gives the names.

## Sample Output Files

The examples below illustrate output generated by three command files, extracting data from the same design database. Note the use of the `-t` switch in the first example.

# Example 1: Using the GEOMETRY View

The `extracta` command file `brd_outline_view.txt`

```
#   View name:

GEOMETRY

#   Select:

CLASS = ’BOARD GEOMETRY’

SUBCLASS = OUTLINE

#  Requested datafields:

CLASS

SUBCLASS

RECORD_TAG

GRAPHIC_DATA_NAME

GRAPHIC_DATA_NUMBER

GRAPHIC_DATA_1

GRAPHIC_DATA_2

GRAPHIC_DATA_3

GRAPHIC_DATA_4

GRAPHIC_DATA_5

GRAPHIC_DATA_6

GRAPHIC_DATA_7

GRAPHIC_DATA_8

GRAPHIC_DATA_9

END
```

when used in the following command:

```
extracta -t test_board brd_outline_view outline
```

creates a file called `outline.txt`:

```
A!CLASS!SUBCLASS!RECORD_TAG!GRAPHIC_DATA_NAME!GRAPHIC_DATA_NUMBER

!GRAPHIC_DATA_1!GRAPHIC_DATA_2!GRAPHIC_DATA_3

!GRAPHIC_DATA_4!GRAPHIC_DATA_5!GRAPHIC_DATA_6

!GRAPHIC_DATA_7!GRAPHIC_DATA_8!GRAPHIC_DATA_9!

J!/usr2/bud/test_board.brd

!Mon May 6 15:34:52 1991!-1500!-1500!4000!4500!1!mils!!!

S!BOARD GEOMETRY!OUTLINE!7 1!LINE!257!-300!3200!3700!3200!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 2!LINE!257!-300!1200!-300!3200!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 3!LINE!257!-200!1200!-300!1200!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 4!LINE!257!-200!1100!-200!1200!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 5!LINE!257!3600!1100!-200!1100!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 6!LINE!257!3600!1200!3600!1100!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 7!LINE!257!3700!1200!3600!1200!0!!!!!

S!BOARD GEOMETRY!OUTLINE!7 8!LINE!257!3700!3200!3700!1200!0!!!!!S!BOARD
GEOMETRY!OUTLINE!8 1!TEXT!260!300!3000!0.000!NO!LEFT

!8 0 100 75 0.000 25 125 0!BOARD GEOMETRY   OUTLINE:!!!
```

⚠ The A-, J-, and the last S-record are broken into several lines for clarity; `extracta` never breaks records into multiple lines.

This data can be used to create a numerical control program to drive a routing machine for cutting boards from raw stock. Note that the outline LINE records all have the same major element number (7) in their RECORD_TAG data field. `extracta` lists them this way whether the lines were added as separate elements or with a single `add_line` command (unless the `-t` option is used).

## Example 2: A Formatted Netlist Report

In this example, you can extract data about the pins and their net assignments in the design, then create a readable netlist report using the sort utility programs. (See SORT Data Fields for details.)

The `extracta` command file `net_view.txt`

```
#
#  View name:
#
LOGICAL_PIN
#
# Select only non-blank nets:
#
   NET_NAME != ''
#
# Datafields:
#
   NET_NAME_SORT
   REFDES_SORT
   PIN_NUMBER_SORT
   FUNC_DES_SORT
   PIN_NAME
   REFDES
   FUNC_DES
   PIN_NUMBER
   NET_NAME
# end of net_view.txt
END
```

when used in the following command:

```
extracta test_board net_view net_pin
```

creates a file called `net_pin.txt`:

```
A!NET_NAME_SORT!REFDES_SORT!PIN_NUMBER_SORT!FUNC_DES_SORT
!PIN_NAME!REFDES!FUNC_DES!PIN_NUMBER!NET_NAME!
```

```
J!/usr2/bud/test_board.brd!Mon May 6 09:22:02 1991
!-1500!-1500!4000!4500!1!mils!!!
S!TP 00000001!TP 00000001! 00000001!F 00000009!CON
PIN!TP1!F9!1!TP1!
S!TN- 00000006!R 00000001! 00000002!F 00000001!B!R1!F1!2!TN-6!
S!TN- 00000005!R 00000001! 00000001!F 00000001!A!R1!F1!1!TN-5!
S!TN- 00000008!R 00000002! 00000002!F 00000002!B!R2!F2!2!TN-8!
S!TN- 00000010!R 00000003! 00000002!F 00000003!B!R3!F3!2!TN-10!
S!TN- 00000012!R 00000004! 00000002!F 00000004!B!R4!F4!2!TN-12!
S!TN- 00000014!R 00000005! 00000002!F 00000005!B!R5!F5!2!TN-14!
S!TN- 00000013!R 00000005! 00000001!F 00000005!A!R5!F5!1!TN-13!
S!TN-00000008!J00000001! 00000022!F 00000031!CON
PIN!J1!F31!22!TN-8!
S!TN-00000014!J00000001!00000021!F00000030!CON
PIN!J1!F30!21!TN-14!
S!TN-00000006!J00000001! 00000020!F 00000029!CON
PIN!J1!F29!20!TN-6!
S!TN-00000012!J00000001!00000016!F 00000025!CON
PIN!J1!F25!16!TN-12!
S!TN-00000010!J00000001!00000023!F 00000020!CON
PIN!J1!F20!23!TN-10!
S!TN-00000022!J00000001! 00000008!F 00000017!CON
PIN!J1!F17!8!TN-22!
S!TN-00000022!J00000001! 00000007!F 00000016!CON
PIN!J1!F16!7!TN-22!
S!VCC!J 00000001! 00000001!F 00000014!CON-PIN!J1!F14!1!VCC!
S!VCC!J 00000001! 00000003!F 00000012!CON-PIN!J1!F12!3!VCC!
S!GND!U 00000003! 00000010!!!U3!!10!GND!
```

```
S!VCC!U 00000003! 00000020!!!U3!!20!VCC!
S!TP 00000001!U 00000003! 00000016!F 00000045!D!U3!F45!16!TP1!
S!TN- 00000004!U 00000003! 00000017!F 00000045!Q!U3!F45!17!TN
4!
S!TN- 00000005!U 00000003! 00000014!F 00000042!D!U3!F42!14!TN
5!
S!TN- 00000013!U 00000003! 00000015!F 00000042!Q!U3!F42!15!TN
13!
S!GND!U 00000001! 00000007!!!U1!!7!GND!
S!VCC!U 00000001! 00000014!!!U1!!14!VCC!
S!TP 00000001!U 00000001! 00000010!F 00000036!B!U1!F36!10!TP1!
S!TN- 00000028!U 00000001! 00000002!F 00000035!B!U1!F35!2!TN
28!
S!TN- 00000028!U 00000001! 00000004!F 00000034!A!U1!F34!4!TN
28!
S!TN- 00000028!U 00000001! 00000005!F 00000034!B!U1!F34!5!TN-28!
```

Note that the A– and J– records are broken into several lines for clarity; `extracta` never breaks records into multiple lines.

# Example 3: Etch Line Coordinates; GEOMETRY View

The `extracta` command file `etch_view.txt`

```
#
#    Allegro X PCB Editor Data Extract Sample
#    Extract Command File
#    Extracts all ETCH elements
#
#   View Name:
GEOMETRY
```

```
#
#   Select:
   CLASS = ETCH#
#   Data fields to be written out:
   CLASS
   SUBCLASS
   NET_N
AME_SORT
   NET_NAME
   RECORD_TAG
   GRAPHIC_DATA_NAME
   GRAPHIC_DATA_NUMBER
   GRAPHIC_DATA_1
   GRAPHIC_DATA_2
   GRAPHIC_DATA_3
   GRAPHIC_DATA_4
   GRAPHIC_DATA_5
   GRAPHIC_DATA_6
   GRAPHIC_DATA_7
   GRAPHIC_DATA_8
   GRAPHIC_DATA_9
   GRAPHIC_DATA_10
#   End of Command File
END
```

when used in the following command:

```
extracta test_board etch_view etch_dat
```

creates a file called `etch_dat.txt`:

```
A!CLASS!SUBCLASS!NET_NAME_SORT!NET_NAME!RECORD_TAG
```

```
!GRAPHIC_DATA_NAME!GRAPHIC_DATA_NUMBER    !GRAPHIC_DATA_1!GRAPHIC_DATA_2!GRAPHIC_DAT
A_3!GRAPHIC_DATA_4

!GRAPHIC_DATA_5!GRAPHIC_DATA_6!GRAPHIC_DATA_7!GRAPHIC_DATA_8

!GRAPHIC_DATA_9!

J!/usr2/bud/test_board.brd!Mon May 6 10:12:27 1991

!-1500!-1500!4000!4500!1!mils!!!

S!ETCH!TOP!TP 00000001!TP1!1

1!LINE!257!1500!2300!1450!2250!12!

S!ETCH!TOP!TP 00000001!TP1!1

2!LINE!257!1450!2250!1175!2250!12!

S!ETCH!TOP!TP 00000001!TP1!1

3!LINE!257!1175!2250!1125!2300!12!

S!ETCH!TOP!TP 00000001!TP1!1 4!LINE!257!1125!2300!900!2300!12!

S!ETCH!TOP!TP 00000001!TP1!2 1!LINE!257!600!2300!900!2300!12!

S!ETCH!BOTTOM!TN- 00000028!TN-28!3

1!LINE!257!1200!2600!1250!2550!12!

S!ETCH!BOTTOM!TN- 00000028!TN-28!3

2!LINE!257!1250!2550!1250!2450!12!

S!ETCH!BOTTOM!TN- 00000028!TN-28!3

3!LINE!257!1250!2450!1200!2400!12!

S!ETCH!BOTTOM!TN- 00000028!TN-28!4

1!LINE!257!1200!2300!1200!2400!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!5

1!LINE!257!1125!2350!975!2350!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!5

2!LINE!257!975!2350!950!2375!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!53!LINE!257!950!2375!850!2375!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!5
```

```
4!LINE!257!850!2375!825!2350!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!5

5!LINE!257!825!2350!700!2350!12!

S!ETCH!BOTTOM!TN- 00000013!TN-13!5

6!LINE!257!700!2350!675!2325!12!S!ETCH!BOTTOM!TN-00000013!TN-13!5 7!LINE!257!675!

2325!675!2200!12!

S!ETCH!TOP!TN- 00000013!TN-13!6 1!LINE!257!600!2200!675!2200!12!50
```

Note that the A– and J–records are broken into several lines for clarity; `extracta` never breaks records into multiple lines.

This data can be used, for example, as input to a program to plot a drawing on a plotter not supported by the layout editor.

# APD: Die-stack Data Extraction

The extraction of die-stack data is available only through a pseudo view with its output written in a fixed format to the output text file. The name of the die stack command file included with the system is `die_stack_view.txt`.

Using the extracta command for extracting die-stack data is the same as extracting other views of a database. You need a design to process, a command file describing the required view, and a single output file.

