



Getting Started with Physical Design

Product Version 23.1
September 2023

© 2024 Cadence Design Systems, Inc.
Printed in the United States of America.

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

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1		17
Physical Layout and Package Design User Guides		17
Design Tools		17
Online Documentation Information		20
Types of Information		20
Printing Documentation on Unix		20
User Guide Conventions		21
Running Commands		21
Default Values in Dialog Boxes		22
Command Syntax Conventions		23
Product Installation Information		25
Late-Breaking Information		25
Cadence Customer Response Center		25
Using Cadence Online Support		26
Education Services		26
2		28
Getting Started		28
PCB Editor: Design Flow		28
APD+: Component-Design Flow		29
Completing a Component-Design Flow Using APD+		29
Program Suite		31
Design Workflow		31
PCB Editor: Design Editing Modes		33
Application Modes, the Pre-Select and the Post-Select Use Models		33
Application Mode Types		34
Mode Activation		34
Mode Verification		34
Design Element Selection Model in Application Mode		35
Customizing Datatips		35
Navigating Design Elements		37
Using the Selection Set		38
Choosing Design Elements with the Superfilter		40

Choosing Design Elements with the Object Browser	40
Default Hover-over Selection	41
Context-sensitive pop-up menus	41
Common Options on the Pop-up Menus	41
Example	46
Snap Pick to Pad Edge options in Dimensioning Environment	46
Snap Pick to Pad Edge options with Move and Copy commands	46
Snap Object at an Offset	47
Application Mode Default Command Execution	48
Etch-edit Application Mode Automatic Command Execution	48
General-edit Application Mode Automatic Command Execution	49
PCB Editor: IFP Application Mode Automatic Command Execution	50
Placement-edit Application Mode Automatic Command Execution	50
Shape-edit Application Mode Automatic Command Execution	50
Support for the undo and redo Commands	50
About the User Interface	50
Start Page	51
The Design Window	52
Panning the Design Window	54
The Menu Bar	54
The Toolbar	54
Customizing the Toolbar	54
The Control Panel	56
Working with Foldable Windows	57
The Options Window Pane	58
Active Class and Subclass Fields	58
The Find Window Pane	59
The Visibility Window Pane	60
The Command Window	63
The WorldView Window	64
Using the WorldView Window Pane	64
Displaying Specific Areas of a Design	65
The WorldView Window Pop-Up Menu	65
The Status Bar	67
Status Bar Panes	67
Customizing Status Bar	68
The About Window	69

Customizing Design Canvas	71
Padstack Designer	73
Maintaining Databases	73
Running DBDoctor	74
Partial Versus Full Database Consistency Checks on Saving	74
Database Revision Support	74
Upgrading Differential Pairs from Release 14.x to Release 15.x	74
Differential Pair Log	74
Removing the DIFFERENTIAL_PAIR Property	75
Converting Spacing Constraints	75
Converting a DRC Mode	76
Converting Environment Variables	76
Setting Up a UNIX Environment	76
Using csh environment to access Cadence Allegro tool set	77
Copying the contents of the cshrc file into your own .cshrc file	77
Using sh/ksh Environment to access Cadence Allegro	77
Copying the contents of the profile file into your own .profile file	77
Starting the Layout Editor from an Operating-System Prompt	78
Starting Allegro PCB Editor	79
Starting Allegro PCB Editor in Safe Mode	80
Setting up a pcbenv Directory for Windows or UNIX	81
Creating or Changing the HOME Variable	81
PCB Editor: Creating New Designs	81
Layout	82
APD+: Creating New Design	84
Opening Existing Designs	85
Saving Automatically	85
Activating the Autosave Utility	86
Changing the Default Name (AUTOSAVE) of the Generated File	86
Enabling a Database Check	86
Disabling the Autosave Facility	86
Saving to an Earlier Version	86
Protecting Files with Edit Locks	87
File Types	88
Setting Up a Working Directory Structure	89
Manipulating Design Elements	91
Using the Mouse	91