## Die-Stack Command File

Because the die-stack data extraction is a pseudo view, the command file does not contain individual entries for each data field but contains a single definition that represents a fixed output of the view data. In the case of the die-stack data view, the definition is DIE_STACK_DEF. The content of the die-stack data command file, `die_stack_view.txt`, is included below:

```
# die_stack_view.txt

#

# edit history:

#   mxl 07/30/2007 Initial version.

#

# note:

#   no fields are allowed in this "view".

#   die-stack data is only available in .sip databases.

DIE_STACK_DEF
```

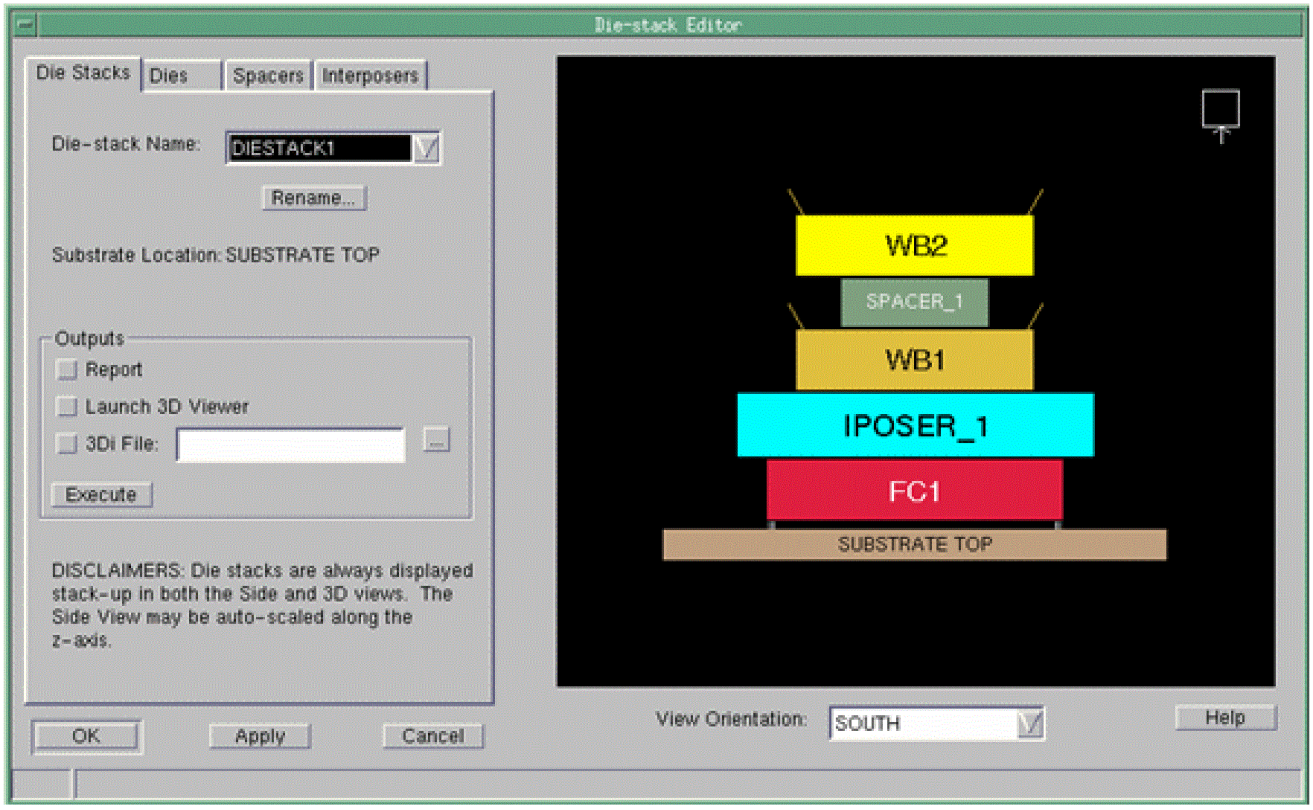
When the tool encounters the keyword DIE_STACK_DEF in the extracta command file, it initiates an internally-defined fixed format extraction of the die-stack data contained in the specified SIP database and writes it out to the specified output file.

⚠ You cannot include any other field definitions in a command file containing the DIE_STACK_DEF keyword.

## Die-Stack Example

DIESTACK1 contains five members at the following stack positions (integer-numbered from the substrate surface outward):

1. FC1, a flip-chip die
2. IPOSER_1, an interposer
3. WB1, a wire bond die
4. SPACER_1, a spacer
5. WB2, a wirebond die




Assuming that this is the only die stack in the design, the `extracta` output of the die-stack view for this die stack is shown below. The first S record represents the die stack itself (no MEMBER_NAME field value).

```
A!DSA_DIE_STACK_NAME!DSA_SUBSTRATE_SURFACE!DSA_MEMBER_NAME!DSA_PART_NUMBER!DSA_MEMBER_T
YPE!DSA_STACK_POSITION!DSA_MEMBER_LAYER!DSA_DIE_TYPE!DSA_DIE_ATTACH_TYPE!DSA_DIE_ORIENT
ATION!DSA_ROTATION!DSA_ORIGIN_X!DSA_ORIGIN_Y!DSA_EXTENTS_X1!DSA_EXTENTS_Y1!DSA_EXTENTS_
X2!DSA_EXTENTS_Y2!DSA_STACK_HEIGHT_MIN!DSA_STACK_HEIGHT_MAX!DSA_CONDUCTOR_MATERIAL!DSA_
CONDUCTOR_THICKNESS!DSA_DIELECTRIC_MATERIAL!DSA_DIELECTRIC_THICKNESS!DSA_BUMP_PACKAGE_D
IAM!DSA_BUMP_DIE_DIAM!DSA_BUMP_MAX_DIAM!DSA_BUMP_HEIGHT!DSA_BUMP_ECOND!
```

```
J!/home/mxl/icpkg/dsedit/extracta/testcases/test_dse_basic.sip!Wed Aug 22 14:46:17
2007!-50000.00!-50000.00!50000.00!50000.00!0.01!microns!!78.000000 mil!14!OUT OF DATE!

S!DIESTACK1!SUBSTRATE TOP!!!!!!!!!0.000!!!-6000.00!-
6000.00!6000.00!6000.00!0.00!496.00!!!!!!!!!!

S!DIESTACK1!SUBSTRATE TOP!FC1!!DIE!1!TOP_COND!STANDARD!FLIP-CHIP!CHIP
DOWN!0.000!0.00!0.00!-5000.00!-
5000.00!5000.00!5000.00!0.00!110.00!!!!!86.00!86.00!92.00!10.00!6897 MHO/MM!

S!DIESTACK1!SUBSTRATE TOP!IPOSER_1!2ZP-IPSR-
FR4CL7!INTERPOSER!2!IP1!!!!0.000!0.00!0.00!-6000.00!-
6000.00!6000.00!6000.00!110.00!216.00!COPPER!6.00!SILICON!100.00!!!!!!

S!DIESTACK1!SUBSTRATE TOP!WB1!!DIE!3!WB1!STANDARD!WIREBOND!CHIP UP!0.000!0.00!0.00!-
4000.00!-4000.00!4000.00!4000.00!216.00!316.00!!!!!!!!!!

S!DIESTACK1!SUBSTRATE TOP!SPACER_1!MY_SPACER_1!SPACER!4!SP2!!!!0.000!0.00!0.00!-
2500.00!-2500.00!2500.00!2500.00!316.00!396.00!!!PHENOLIC!80.00!!!!!!

S!DIESTACK1!SUBSTRATE TOP!WB2!!DIE!5!WB2!STANDARD!WIREBOND!CHIP UP!0.000!0.00!0.00!-
4000.00!-4000.00!4000.00!4000.00!396.00!496.00!!!!!!!!!!
```

## Die-Stack Data Output File

The output for a die-stack data extraction is written by the `extracta` command in a format internally-defined by a fixed set of data fields. You have no control over the output of the data for a die-stack extraction.

A block of data is written out for each die stack in the specified design. The first record of each block contains data specific to the overall die stack being processed. The remaining records each contain the data for one member of that die stack. The member records are output as they exist in the die stack, ordered from the substrate surface outward.

You may need to process the contents of the output file to reformat the data into a form more suited to your needs.

The following data fields are predefined for die-stack data output in the following order:

| Output Field Name | Description |
|---|---|
| DSA_DIE_STACK_NAME | Specifies the current die stack name, repeated for each member. |

| | |
|---|---|
| DSA_SUBSTRATE_SURFACE | Specifies the surface on which the die-stack is located (one of SUBSTRATE TOP or SUBSTRATE BOTTOM). |
| DSA_MEMBER_NAME | Specifies the refdes of the current member. |
| DSA_PART_NUMBER | Represents the part number that you assigned to the spacer or interposer. |
| DSA_MEMBER_TYPE | Specifies either a DIE, SPACER, or INTERPOSER. |
| DSA_STACK_POSITION | Represents the integer position of the member in the stack. |
| DSA_MEMBER_LAYER | Represents the Etch subclass of the member etch objects (pads, clines, or shapes for interposers). |
| DSA_DIE_TYPE | Specifies the type of die: STANDARD or CO-DESIGN. |
| DSA_DIE_ATTACH_TYPE | Specifies the attach type of the die: WIREBOND or FLIPCHIP. |
| DSA_DIE_ORIENTATION | Specifies either CHIP-UP or CHIP-DOWN (relative to top substrate surface) |
| DSA_ROTATION | Represents the stack/member rotation: 0, 90, 180, or 270 degrees. |
| DSA_ORIGIN_X | Indicates the X-coordinate of stack or member origin. |
| DSA_ORIGIN_Y | Indicates the Y-coordinate of stack or member origin. |
| DSA_EXTENTS_X1 | Indicates the lower-left x-coordinate of stack or member extents. |
| DSA_EXTENTS_Y1 | Indicates the lower-left y-coordinate of stack or member extents. |
| DSA_EXTENTS_X2 | Indicates the upper-right x-coordinate of stack or member extents. |
| DSA_EXTENTS_Y2 | Indicates the upper-right y-coordinate of stack or member extents. |
| DSA_STACK_HEIGHT_MIN | Indicates the minimum height of stack or member above the substrate surface. |

| | |
|---|---|
| DSA_STACK_HEIGHT_MAX | Indicates the maximum height of stack member above the substrate surface. |
| DSA_CONDUCTOR_MATERIAL | Specifies the conductor material of the interposer. |
| DSA_CONDUCTOR_THICKNESS | Specifies the conductor thickness of the interposer. |
| DSA_DIELECTRIC_MATERIAL | Specifies the dielectric material of the spacer or interposer. |
| DSA_DIELECTRIC_THICKNESS | Specifies the dielectric thickness of the spacer or interposer. |
| DSA_BUMP_PACKAGE_DIAM | Specifies the diameter of the flip-chip die bump at the package surface. |
| DSA_BUMP_DIE_DIAM | Specifies the diameter of the flip-chip die bump at die surface. |
| DSA_BUMP_MAX_DIAM | Specifies the maximum diameter of the flip-chip bump. |
| DSA_BUMP_HEIGHT | Specifies the bump height. |
| DSA_BUMP_ECOND | Specifies the electrical conductivity of the bump. |

7

# Generating Test Coupons

Allegro X PCB Editor features an automatic test coupon generator. Manufacturers use test coupons to measure the quality and accuracy of the Printed Circuit Board (PCB)/Multi-Chip Module (MCM) fabrication.

⚠ Although Allegro X Advanced Package Designer has this functionality, it is rarely used in the products.

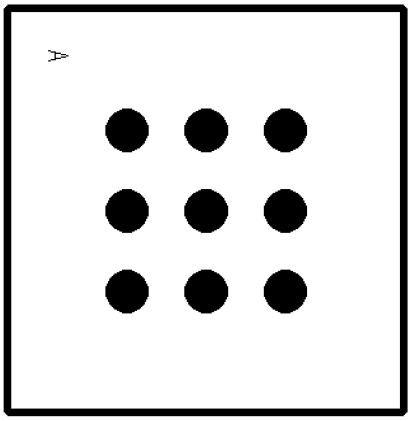
You can generate the following coupons per ANSI/IPC-D-275 standards:

- Coupon A to evaluate solderability of plated through holes.
- Coupon B to evaluate plating thickness, thermal stress and Type 1 bond strength.
- Coupon C to evaluate plating adhesion and solderability.
- Coupon D to evaluate interconnection resistance and continuity.
- Coupon E to evaluate surface resistance, bulk resistance, and material cleanliness after exposure to cyclic temperature and humidity.
- Coupon F to evaluate drill holes for break-out, etch back, or hole clean.
- Coupon G to evaluate solder resist adhesion.

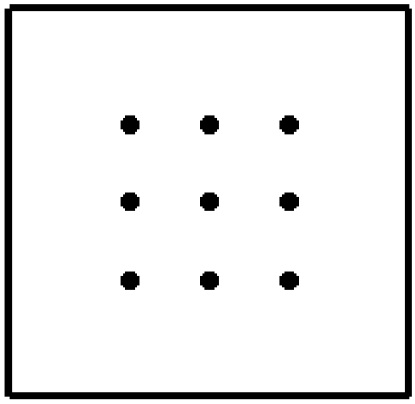
Examples of each coupon type follow.

# Coupon Examples

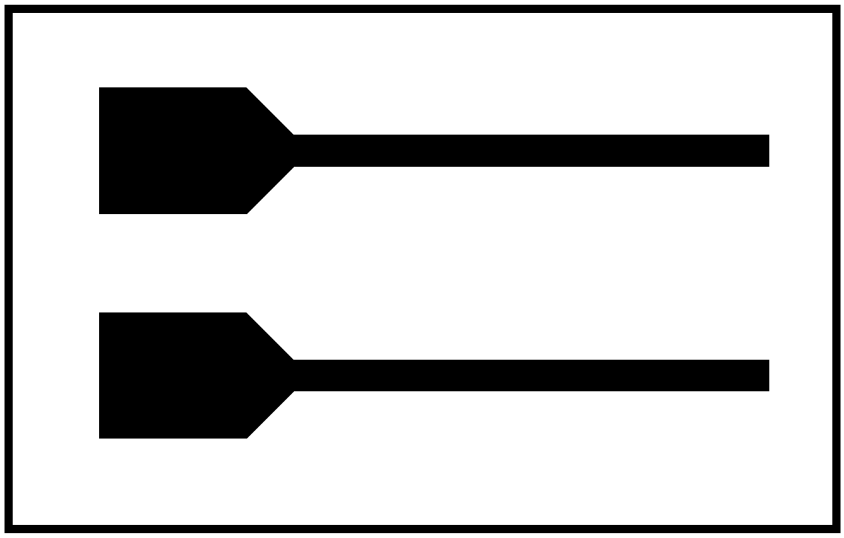
### Coupon A



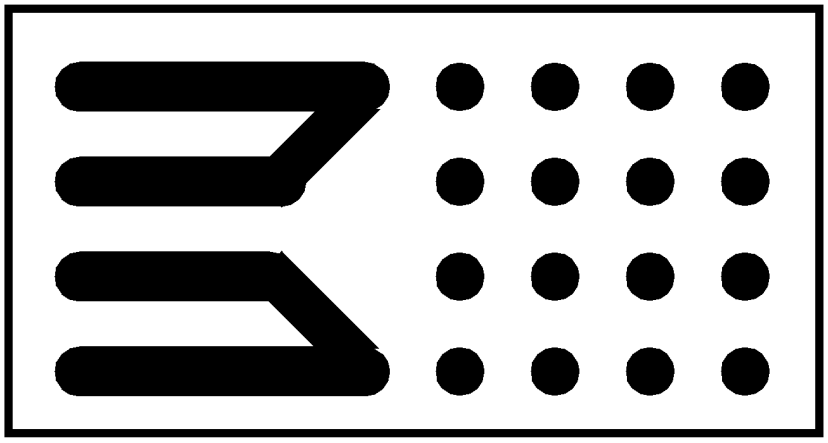
### Coupon B



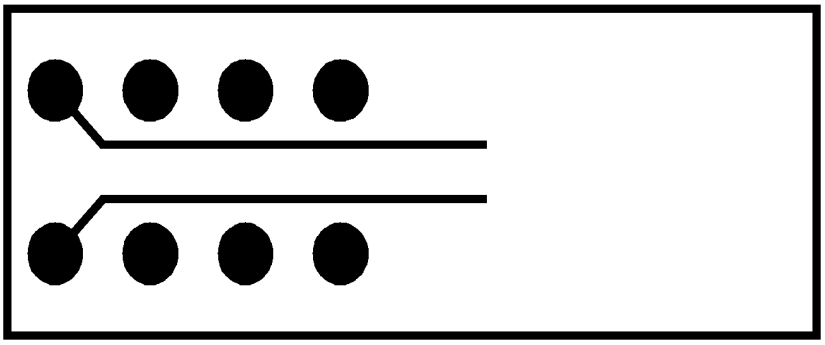
### Coupon C



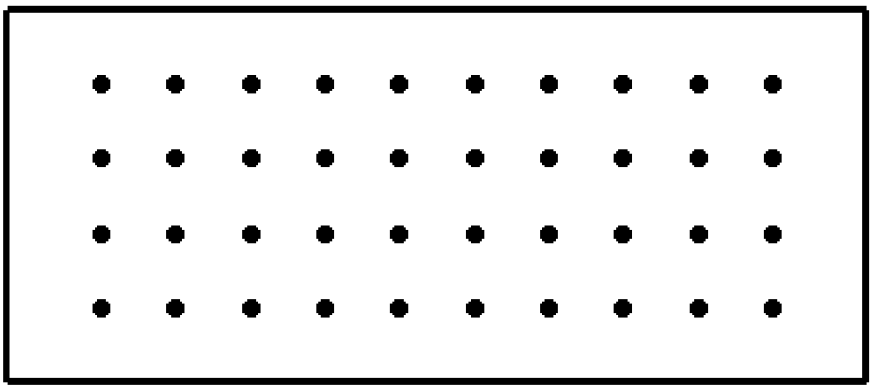
## Coupon D



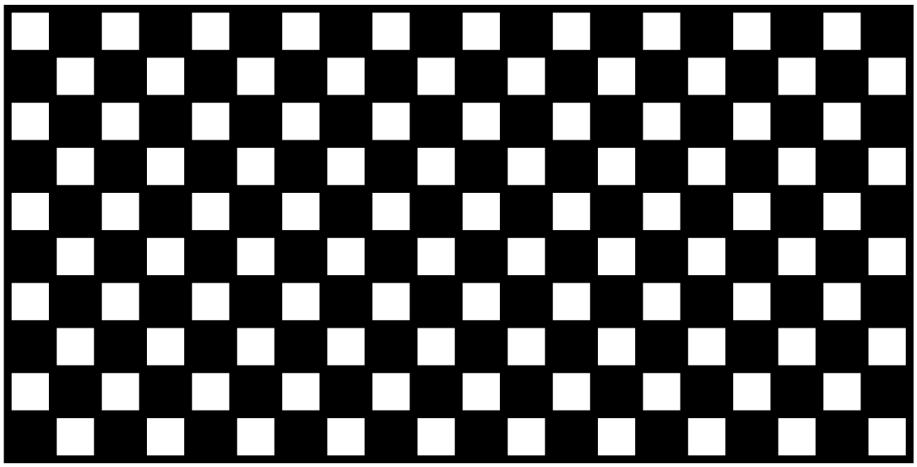
## Coupon E



## Coupon F



### Coupon G



## Related Topics

- *create coupons*

# Appendix A: Extract Data Dictionary

This section contains descriptions of data fields used by the layout editor views and the `extracta` command. Data fields that identify database views are listed and described in alphabetical order.

The Data Dictionary includes all data fields that you can extract and their available views. Legal views are:

| | |
|---|---|
| A = COMPONENT | G = GEOMETRY |
| B = COMPONENT_PIN | H = FULL_GEOMETRY |
| C = FUNCTION | I = SYMBOL |
| D = LOGICAL_PIN | J = BOARD |
| E = NET | K = CONNECTIVITY |
| F = COMPOSITE_PAD | L = LAYER |
| M = PAD_DEF | |

This topic discusses the following:



# Data Fields and Legal Views

This section provides a short description of each data field. See "Data Field Descriptions" for a comprehensive description.

| Data Field | Description | Legal Views |
|---|---|---|
| APADFLASH | anti-pad flash name | ABCDEFGHIJKL |
| APADHGHT | anti-pad height | ABCDEFGHIJKL |
| APADSHAPE1 | anti-pad geometry | ABCDEFGHIJKL |
| APADSHAPENAME | anti-pad shape name | ABCDEFGHIJKL |
| APADWIDTH | anti-pad width | ABCDEFGHIJKL |

| | | |
|---|---|---|
| APADXOFF | anti-pad x offset | ABCDEFGHIJKL |
| APADYOFF | anti-pad y offset | ABCDEFGHIJKL |
| BOARD_ACCURACY | accuracy of unit measurements | ABCDEFGHIJKL |
| BOARD_DRC_STATUS | (UP_TO_DATE/OUT_OF_DATE) | ABCDEFGHIJKL |
| BOARD_EXTENTS_X1 | board extents low left-X | ABCDEFGHIJKL |
| BACKDRILL_TOP_LAYER | layer to which to backdrill from the top side | B, D, F, H |
| BACKDRILL_BOTTOM_LAYER | layer to which to backdrill from the bottom side | B, D, F, H |
| BOARD_EXTENTS_Y1 | board extents low left-Y | ABCDEFGHIJKL |
| BOARD_EXTENTS_X2 | board extents up right-X | ABCDEFGHIJKL |
| BOARD_EXTENTS_Y2 | board extents up right-Y | ABCDEFGHIJKL |
| BOARD_LAYERS | etch layers in the board | ABCDEFGHIJKL |
| BOARD_NAME | name of the drawing | ABCDEFGHIJKL |
| BOARD_UNITS | units used in the drawing | ABCDEFGHIJKL |
| CLASS | class of entity | .B.C..FGH..K. |
| COMP_CLASS | component class (IC, IO, etc.) | ABCD.FG.I.K |
| COMP_DEVICE_TYPE | component device type | ABCD.FG.I.K |
| COMP_MAX_POWER_DISS_<br>DEVICE_INSTNCE | maximum power dissipation<br>maximum power diss - from device<br>maximum power diss - from inst only | ABCD.FGHI.K<br>ABCD.FGHI.K<br>ABCD.FGHI.K |
| COMP_PACKAGE | component package name | ABCD.FG.I.K |
| COMP_PLACEMENT_LAYER | etch layer name | A |
| COMP_ZONE_NAME | component zone name | A |
| DRILL_FIGURE_CHAR | drill hole figure character | .B.D.F.H. |