Left Mouse Button	91
Middle Mouse Button	92
Right Mouse Button	92
Keyboard Shortcuts	92
Select by Window	92
Select by Group	92
Deselect Support	93
Viewing a Design	93
Roaming	93
Zooming	93
View Functions	96
Customizing the User Interface	97
Changing Fonts	97
Command Browser	98
Optimizing the Display	98
Running Commands in the Background	98
PCB Editor: Design Flow	98
APD: Component-Design Flow	99
Completing a Component-Design Flow Using APD	99
Program Suite	101
Design Workflow	101
PCB Editor: Design Editing Modes	103
Application Modes, the Pre-Select and the Post-Select Use Models	103
Application Mode Types	104
Mode Activation	104
Mode Verification	104
Design Element Selection Model in Application Mode	105
Customizing Datatips	105
Navigating Design Elements	107
Using the Selection Set	108
Choosing Design Elements with the Superfilter	110
Choosing Design Elements with the Object Browser	110
Default Hover-over Selection	111
Context-sensitive pop-up menus	111
Common Options on the Pop-up Menus	111
Application Mode Default Command Execution	118
Etch-edit Application Mode Automatic Command Execution	118

General-edit Application Mode Automatic Command Execution	119
PCB Editor: IFP Application Mode Automatic Command Execution	120
Placement-edit Application Mode Automatic Command Execution	120
Shape-edit Application Mode Automatic Command Execution	120
Support for the undo and redo Commands	120
About the User Interface	120
Start Page	121
The Design Window	122
The Menu Bar	124
The Toolbar	124
The Control Panel	126
The Command Window	133
The WorldView Window	134
The Status Bar	137
The About Window	139
Customizing Design Canvas	141
Padstack Designer	143
Maintaining Databases	143
Running DBDoctor	144
Partial Versus Full Database Consistency Checks on Saving	144
Setting Up a UNIX Environment	147
Using csh environment to access Cadence Allegro tool set	147
Using sh/ksh Environment to access Cadence Allegro	147
Starting the Layout Editor from an Operating-System Prompt	148
Starting Allegro PCB Editor	149
Starting Allegro PCB Editor in Safe Mode	149
Setting up a pcbevn Directory for Windows or UNIX	150
Creating or Changing the HOME Variable	150
PCB Editor: Creating New Designs	150
APD: Creating New Design	152
Opening Existing Designs	153
Saving Automatically	154
Activating the Autosave Utility	155
Changing the Default Name (AUTOSAVE) of the Generated File	155
Enabling a Database Check	155
Disabling the Autosave Facility	155
Saving to an Earlier Version	155

Protecting Files with Edit Locks	155
File Types	157
Setting Up a Working Directory Structure	158
Manipulating Design Elements	160
Using the Mouse	160
Select by Window	161
Select by Group	161
Deselect Support	162
Viewing a Design	162
Roaming	162
Zooming	162
View Functions	165
Customizing the User Interface	165
Changing Fonts	165
Optimizing the Display	166
Running Commands in the Background	166
3	167
Using the Layout Editor	167
Limits	167
Setting Drawing Parameters	171
Specifying Text Size	172
Specifying Grids	173
Displaying Net names	174
About Classes and Subclasses	176
Creating User-Defined Subclasses	180
User-Defined Subclass Mapping for DFM Checks	180
Creating Custom Subclass Mapping	181
Working with Highlighting and Coloring	181
Highlighting Design Elements	181
Assigning Colors to Design Elements	182
Unassigning Colors	183
The Color Dialog Box	184
Using the Layers Grid	184
Assigning Subclass Colors and Enabling Visibility	185
Controlling Ratsnest Colors	186
Controlling the Visibility of Individual Elements with Shadow Mode	187