| | | |
|---|---|---|
| DRILL_FIGURE_HEIGHT | drill hole figure height | .B.D.F.H. |
| DRILL_FIGURE_ROTATION | drill hole figure rotation | .B.D.F.H. |
| DRILL_FIGURE_SHAPE | drill hole figure shape | .B.D.F.H. |
| DRILL_FIGURE_WIDTH | drill hole figure width | .B.D.F.H.. |
| DRILL_HOLE_NAME | drill hole name | .B.D.F.H. |
| DRILL_HOLE_NAME2 | slot minor dimension | .B.D.F.H. |
| DRILL_HOLE_NEGTOL | drill hole negative tolerance | .B.D.F.H |
| DRILL_HOLE_PLATING | drill hole plating information | .B.D.F.H. |
| DRILL_HOLE_POSTOL | drill hole positive tolerance | .B.D.F.H |
| DRILL_HOLE_X | drill hole x–coordinate | .B.D.F.H. |
| DRILL_HOLE_Y | drill hole y–coordinate | .B.D.F.H. |
| DRILL_ARRAY_ROWS | drill hole array row coordinates (plural vias) | .B.D.F.H. |
| DRILL_ARRAY_COLUMNS | drill hole array column coordinates (plural vias) | .B.D.F.H. |
| DRILL_ARRAY_CLEARANCE | drill hole clearances in arrays (plural vias) | .B.D.F.H. |
| DRILL_ARRAY_LOCATIONS | drill hole locations in format X1, Y1, X2, Y2... (plural vias) | .B.D.F.H. |
| EMBEDDED_LAYER | embedded layer name | A.....I |
| EMBEDDED_LAYER_SORT | sort embedded layers | A.....I |
| EMBEDDED_STATUS | embedded status | A.....I |
| EMBEDDED_ATTACH | embedded attach method | A.....I |
| END_LAYER_NAME | end layer (subclass) name | .B.D.F |
| END_LAYER_NUMBER | end layer number | .B.D.F |

| | | |
|---|---|---|
| FIXFLAG | internal layer fixed or optional (f = fixed, o = optional) | ABCDEFGHIJKL |
| FUNC_DES | function designator | .BCD.F |
| FUNC_DES_SORT | function designator—for sort | .BCD.F |
| FUNC_SLOT_NAME | function slot name | .BCD.F |
| FUNC_TYPE | function type | .BCD.F |
| GRAPHIC_DATA_NAME | graphic element name | .B.D.FGH...K |
| GRAPHIC_DATA_NUMBER | graphic element number | .B.D.FGH...K |
| GRAPHIC_DATA_1 | graphic data field 1 | .B.D.FGH...K |
| GRAPHIC_DATA_2 | graphic data field 2 | .B.D.FGH...K |
| GRAPHIC_DATA_3 | graphic data field 3 | .B.D.FGH...K |
| GRAPHIC_DATA_4 | graphic data field 4 | .B.D.FGH...K |
| GRAPHIC_DATA_5 | graphic data field 5 | .B.D.FGH...K |
| GRAPHIC_DATA_6 | graphic data field 6 | .B.D.FGH...K |
| GRAPHIC_DATA_7 | graphic data field 7 | .B.D.FGH...K |
| GRAPHIC_DATA_8 | graphic data field 8 | .B.D.FGH...K |
| GRAPHIC_DATA_9 | graphic data field 9 | .B.D.FGH...K |
| GRAPHIC_DATA_10 | graphic data field 10 | .B.D.FGH...K |
| LAYER | layer name | .....L |
| LAYER_ARTWORK | POSITIVE, NEGATIVE, or blank | .....L |
| LAYER_CONDUCTOR | specifies YES/NO for conductor layer | .....L |
| LAYER_DIELECTRIC_CONSTANT | dielectric constant | .....L |
| LAYER_ELECTRICAL_ CONDUCTIVITY | electrical conductivity | .....L |

| | | |
|---|---|---|
| LAYER_MATERIAL | specifies material type | .....L |
| LAYER_LOSS_TANGENT | represents energy lost | .....L |
| LAYER_SORT | sorts layer data | .....L |
| LAYER_SUBCLASS | subclass name | .....L |
| LAYER_SHIELD_LAYER | YES indicates a shield layer | .....L |
| LAYER_THERMAL_CONDUCTIVITY | thermal conductivity | .....L |
| LAYER_THICKNESS | layer thickness | .....L |
| LAYER_TYPE | basic layer type in cross-section | .....L |
| LAYER_USE | EMBEDDED_PLANE | .....L |
| NET_CAPACITANCE | capacitance of total net | .B.DEFGH..K |
| NET_ETCH_LENGTH | total etch (only) length on a net | .B.DEFGH..K |
| NET_ETCH_WIDTH_AVERAGE | average width of net | .B.DEFGH..K |
| NET_IMPEDANCE_AVERAGE | average impedance of net | .B.DEFGH..K |
| NET_IMPEDANCE_MAXIMUM | maximum impedance of segment | .B.DEFGH..K |
| NET_IMPEDANCE_MINIMUM | minimum impedance of segment | .B.DEFGH..K |
| NET_INDUCTANCE | inductance of total net | .B.DEFGH..K |
| NET_MANHATTAN_LENGTH | total Manhattan connect length | .B.DEFGH..K |
| NET_NAME | net name | .B.DEFGH..K |
| NET_NAME_SORT | net name - for sort | ....E.....K |
| NET_PATH_LENGTH | length of etch + Manhattan rats | .B.DEFGH..K |
| NET_PIN_COUNT | number of pins on a net | .E |
| NET_PROPAGATION_DELAY_ ACTUAL | calculated propagation delay on the net | .B.DEFGH..K |
| NET_RAT_CONNECT | sort field for connectivity | .B |

| | | |
|---|---|---|
| NET_RATT_COUNT | number of RatTs on a net | .E |
| NET_RAT_SCHEDULE | net ratsnest schedule | .B |
| NET_RESISTANCE | resistance of total net | .B.DEFGH..K |
| NET_STATUS | net (RAT) status | .B.DEFGH..K |
| NET_VIA_COUNT | number of vias in a net | .B.DEFGH..K |
| NODE_CONNECTS | number of connections | .B.D.FGH..K. |
| NODE_1_NUMBER | used for connectivity view | .....K. |
| NODE_2_NUMBER | used for connectivity view | .....K. |
| NODE_SORT | used for connectivity view | .....K. |
| PAD_FLASH | pad flash name | .B.D.F.H..K |
| PADHGHT | pad height | .....M. |
| PADSHAPE1 | pad geometry | .....M. |
| PAD_SHAPE_NAME | pad shape name | .B.D.F.H..K |
| PAD_STACK_ NAME | padstack name | .B.D.F.H..K |
| PAD_STACK_SOURCE_NAME | Blank for normal padstacks and the name of the "derived from" padstack for instance-edited padstacks | .B.D.F.H..K |
| PAD_TYPE | pad type | .B.D.F.H..K |
| PADWIDTH | pad width | .....M. |
| PADXOFF | X offset for pad | .....M. |
| PADYOFF | Y offset for pad | .....M. |
| PIN_COMMON_CODE | common pin if non-blank | .B.D.FG.H..K |
| PIN_EDITED | YES=pin has been instance edited | .B.D.FGH..K |

| | | |
|---|---|---|
| PIN_NAME | pin name | .B.D.FGH..K |
| PIN_NUMBER | pin number | .B.D.FGH..K |
| PIN_NUMBER_SORT | pin number—for sort | .B.D.FGH..K |
| PIN_ROTATION | rotation of the pin, in degrees, relative to its symbol. This is essentially the rotation of the pin as it exists in the symbol's drawing (`.dra`) database | .B.D.FGH..K |
| PIN_ROTATION_ABSOLUTE | rotation of the pin, in degrees, relative to the actual drawing. This takes into account, for instance, the rotation of the symbol as well as the rotation of the pin. | .B.D.FGH..K |
| PIN_SWAP_CODE | pin swap code | .B.D.FGH..K |
| PIN_TYPE | pin type | .B.D.FGH..K |
| PIN_X | pin absolute x–coordinate | .B.D.FGH..K |
| PIN_Y | pin absolute y–coordinate | .B.D.FGH..K |
| PLACEMENT_LAYER | etch layer name | A.......I |
| PLACEMENT_LAYER_SORT | sort placement layers | A.......I |
| RAT_CONNECTED | YES/NO for "ratsnest is connected" | .........K. |
| REC_NUMBER | record of what is being output for a particular padstack | ......GH..K |
| RECORD_TAG | record association tag | ......GH..K |
| REFDES | reference designator | ABCD.FG.I.K |
| REFDES_SORT | reference designator - for sort | ABCD.FGHI.K |
| SEG_CAPACITANCE | capacitance of an etch segment | ......GH..K |
| SEG_IMPEDANCE | impedance of an etch segment | ......GH..K |

| | | |
|---|---|---|
| SEG_INDUCTANCE | inductance of an etch segment | ......GH..K |
| SEG_PROPAGATION_DELAY | propagation delay | ......GH..K |
| SEG_RESISTANCE | resistance of an etch segment | ......GH..K |
| START_LAYER_NAME | start layer (subclass) name | .B.D.F.H..K |
| START_LAYER_NUMBER | start layer number | .B.D.F.H..K |
| SUBCLASS | subclass of entity | ......GH |
| SYM_BOX_X1 | symbol bounding box low left–X | ABCD.FGHI.K |
| SYM_BOX_Y1 | symbol bounding box low left–Y | ABCD.FGHI.K |
| SYM_BOX_X2 | symbol bounding box low left–X | ABCD.FGHI.K |
| SYM_BOX_Y2 | symbol bounding box low left–Y | ABCD.FGHI.K |
| SYM_CENTER_X | symbol body center x–coordinate | ABCD.FGHI.K |
| SYM_CENTER_Y | symbol body center y–coordinate | ABCD.FGHI.K |
| SYM_EXTENTS_X1 | symbol extents low left–X | ABCD.FGHI.K |
| SYM_EXTENTS_Y1 | symbol extents low left–X | ABCD.FGHI.K |
| SYM_EXTENTS_X2 | symbol extents up right–X | ABCD.FGHI.K |
| SYM_EXTENTS_Y2 | symbol extents up right–Y | ABCD.FGHI.K |
| SYM_HAS_PIN_EDIT | YES=symbol has an edited pin | ABCD.FGHI.K |
| SYM_LOGICALPIN_COUNT | number of logical pins on a symbol | AC.FI.J |
| SYM_MECHPIN_COUNT | number of mechanical pins on a symbol | AC.FI.J |
| SYM_MIRROR | symbol mirror flag | ABCD.FGHI.K |
| SYM_NAME | symbol name | AC.FI.J |
| SYM_PIN_COUNT | number of pins on a symbol | ABCD.FGHI.K |

| | | |
|---|---|---|
| SYM_ROTATE | symbol rotation angle | ABCD.FGHI.K |
| SYM_TYPE | MECHANICAL, PACKAGE, or FORMAT | ABCD.FGHI.K |
| SYM_X | symbol x–coordinate | ABCD.FGHI.K |
| SYM_Y | symbol y–coordinate | ABCD.FGHI.K |
| SYM_ZONE_NAME | stackup zone name | A.....I |
| SYM_ZONE_NAME_SORT | sort stackup zone name | A.....I |
| TEST_POINT | pin/via test point code | .B.FGH. . .K |
| TRELFLASH | thermal relief flash name | .B.D.F.H..K |
| TRELHGHT | thermal relief height | .B.D.F.H..K |
| TRELSHAPE1 | thermal relief geometry | .B.D.F.H..K |
| TRELSHAPENAME | thermal relief shape name | .B.D.F.H..K |
| TRELWIDTH | thermal relief width | .B.D.F.H..K |
| TRELXOFF | thermal relief x offset | .B.D.F.H..K |
| TRELYOFF | thermal relief y offset | .B.D.F.H..K |
| VIAFLAG | surface mount or through hole type | . . . . .FGH. . K |
| VIA_MIRROR | YES=via is mirrored | . . . . .FGH. . K |
| VIA_X | via x–coordinate | . . . . .FGH . .K |
| VIA_Y | via y–coordinate | . . . . .FGH. . K |

# Data Field Descriptions

### APADFLASH

The anti-pad flash name.

### APADHGHT

The anti-pad height.

### APADSHAPE1

The anti-pad geometry.

### APADSHAPENAME

The anti-pad shape name.

### APADWIDTH

The anti-pad width.

### APADXOFF

The anti-pad X offset.

### APADYOFF

The anti-pad Y offset.

### BACKDRILL_TOP_LAYER

The layer to which to backdrill from the top side. The top of the design is layer number 0, and the bottom is the total number of layers minus one. For example, a 4-layer design contains layers 0 through 3. If no backdrilling occurs for a pin or via from a side, the extract value is null.

### BACKDRILL_BOTTOM_LAYER

The layer to which to backdrill from the bottom side. The top of the design is layer number 0, and the bottom is the total number of layers minus one. For example, a 4-layer design contains layers 0 through 3. f no backdrilling occurs for a pin or via from a side, the extract value is null.

### BOARD_ACCURACY

The accuracy of the design—the user units of one database unit. For example, if the design is in inches with two decimal places, the accuracy will be .01 (inches). This field is written into all files as part of the J header record.

### BOARD_DRC_STATUS

The current DRC status of the design. This field is OUT_OF_DATE if a batch DRC needs to be run. The design can become OUT_OF_DATE by skipping DRC of shapes or by canceling out of a DRC. If no DRC is required the field is UP_TO_DATE.

### BOARD_EXTENTS_X1

### BOARD_EXTENTS_X2

### BOARD_EXTENTS_Y1

### BOARD_EXTENTS_Y2

The lower left (X1, Y1) and upper right (X2, Y2) of the extents of the design in user units. Note, this defines the minimum and maximum coordinates that can be used in the drawing; it does not indicate the actual extent of the design outline or any other geometry. This field is written into all files as part of the J header record.

### BOARD_NAME

The drawing name. This field is written into all files as part of the J header record.

### BOARD_LAYERS

The total number of etch/wire layers in the design. This is the total number of subclasses defined for class etch.

### BOARD_UNITS

The units used for specifying distance values and position coordinates. Units can be mils, inches, millimeters, centimeters, or microns. This field is written into all files as part of the J header record.

### CLASS

The class of the entity. Normally this field is for geometry, and indicates the type of geometry (ETCH, PACKAGE GEOMETRY, BOARD GEOMETRY, and so on). It is also used to indicate pins versus vias in the PAD and COMPOSITE PAD views (PIN for pins, VIA CLASS for vias). Legal values for the CLASS data field are listed below.

| | |
|---|---|
| ANALYSIS | BOARD GEOMETRY |
| COMPONENT VALUE | DEVICE TYPE |
| DRAWING FORMAT | DRC ERROR CLASS |
| ETCH | MANUFACTURING |
| PACKAGE GEOMETRY | PACKAGE KEEPIN |
| PACKAGE KEEPOUT | PIN |
| REF DES | ROUTE KEEPIN |
| ROUTE KEEPOUT | TOLERANCE |
| USER PART NUMBER | VIA CLASS |
| VIA KEEPOUT | |

To specify a legal value in a filter

- Type

  CLASS="BOARD GEOMETRY"

### COMP_CLASS

The class of the component: IC, IO, or DISCRETE.

### COMP_DEVICE_TYPE

The device type of the component. The device type is defined for the component during `Netlist In`.

### COMP_MAX_POWER_DISS

The maximum power dissipation in watts for a component as defined by the MAX_POWER_DISS property defined for refdes instances. If this property has not been set on a refdes, then the PACKAGEPROP MAX_POWER_DISS from the device file during `Netlist In`, if it exists, will be used. The maximum power dissipation in watts for the component.

### COMP_MAX_POWER_DISS_DEVICE

The _DEVICE suffix on the COMP_MAX_POWER_DISS data field retrieves data from only component definitions.

### COMP_MAX_POWER_ DISS_INSTANCE

The _INSTANCE suffix on the COMP_MAX_POWER_DISS data field retrieves data from only a specific reference designator.

### COMP_PACKAGE

The package to which the component is assigned. The package is defined for the component during `Netlist In`. This field exists for all components. After a component has been placed on the design, a symbol that has the same name as the package is used to represent it. For placed symbols, the symbol name is always the same as the package name.

### COMP_PLACEMENT_LAYER

The etch layer on which the component is placed.

### COMP_ZONE_NAME

The stackup zone name in which the component is placed.

### DRILL_FIGURE_CHAR

The character to display with the graphic figure that represents a drill hole.

### DRILL_FIGURE _HEIGHT

The height in user units of the graphic figure that displays to represent a drill hole. Normally used with DRILL_FIGURE_WIDTH and DRILL_FIGURE_ROTATION.

### DRILL_FIGURE _ROTATION

The angle of rotation of the graphic figure that displays to represent a drill hole. The value is in degrees with three decimal places of accuracy.

### DRILL_FIGURE _SHAPE

The graphic figure (shape) to represent a drill hole.

This is one of the standard graphic element names.

See GRAPHIC_DATA_NAME.

### DRILL_FIGURE _WIDTH

The width in user units of the graphic figure that displays to represent a drill hole. Normally used with DRILL_FIGURE_HEIGHT and DRILL_FIGURE_ROTATION.

### DRILL_HOLE_NAME

The name of the drill hole. This name is used to tell the NC Drill program what size drill bit to use. Note that DRILL_FIGURE_HEIGHT and DRILL_FIGURE_WIDTH are used to control the display of the drill hole. When used for slots, represents the slot's major dimension.

### DRILL_HOLE_NAME2

When used for slots, represents the slot's minor dimension.

### DRILL_HOLE_NEGTOL

When used for slots, represents the drill hole's negative tolerance.

### DRILL_HOLE_PLATING

A description of whether the drill hole is plated. Values are PLATED, NON_PLATED, or OPTIONAL.

### DRILL_HOLE_POSTOL

When used for slots, represents the drill hole's positive tolerance.

### DRILL_HOLE_X

### DRILL_HOLE_Y

The X or Y coordinate in user units of the drill hole.

### DRILL_ARRAY_ROWS

### DRILL_ARRAY_COLUMNS

The parameters of rows and columns of drill holes in plural vias, in user units.

### DRILL_ARRAY_CLEARANCE

The clearance between all drill holes in a plural via array.

**DRILL_ARRAY_LOCATIONS**

The location of all drill holes in a plural via array. The format is X1, Y1, X2, Y2...

**EMBEDDED_LAYER**

The internal embedded layer on which the symbol is placed.

**EMBEDDED_LAYER_SORT**

Sort internal embedded layers.

**EMBEDDED_STATUS**

The orientation of the component on the layer.The status of embedded component can be BODY_UP, BODY_DOWN, PROTRUDING_ALLOWED, and NOT_EMBEDDED.

**EMBEDDED_ATTACH**

The method used for connecting the components to the embedded layer. The embedded attach method can be DIRECT_ATTACH and INDIRECT_ATTACH.

**END_LAYER_NAME**

The layer name (subclass) closest to the BOTTOM of the design that has a pad for a particular COMPOSITE PAD. This field is normally used with START_LAYER_NAME. Normal through holes (pins or vias) have the START_LAYER_NAME as TOP and the END_LAYER_NAME as bottom. Surface mount devices typically have pads with START_LAYER_NAME and END_LAYER_NAME the same. Blind and buried vias have START_LAYER_NAME and/or END_LAYER_NAME as something other than TOP or BOTTOM.

**END_LAYER_NUMBER**

The field is used like END_LAYER_NAME, except that the value is the actual layer number rather than the subclass name. The top of the design is layer number 0, and the bottom is the total number of layers minus one. For example, a 4-layer design contains layers 0 through 3.

**FIXFLAG**

The internal layer fixed or optional (f = fixed, o = optional).

**FUNC_DES**

The function designator.

**FUNC_DES_SORT**

The function designator in a form for sorting. The sort form enables a standard text string sort, to sort the numeric part of the function designator in correct numeric order. This field should not be used for display purposes.

### FUNC_SLOT_NAME

The slot name of the component to which a function has been assigned.

### FUNC_TYPE

The function type of the function. The function type may be specified when the function is loaded during netlist process in g. If the component has only one type of function, the function type is the same as the device type.

### GRAPHIC_DATA_NAME

For standard figures, the GRAPHIC_DATA_NAME is the figure name used in the layout editor. For other figures, the name is indicated in the description of the *GRAPHIC_DATA_N* fields.

| | |
|---|---|
| FIGURE_NULL | 0 |
| FIGURE_CIRCLE | 2 |
| FIGURE_OCTAGON | 3 |
| FIGURE_CROSS | 4 |
| FIGURE_SQUARE | 5 |
| FIGURE_RECTANGLE | 6 |
| FIGURE_DIAMOND | 7 |
| FIGURE_OBLONG_X | 11 |
| FIGURE_OBLONG_Y | 12 |
| FIGURE_HEXAGON_X | 15 |
| FIGURE_HEXAGON_Y | 16 |
| FIGURE_TRIANGLE_1 | 18 |

### GRAPHIC_DATA _NUMBER

This field may used instead of the GRAPHIC_DATA_NAME to save disk space and to make the post processor's job easier.