Shadow Mode Display Options	187
Setting Graphics Transparency	188
Creating My Favorites' Folder	189
Saving and Reusing Color Palettes	189
Customizing Design Colors	191
Using the Nets Grid	192
Net Color Inheritance	193
Saving Visibility Assigned to Classes and Subclasses	193
APD+: Highlighting Sets of Pins	194
Plotting a Design	194
Working with Text	195
Defining Text Characteristics	195
Adding Text to Drawings	195
Editing Existing Text or Labels	196
Finding Design Elements	197
The Find Filter Window Pane	197
Determining the Element Selection Hierarchy	200
Using Show Element	200
Finding Elements by Name or Property	200
Using Show Property	201
Using Find by Property from the Console Window Prompt	202
Finding by Name from the Command Window Prompt	202
Using Wild Cards	204
Highlighting Chosen Elements	204
Finding Elements by Using the pick Commands	205
Using Temporary Group Mode	205
Finding Objects by Query	205
Defining a Query	206
Viewing Matching Objects List	207
Displaying Matching Objects in the PCB Editor	212
Finding Buses in Composer/Allegro Design Entry HDL or System Connectivity Manager and Allegro PCB Editor	213
Identifying Buses	213
Bus Selection Syntax	214
Using Buses	214
Highlighting and Dehighlighting Design Elements	215
Automating Design Tasks with Scripts and Macros	215

Using Environment Commands with Scripts	216
Displaying Connectivity	216
Using Data Browsers	216
Displaying Quickview Information	216
Using Qvupdate to Display Quickview Information	218
Database and Library Selections	219
Using Strokes and Associated Commands	220
Default .strokes File	221
Running Commands Using Strokes	221
The Stroke Editor	222
Defining Aliases	224
Assigning Function and Control Keys	225
Layout Editor Limits and Accepted Characters	226
Setting Drawing Parameters	230
Specifying Text Size	231
Specifying Grids	231
Displaying Net names	233
Classes and Subclasses in Layout Editors	234
User-Defined Subclass Creation	238
User-Defined Subclass Mapping for DFM Checks	238
Creating Custom Subclass Mapping	239
User-Defined Subclass Mapping for DFM Checks	239
Creating Custom Subclass Mapping	240
Working with Highlighting and Coloring	240
Highlighting Design Elements	240
Assigning Colors to Design Elements	241
Unassigning Colors	242
The Color Dialog Box	242
Using the Layers Grid	243
Assigning Subclass Colors and Enabling Visibility	244
Controlling Ratsnest Colors	245
Controlling the Visibility of Individual Elements with Shadow Mode	246
Shadow Mode Display Options	246
Setting Graphics Transparency	247
Creating My Favorites' Folder	248
Saving and Reusing Color Palettes	248
Customizing Design Colors	250

Using the Nets Grid	251
Saving Visibility Assigned to Classes and Subclasses	252
APD: Highlighting Sets of Pins	253
Plotting a Design	253
Working with Text	254
Defining Text Characteristics	254
Adding Text to Drawings	254
Editing Existing Text or Labels	255
Finding Design Elements	255
The Find Filter Window Pane	256
Determining the Element Selection Hierarchy	259
Using Show Element	259
Using Show Property	260
Highlighting and Dehighlighting Design Elements	274
Automation of Design Tasks Using Scripts and Macros	274
Using Environment Commands with Scripts	275
Using Data Browsers	275
Displaying Quickview Information	275
Using Qvupdate to Display Quickview Information	277
Database and Library Selections	278
Using Strokes and Associated Commands	279
Default .strokes File	280
Running Commands Using Strokes	280
The Stroke Editor	281
Defining Aliases	283
4	285
Managing Environment Variables	285
The Global Environment File	285
Variables	286
Library Path Variables	286
System Variables	286
Setting User-Defined Variables	287
Modifying a Local env File	288
Defining Library Path Variables in a Local env File	289
Setting Variables at the Console Window Prompt	290
The set Command	290

The settoggle Command	291
Command Examples for settoggle	292
The User Preferences Editor	293
Customizing the User Preferences Editor	294
Setting Project Level and Site Customization Variables	295
Project File Variables	295
Setting .cpm Variables	296
Site Customization	296
Using CDS_SITE Functionality	297
Customizing Default Measurement Units Using CDS_SITE	299
Environment Compatibility	300
5	301
Managing Allegro X Physical Databases	301
Database Compatibility Across Platforms	301
Database Compatibility with Previous Software Releases	301
Uprevving	301
Database UPREV (DBDoctor)	301
Saving – Partial Versus Full Database Consistency Checks	302
Script Compatibility	302
SKILL Compatibility	302
APD+: Using the Package Design Integrity Tool	302
Package Design Integrity Checks	302
General	303
Logic	307
Manufacturing	312
Signal Integrity	317
Wire Bonding	319
Adding Checks Using SKILL Functions	324
packageDesignCheckAddCategory	324
NAME	324
SYNOPSIS	324
FUNCTION	324
NEEDS	324
RETURNS	325
SEE ALSO	325
packageDesignCheckAddCheck	325