### GRAPHIC_DATA_N fields

The GRAPHIC_DATA_N (where N is 1 to 10) fields are used differently depending on the graphic element as defined by the *GRAPHIC_DATA_NAME* (and GRAPHIC_DATA_NUMBER) field as follows:

```
GRAPHIC_DATA_NAME     figure name-standard figure

GRAPHIC_DATA_NUMBER     figure number (1 to 255)

GRAPHIC_DATA_1     x-coord of the figure

GRAPHIC_DATA_2     y-coord of the figure

GRAPHIC_DATA_3     width of the figure

GRAPHIC_DATA_4     height of the figure

GRAPHIC_DATA_NAME     ARC-arc

GRAPHIC_DATA_NUMBER     256

GRAPHIC_DATA_1     x-coord of start of arc

GRAPHIC_DATA_2     y-coord of start of arc

GRAPHIC_DATA_3     x-coord of end of arc

GRAPHIC_DATA_4     y-coord of end of arc

GRAPHIC_DATA_5     x-coord of center of arc's curve

GRAPHIC_DATA_6     y-coord of center of arc's curve

GRAPHIC_DATA_7     radius of arc's curve

GRAPHIC_DATA_8     width of the arc

GRAPHIC_DATA_9     direction (CLOCKWISE or COUNTER-CLOCKWISE)

GRAPHIC_DATA_10     type of arc (see LINE)

GRAPHIC_DATA_NAME     LINE-line (segment)

GRAPHIC_DATA_NUMBER     257

GRAPHIC_DATA_1     x-coord of one end of line

GRAPHIC_DATA_2     y-coord of one end of line

GRAPHIC_DATA_3     x-coord of other end of line

GRAPHIC_DATA_4     y-coord of other end of line

GRAPHIC_DATA_5     width of the line

GRAPHIC_DATA_10     type of line:

CONNECT connect line (cline)

NOTCONNECT not a connect line
```

```
SHAPE part of a shape outline

VOID part of a void within a shape

POLYGON part of an unfilled

shape

(including keepins/

keepouts and rotated rectangles

GRAPHIC_DATA_NAME     RECTANGLE-rectangle

GRAPHIC_DATA_NUMBER     259

GRAPHIC_DATA_1     x-coord of lower left corner

GRAPHIC_DATA_2     y-coord of lower left corner

GRAPHIC_DATA_3     x-coord of upper right corner

GRAPHIC_DATA_4     y-coord of upper right corner

GRAPHIC_DATA_5     fill code: 0=unfilled,

1=filled

GRAPHIC_DATA_NAME     TEXT - a line of text

GRAPHIC_DATA_NUMBER     260

GRAPHIC_DATA_1     x-coord of text line

GRAPHIC_DATA_2     y-coord of text line

GRAPHIC_DATA_3     angle of rotation of the text

GRAPHIC_DATA_4     mirror code of text = YES or NO

GRAPHIC_DATA_5     justification. LEFT, RIGHT, or

CENTER.

GRAPHIC_DATA_6     font data. Stored as 1 field with size,

font, height, width, slant,

character_spacing,line_spacing and

photoplot_width separated by spaces.

Values: 0=ANSI, 1=ISO, or 2=MICRO.

GRAPHIC_DATA_7     the text string.
```

```
GRAPHIC_DATA_NAME     T-ratsnest (CONNECTIVITY view only)

GRAPHIC_DATA_NUMBER     261

GRAPHIC_DATA_1     x-coord of the T

GRAPHIC_DATA_2     y-coord of the T

GRAPHIC_DATA_NAME     RATSNEST (CONNECTIVITY view only)

GRAPHIC_DATA_NUMBER     262

GRAPHIC_DATA_1     x-coord of one end of the ratsnestGRAPHIC_DATA_2 y-coord of one
end of the ratsnest

GRAPHIC_DATA_3     x-coord of the other end of the ratsnest
```

### LAYER

The layer name.

### LAYER_ARTWORK

The artwork type to use for the layer (subclass). The value is POSITIVE or NEGATIVE for a conductor layer, or blank if the layer is not a conductor.

### LAYER_CONDUCTOR

YES indicates that the layer is a conductor layer; NO indicates it is not.

### LAYER_DIELECTRIC _CONSTANT

The dielectric constant of the layer. This field is only meaningful for dielectric layers (LAYER_CONDUCTOR = NO).

### LAYER_ ELECTRICAL_CONDUCTIVITY

The electrical conductivity of the layer.

### LAYER_MATERIAL

The type of material for the layer. Common values are COPPER for conductor layers and FR–4, G–9, and so on, for dielectric layers.

### LAYER_LOSS_TANGENT

A number between–but not including–zero and one that represents the amount of energy lost to the dielectric from a signal travelling through an adjacent conductor.

### LAYER_SORT

This field is an integer that can be used to sort the records from a LAYER view into the correct sequence.

### LAYER_SUBCLASS

The subclass name for the layer.

### LAYER_SHIELD_LAYER

YES indicates a shield layer.

### LAYER_THERMAL_CONDUCTIVITY

The thermal conductivity of the layer.

### LAYER_THICKNESS

The thickness of the layer.

### LAYER_TYPE

The basic type of layer in the board/package stackup cross-section. Must be one of a set of the following pre-defined types:

- Conductor
- Dielectric
- Diestack
- Plane

### LAYER_USE

The use of the layer. Layers can be used as EMBEDDED PLANEs.

### NET_CAPACITANCE

The capacitance of a net.

> Presence of this field in your view results in a slower run.

### NET_ETCH_LENGTH

The total length of the etch on a net. See NET_PATH_LENGTH and NET_MANHATTAN_LENGTH.

### NET_ETCH_WIDTH _AVERAGE

The average trace width of a net.

### NET_IMPEDANCE _AVERAGE

The average impedance for a net.

### NET_IMPEDANCE _MAXIMUM

The maximum impedance of any segment of a net.

### NET_IMPEDANCE _MINIMUM

The minimum impedance of any segment of a net.

### NET_INDUCTANCE

The inductance of a net.

### NET_MANHATTAN _LENGTH

Total length of a net using the manhattan distance of all the pin–to–pin connections. See also NET_ETCH_LENGTH and NET_PATH_LENGTH.

### NET_NAME

The name of the net.

### NET_NAME_SORT

The net name in a form for sorting. The sort form enables a standard text string sort, to sort the numeric part of the net name in correct numeric order. This field should not be used for display purposes.

### NET_PATH_LENGTH

The length of the net using actual etch for connections already made and the manhattan distance for connections that have not yet been made.

### NET_PROPAGATION _DELAY_ACTUAL

The delay contributed by each cline on the net. Each cline is counted once.

> Presence of this field in your view results in a slower run.

### NET_RAT_CONNECT

A text field used to sort pins into an order useful for determining the physical connectivity of the pins in the net. This field is used internally by the layout editor.

### NET_RAT_SCHEDULE

A text field used to sort pins into the order indicated by the ratsnest records for the net. This field is most meaningful for nets that have been scheduled by the $SCHEDULE section in `netin`. Nets with the NO_RAT property are ratsnested if this field is requested.

The field has three parts, separated by spaces:

- The first is a subnet number.

- The second is the pin sequence number on that subnet.
  The first subnet is 1; the first element on a subnet is 1. Subsequent subnets are created when a pin is connected by a ratsnest to more than one other pin.
- The third part of the field is used only for the first pin of any subnet other than the first.
  It is the pin designator of the pin to which this pin is connected. In other words it is the parent of this subnet. Each pin other than the first pin on the subnet is assumed to be connected to the previous pin.

Consider the following:

U1.1-----U2.2------U3.3 | | U4.------U5.5

The Netlist In $SCHEDULE would have been:

```
signal_name; U1.1 U2.2 U3.3 ; U2.2 U4.4 U5.5
```

The NET_RAT_SCHEDULE field for the pins would be:

```
U1.1 - "1 1"

U2.2 - "1 2"

U3.3 - "1 3"

U4.4 - "2 1 U2.2"

U5.5 - "2 2"
```

Note that an equivalent schedule is:

```
signal_name; U1.1 U2.2 U4.4 U5.5 ; U2.2 U3.3
```

and NET_RAT_SCHEDULE:

```
U1.1 - "1 1"

U2.2 - "1 2"

U4.4 - "1 3"

U5.5 - "1 4"

U3.3 - "2 1 U2.2"
```

There is no guarantee which equivalent schedule will be returned.

The NET_RAT_SCHEDULE is only available when using the COMPONENT_PIN view. If it is requested, the pins are read from the database on a net by net basis, rather than on a component by component basis. This means that only pins on nets are extracted if this field is requested.

**NET_RESISTANCE**

The total resistance of a net.

### NET_STATUS

The (rat) status of the net. This field is: SCHEDULED if the net has been scheduled by Auto Schedule or `Netlist In`; NO_RAT if the NO_RAT property has been set; or REGULAR.

### NET_VIA_COUNT

The number of vias on a net.

### NODE_CONNECTS

This field indicates how many connections there are for particular objects. For connect lines, it will equal 0, 1, or 2, depending on how many ends connect to a T, PIN, or VIA. For Ts, PINS, or VIAS, it is the number of connect lines that connect to it. This field is blank for shapes.

### NODE_1_NUMBER

### NODE_2_NUMBER

### NODE_SORT

These fields are used in the connectivity view so that a follow-up program can construct the connectivity of a net. The nets are used to create records that represent either a node of the net or a connection. NODE_SORT is used to sort nodes (pins, vias, and Ts) before connections that reference them (ratsnests and standard etch geometry). The usual sequence of data fields would then be *NET_NAME_SORT* (or *NET_NAME*), *NODE_SORT*, *NODE_1*, *NODE_2*, and *RECORD_TAG*. The text file can be sorted using a standard left–to–right sort.

For nodes, *NODE_1* equals *NODE_2* and is the node number. For connections, *NODE_1* is the node number of one end of the connection and *NODE_2* is the node number of the other end of the connection. Shapes can be extracted with this view, but their *NODE_1* and *NODE_2* fields are blank. For ratsnests, in addition to the *NODE_1* and *NODE_2* fields, the *RAT_CONNECTED* field indicates whether there is an electrical connection between the two nodes.

### PAD_FLASH

The name of the flash to be used for this pad. The flash maps a pad to a specific aperture when creating artwork. This is typically done for non-standard pads such as thermal-relief pads or anti-pads.

### PADHGHT

The height of the pad.

### PADSHAPE1

The pad geometry.

### PAD_SHAPE_NAME

Describes the shape of the pad. When used in the FULL_GEOMETRY view, the geometry of the pad is indicated. On other views, this field describes the composite pad used for display in the interactive editor. If the shape is a standard geometry, this field is the same as the *GRAPHIC_ DATA_NAME* field (for example, FIG_CIRCLE or FIG_RECTANGLE). If the shape is a non-standard geometry, this field contains the name of the shape preceded by FIG_SHAPE and a space.

### PAD_STACK_NAME

The name of the padstack.

### PAD_STACK_SOURCE_NAME

Blank for normal padstacks and the name of the "derived from" padstack for instance-edited padstacks.

### PAD_TYPE

Describes the kind of pad as either regular, thermal, or anti–pad. In the FULL_GEOMETRY view, the type of the actual pad used contains ANTI, THERMAL, or REGULAR statements. In other views, it contains the value COMPOSITE to indicate that the pad refers to the composite pad.

### PADWIDTH

The width of the pad.

### PADXOFF

The X coordinate for the offset of the pad.

### PADYOFF

The Y coordinate for the offset of the pad.

### PIN_COMMON_CODE

A code to indicate whether a pin is common to (shared by) multiple functions in the device. If the pin is not common, the field is blank. If it is common and the view is LOGICAL_PIN, the value is a number increasing from 1. This makes it possible for post processors (such as netlist generators) to filter out the redundant occurrences of the same physical. If it is common and the view is other than LOGICAL_PIN (COMPONENT_ PIN, GEOMETRY, and so on), it is a number and can be the number of functions that share the pin. It should only be treated as blank or non-blank.

### PIN_EDITED

The value is YES if the pin has been instance edited. This occurs when the pin is either moved or the padstack has been edited.

### PIN_NAME

The name of the pin. If the actual internal pin name was generated by the layout editor, this field is

the same as the PIN_NUMBER. The layout editor generates pin names when there are no explicitly defined functions in the device file (in which case the internal name becomes TP-*nnn*, where *nnn* is the pin number).

**PIN_NUMBER**

The pin number. If a pin is part of a function that has not been assigned to a component, this field is blank.

**PIN_NUMBER_SORT**

The pin number in a form for sorting. The sort form enables a standard text string sort, to sort the numeric part of the pin number in correct numeric order. This field should not be used for display purposes. As a convenience, if the pin number is blank (unassigned functions), the pin name is put into the *pin_number_sort* field. The fact that it is a pin name and not a pin number can be obtained by the fact the *PIN_NUMBER* and/or *FUNC_SLOT_ NAME* fields are blank.

**PIN_ROTATION**

The degree of pin rotation relative to its symbol. This is the rotation of the pin as it exists in the symbol drawing (`.dra`) database.

**PIN_ROTATION_ABSOLUTE**

The degree of pin rotation relative to the actual drawing, accounting for the rotation of the symbol as well as the rotation of the pin.

**PIN_SWAP_CODE**

A number indicating whether a pin can be swapped with other pins of the same function. All pins of a function that have the same number can be swapped with each other. This field is blank when pins cannot be swapped. This field only represents the swap code as defined by the device file. Properties may be attached to functions that override the pin's ability to be swapped, but they do not affect this field.

**PIN_TYPE**

The type of pin, sometimes called pin use uses the PINUSE property when it has been defined. Values are IN, OUT, BI, TRI, OCA, OCL, UNSPEC, DISCRETE, NC, POWER, and GROUND.

**PIN_X**

**PIN_Y**

The absolute (relative to the design not the symbol) X and Y coordinate, in user units, of the location of the pin. Mirror and rotation calculations have been performed in calculating the coordinate. If the pin is part of an unplaced component (or unassigned function), the field is blank.

**PLACEMENT_LAYER**

The etch layer on which the symbol is placed.

### PLACEMENT_LAYER_SORT

Sort placement layers.

### PROPAGATION_DELAY_ACTUAL

Calculates and reports the delay of a net or a segment. Only legal in views that support nets and geometry.

> Presence of this field in your view results in a slower run.

### RAT_CONNECTED

This field is available in the CONNECTIVITY view for ratsnest connection records. YES means that the nodes (NODE_1_NUMBER and NODE_2_NUMBER) are connected. The route for the connection must be determined by analyzing the net topology. This may not be straightforward if shapes are involved. See also NODE 1_NUMBER, NODE_2_NUMBER, NODE_SORT, and NODE_CONNECTS.

### REC_NUMBER

The record of what is being output for a particular padstack. (For example: .00001 = TOP, 00002 - internal pad def, 00003 = BOTTOM, then top SM, bottom SM, top paste, bottom paste.)

### RECORD_TAG

A field to establish association for geometric elements. Certain geometry is considered to be a sub-element of a major geometric element. For example, text lines are sub-elements of a major element *text*; line and arc segments are sub-elements of major element *line*; and patches are a sub-element of a major element *shape*. Rectangles and the layout editor-defined graphic figures never have any sub-elements.

The purpose of RECORD_TAG is to preserve the association of the sub-elements to the major element during sorts of the text file.

RECORD_TAG consists of two parts separated by a space.

- The first is a number that indicates the major geometric element number.
  This number starts at 1 (one) and increases for each major element in the database.
- The second is the sub-element number, which starts at 1 (one) for the first sub-element of each major element and then increases by one for each subsequent sub-element.

The *line* major element is handled in two ways. Line or arc segments that are attached on nets (generally ETCH) or symbols (generally PACKAGE GEOMETRY) are considered part of the line generated by a single `add_line` command. However, the major element for all other segments

(generally BOARD GEOMETRY) are part of a major element determined by the actual polygon defined by connecting the endpoints of all the individual segments.

### REFDES

The reference designator of the component. If a function (or its pins) has not been assigned, this field is blank. This field is also blank for a symbol, with no component assigned to it. That is, the field is the actual name given to the component, and not generated by looking for display text of class REFDES.

### REFDES_SORT

The reference designator in a form for sorting. The sort form enables a standard text string sort, to sort the numeric part of the reference designator in correct numeric order. This field should not be used for display purposes. If the *refdes* field is blank (unassigned functions), the function designator is used to fill in the value of this field.

⚠ It is a function designator and not a reference designator because the *REFDES* or *FUNC_SLOT_ NAME* fields are blank.

### SEG_CAPACITANCE

The capacitance of an etch segment.

Presence of this field in your view results in a slower run.

### SEG_IMPEDANCE

The impedance of an etch segment.

### SEG_INDUCTANCE

The inductance of an etch segment.

### SEG_PROPAGATION _DELAY

The propagation delay of an etch segment.

### SEG_RESISTANCE

The resistance of an etch segment.

### START_LAYER _NAME

The start layer name (subclass) for a COMPOSITE PAD. See the description of END_LAYER_NAME.

### START_LAYER _NUMBER

The start layer number for a COMPOSITE PAD. See the description of END_LAYER_NUMBER and END_LAYER_NAME.

**SUBCLASS**

The subclass name of the geometric element. For ETCH and PAD data, this is the name of the layer on which it has been placed. For other geometry, it is the name of the subclass that was active when it was added to the design. To specify a legal value for a subclass in a filter for example, enter SUBCLASS = "TITLE_BLOCK" (a legal subclass value for the DRAWING FORMAT class). You can also define new subclasses using the *Define* option. See CLASS.

**SYM_BOX_X1**

**SYM_BOX_Y1**

**SYM_BOX_X2**

The lower left (X1,Y1) and upper right (X2,Y2) coordinates, in user units, of a rectangle that bounds the symbol. These fields are derived from the symbol's PLACE_BOUND_TOP rectangle(s). If the user does not explicitly create one when building the symbol, the rectangle is automatically generated by the layout editor.

**SYM_CENTER_X**

**SYM_CENTER_Y**

The X (Y) coordinate, in user units, of the symbol's body center. These fields are taken from the text point of text attached to the symbol with subclass BODY_CENTER. If that text does not exist, it is calculated from the PLACE_BOUND_TOP rectangle(s).

**SYM_EXTENTS_X1**

**SYM_EXTENTS_Y1**

**SYM_EXTENTS_X2**

The lower left (X1,Y1) and upper right (X2,Y2) coordinates, in user units, of a rectangle that encloses the drawing extents of the symbol.

**SYM_HAS_PIN_EDIT**

The value is YES if the symbol contains any instance-edited pins.

**SYM_MIRROR**

The field indicates whether a symbol is mirrored. If the symbol is mirrored, this field is YES.

**SYM_NAME**

The name of the symbol. For a placed component, SYM_NAME should be the same as the package name.

**SYM_ROTATE**

The angle of rotation in degrees, with three decimal places of accuracy.

**SYM_TYPE**

The type of symbol. Values can be: MECHANICAL, BOARD, or FORMAT.

**SYM_X SYM_Y**

The X (Y) coordinate of the origin of the symbol. Note that mirroring and rotation in the layout editor do not change the origin of the symbol.

**SYM_ZONE_NAME**

The name of stackup zone in which the symbol is placed.

**SYM_ZONE_NAME**

Sort stackup zone names.

**TEST_POINT**

This field indicates whether a via is a test point. Values are TOP, BOTTOM, TOP_MANUAL, and BOTTOM_MANUAL indicating whether the test point test point is on the top or bottom and whether it was added manually or automatically. The field is blank if the via is not a test point.

**TRELFLASH**

The thermal relief flash name.

**TRELHGHT**

The height of the thermal relief.

**TRELSHAPE1**

The thermal relief geometry.

**TRELSHAPENAME**

The thermal relief shape name.

**TRELWIDTH**

The width of the thermal relief.

**TRELXOFF**

The X coordinate for the offset of the thermal relief.

**TRELYOFF**

The Y coordinate for the offset of the thermal relief.

### VIAFLAG

The surface mount or though-hole type (v = surface mount, empty = through-hole).

### VIA_MIRROR

This field indicates whether the via is mirrored. The value is YES if it is mirrored, NO if it is not.

### VIA_X

The X coordinate of the via.

### VIA_Y

The Y coordinate of the via.

# Appendix B: Combining DFM Rules

When obtaining DFM rules from multiple PCB fabricators there may be subtle differences from one PCB Fabricator company to another in the DFM rules. This is typically not an issue when running low quantities of PCBs, where one fabricator can manage the work load, but in high quantity situations, limiting production to one fabricator is not a viable solution.

To account for differences in DFM rules across fabricators, you, as a PCB Designer, will create a common denominator rule set that will apply for all fabricators. Today this process is accomplished manually, which is error prone, or by an in-house software solution, which might work if the right resources are available to create and maintain the tools.

Use DesignTrue DFM Rule Aggregator to combine DFM rules and produce a set of rules with the common denominator based on the most conservative values. DesignTrue DFM Rule Aggregator is primarily focused on technology files obtained by fabricator partners in the DesignTrue DFM Rules Request (https://pcb.cadence.com/dfm_customer/signin) web portal.