NAME	325
FUNCTION	325
NEEDS	326
RETURNS	327
SEE ALSO	327
packageDesignCheck.LogError	327
NAME	327
FUNCTION	327
NEEDS	327
RETURNS	328
SEE ALSO	328
packageDesignCheckDrc.Error	328
NAME	328
SYNOPSIS	328
FUNCTION	328
NEEDS	328
RETURNS	328
SEE ALSO	328
6	329
Configuration	329
UNIX-Based Installation Directory Information and Troubleshooting	329
Files That Reference the Installation Directory	330
Checking File References to the Installation Directory	330
Automatically Correcting Installation Directory References	330
Example cshrc File	332
Example profile File	332
Displaying UI Dialog Boxes Correctly	333
Windows-Based Installation Directory Information	334
Licensing Issues	334
Compatibility for Libraries and Designs	334
Symbol Library and Padstacks	334
IBM DFS	334
7	336
Component Design Methodology for Allegro X Advanced Package Designer	336
	336

Problem Statement	337
Acronyms Use	338
A Structured Approach	340
Component Design Considerations and Trade-off Analysis	342
Component Cost Considerations	343
Electrical and Thermal Considerations	343
Foundry Constraint Considerations	344
Summary	344
Designing the Physical Component	344
Introduction	344
Problem Statement	345
A Hybrid Solution	345
MCAD-to-ECAD Data Transfer	346
Component Information	346
Information Transfer	347
The DXF Medium	347
IC-to-Component Transfer	348
Die Information	348
DIE Format	348
Substrate Definition	348
Stackup information	348
Layer Thickness	350
Layer Materials	350
Layer Type	350
Template Files	350
Constraint Definition	351
Introduction	351
Physical Constraints	351
Electrical Constraints	351
Template Files	352
Placement	352
Die-to-Die Placement	353
Die-to-Component Placement	354
Thermal Analysis	354
Die-to-Component I/O Net Assignment	355
Introduction	355
Pin Assignment	355

Priority Nets	355
Routing Concerns	355
Plating Bar (Optional)	356
Pre-Route Signal Integrity Analysis	357
Modeling Wire Bonds	357
Die-to-Die Versus I/O Connections	357
Simulation	358
Estimation	358
Reflections and Delays	358
Simultaneous Switching Noise	359
SSN Simulation	360
Reducing SSN	360
Reducing EMI	361
Power and Ground Plane Definition	361
Introduction	361
Scope	361
Geometry	362
SSN Effects	362
Editing Planes	362
Routing	362
Wire Bond Routing	363
Wire Bond Routing Constraints	363
Component I/O Z-direction Routing	363
Die-to-Component Interconnect	364
Die to Die Interconnect	364
Power and Ground Vias	365
Component-to-Plating Bar (Optional)	365
Post-route Signal Integrity Analysis	366
Crosstalk Effects	366
False Triggers	366
Resolving Crosstalk	366
Constraints and Requirements	366
8	368
Allegro X Advanced Package Designer Flows	368
IC-Driven Flow	368
Component Design Task Flows	379

Performing Component Route Feasibility Based on Die Pin Matrix from IC Layout	379
Component Route Feasibility Based on Die Pin Matrix Flow	382
Establishing Component Route Feasibility in a Standard Component, both Manually and Automatically	383
Manual Flow with APD	383
Component Route Feasibility in a Standard Component: Manual Method Flow	386
Component Route Feasibility in a Standard Component: Automatic Method Flow	388
Creating a Set of Split Rings Around a Complex Wire Bond Die	390
9	393
Classes and Subclasses in Layout Editors	393
10	405
Glossary	405
A	405
B	406
C	408
D	412
E	416
F	418
G	420
H	420
I	421
J	422
K	423
L	423
M	424
N	426
O	427
P	427
R	431
S	434
T	439
U	441
V	441
W	442
X	443

Physical Layout and Package Design User Guides

The Physical Layout and Package User Guides describe design methodologies and concepts for:

- Physical layout systems of printed circuit boards (PCBs)
- Packaging of single die
- Packaging of one or more die components and any number of discrete components

 Many features are common to all three layout editors: Allegro X PCB Editor, Allegro X Advanced Package Designer, and System-in-Package tools. 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. When the documentation does not note a specific editor, this means that the feature applies to all three layout editors.