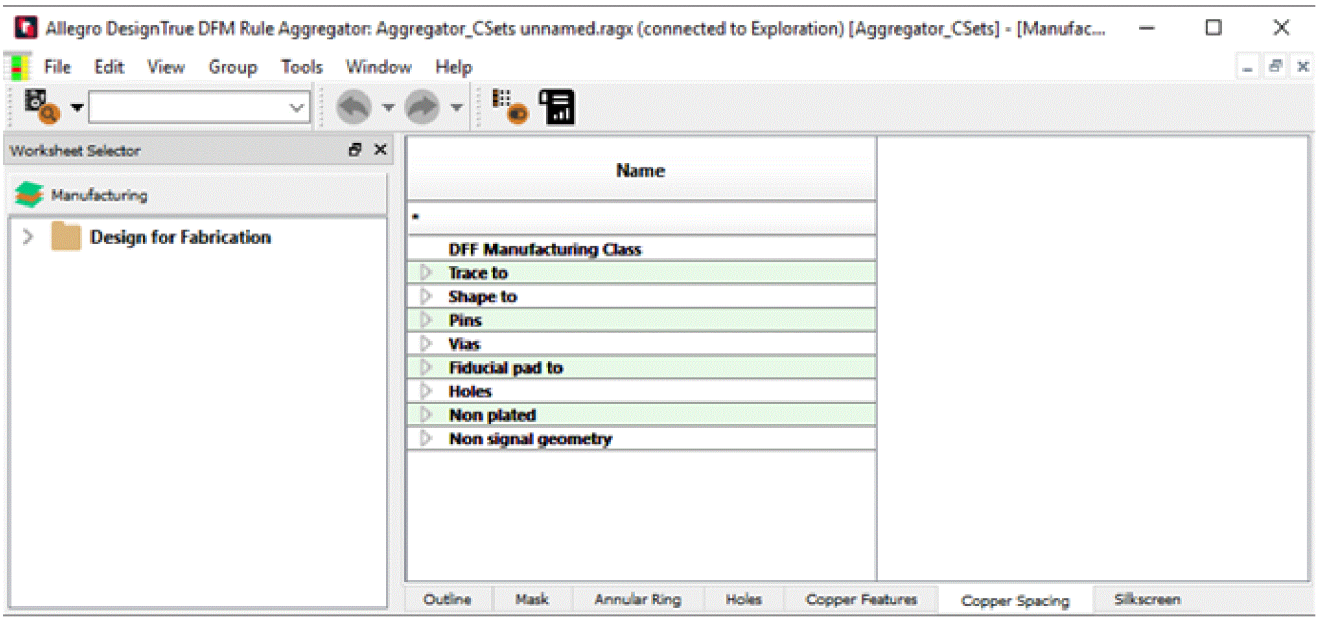
⚠ For more information on the DesignTrue DFM Rules Request Web portal, refer to the *Cadence DesignTrue DFM Web Rules Rule Request Guide*.

This version of the DesignTrue DFM Rule Aggregator is based on Design for Fabrication rule aggregation. Design for Assembly and Design for Test may be added in a future release.

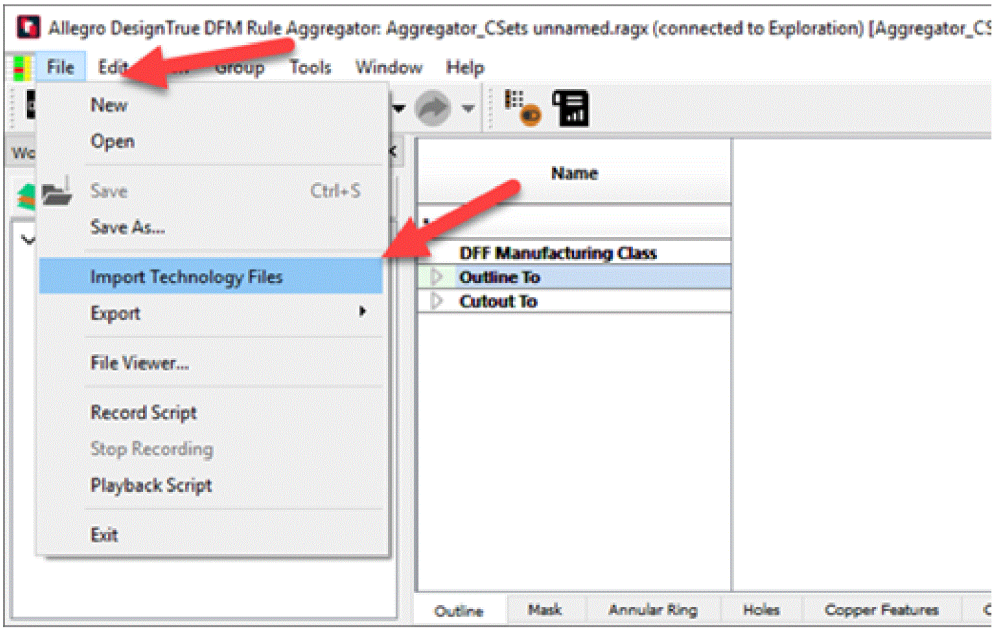
The DesignTrue DFM Rule Aggregator is available under the *Venture*, *Enterprise*, *Sip Layout XL*, and *Silicon Layout* licenses.

## DesignTrue DFM Rule Aggregator Basics

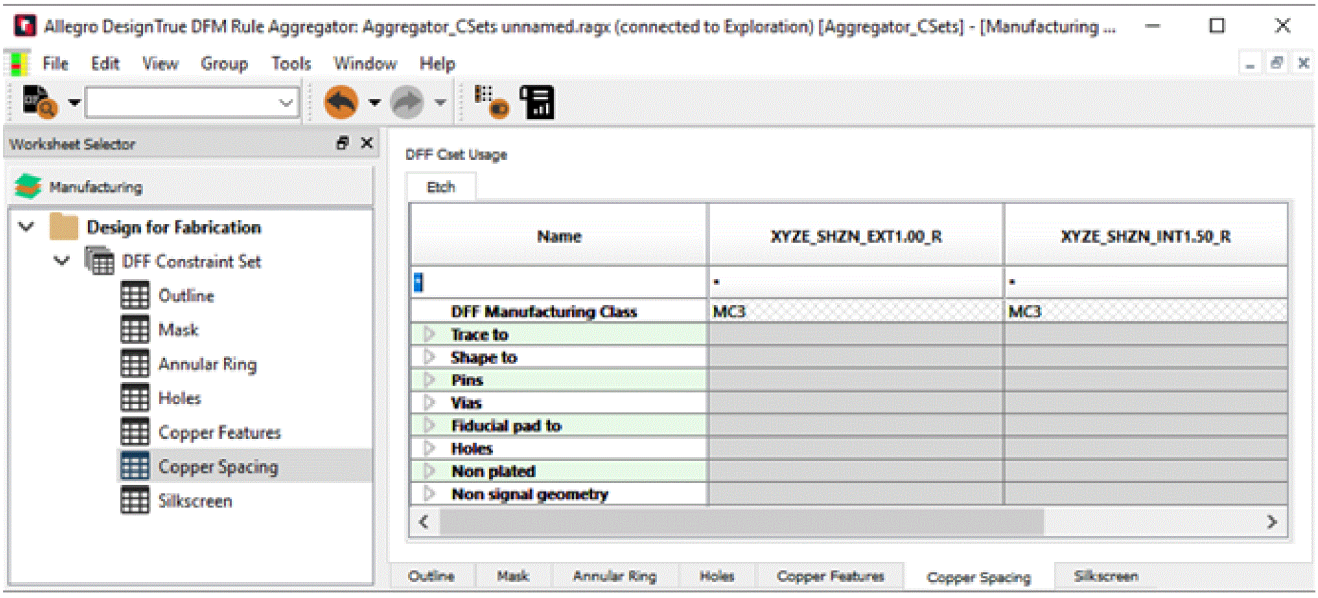
1. To start DesignTrue DFM Rule Aggregator, enter the following command in the system command window or shortcut: `consmgr.exe -dfmAggr`.




2. Import technology files using *File – Import Technology Files*.



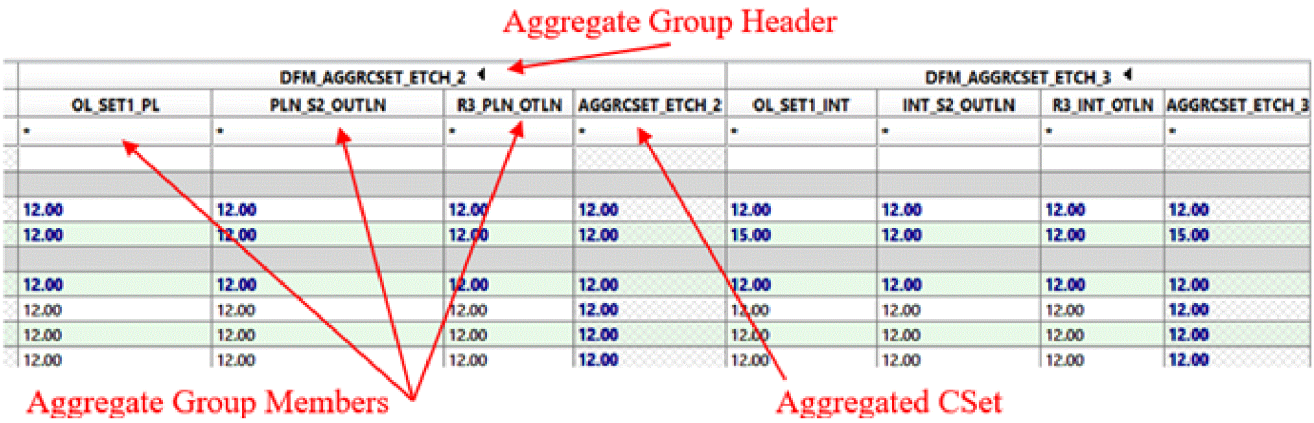

3. Select the files and click *Open*.
   The technology files are imported into the DFM Rule Aggregator.




When using rules obtained from the DesignTrue DFM Web Portal for PCB Fabricators, use *Group – Auto Group All*. Auto grouping is based on attributes stored in the technology file and combines rules that have the same copper weight, manufacturing class, and other attributes.

For non DesignTrue DFM Web Portal technology files, use the following manual process:

1. Select the CSets to be grouped (highlighted in green).
2. Right-click and select *Create Group*.




The Aggregate CSET Group is created. The CSets that are members of the group are positioned on the right side of the group, the aggregated CSET is the leftmost CSet.
When a group is created, the aggregated CSet is created and populated with the most conservative value for each DFM rule. Rule values can be overridden by selecting a different value in the group and then choosing the *Override* pop-up option. The selected value is now placed in the Aggregated CSet and highlighted to indicate an overridden value was selected.

| | | | | | |
|---|---|---|---|---|---|
| All via pads | 3.75 | 3.50 | 3.00 | 3.75 | 3.7 |
| Thru via pad | 3.75 | 3.50 | 3.00 | 3.75 | 3.7 |
| BB via pad | 3.75 | 3.50 | 3.00 | 3.75 | 3.7 |
| Micro via pad | 3.75 | 3.50 | 3.00 | 3.00 | 3.7 |
| Fiducial pad | 5.00 | 5.00 | 5.00 | 5.00 | 5.0 |
| Non plated holes | | | | | |

3. When aggregation is complete, choose *File – Export – Aggregated CSet*.
4. Specify a filename and save.
   A technology file of the aggregated CSets is created. Import this technology file into a drawing to create DFF CSets to be later assigned to design layers.