For information on new features, see the *What's New* document in the user documentation.

Design Tools

Based on the licenses you have purchased and the product choices made by the installer, you may have access to these tools. .

Tool	Used For...
Allegro X PCB Editor	Creating graphic symbols that represent packages, mechanical elements, drawing formats, and custom pads Performing placement, interactive routing, and manufacturing output processes Preparing the design (for example, defining properties, constraints and other design rules, creating keepin and keepout areas, and so on)

Allegro X Advanced Package Designer (APD)	<p>One of a suite of tools that helps you streamline the integration of multiple high-pin-count chips onto a single substrate. The product suite targets the major challenges of connectivity definition and management, physical concept prototyping of the floorplan, including multi-chip die stacks, and die I/O planning to optimize and minimize substrate connectivity routing and signal integrity challenges. You use APD for constraint and rules-driven, detailed physical substrate construction and manufacturing preparation.</p> <p>Enables constraint driven substrate interconnect design, extraction, modeling, and signal integrity analysis with the following features:</p> <ul style="list-style-type: none"> ○ Supports a full front-to-back physical implementation flow for IC Package design. ○ Determines the best package and substrate options early in the IC design cycle. ○ Provides comprehensive design rule- and electrical constraint–driven layout. ○ Incorporates design for manufacturing (DFM) methodologies ○ Improves design flow with intrinsic support for all industry standards <p>Models entire design with Cadence 3D Design Viewer</p>
Padstack Editor	<p>A graphical user interface that lets you create and visualize multi-layer padstacks. This tool eases the definition of complex padstack geometries by visualizing the padstack from the cross-section and plane views.</p>
Allegro \$PRODUCTDEHDL, Allegro \$PRODUCTDECIS, third-party, or OrCAD X Capture CIS schematic or netlist	<p>Assigning Allegro PCB Editor properties to schematic symbols and nets that drive Allegro PCB Editor in placement and routing Generating the information necessary for backannotation to a schematic by running backannotation</p>

Allegro PCB EditorROUTER	A tool that handles high-density designs requiring complex design rules. The Allegro PCB EditorROUTER uses powerful, shape-based algorithms to efficiently use the routing area. In addition, the Allegro PCB EditorROUTER integration with Allegro PCB Editor layout, Allegro PCB SI, and APD provides high-speed constraint management across the entire design flow.
SigNoise	Obtaining timing delay estimates for unrouted connections and noise reflection data before routing
EMControl	Checking compliance to electromagnetic interference rules
Allegro X Constraint Manager	A spreadsheet-based product, which acts as a command center for the correct-by-design process. Constraint Manager establishes, manages, reviews, and validates physical and spacing constraints as well as electrical design rules or constraints that control interconnect signal quality. This powerful tool allows you to graphically create, edit, and review topology templates or electronic blueprints. It provides real-time updates of the spreadsheets, and automatically integrates the results for you.
Allegro X PCB SI	A tool that offers an integrated high-speed design and analysis environment for creating digital PCB systems and integrated circuit (IC) package designs. Allegro PCB SI allows you to explore and resolve electrical performance-related issues in all stages of the design cycle. By exploring and making trade-offs among timing, signal integrity, crosstalk, power delivery, and EMI, you can optimize electrical performance before committing to final design for manufacture.
AXL-SKILL (Allegro Extension Language)	Creating programming constructs that let you access design databases and create custom interactive commands.

Also installed is a number of programs that you can run from an operating system prompt. These programs may display graphical user interfaces when run, or may require that you enter arguments and options from the keyboard.

Online Documentation Information

Many Cadence products are sold and licensed in different configurations based on features and price. However, the online documentation describes the full set of features. Therefore, the information in the online documentation may contain information on features not supported in the configuration you are using.

Use the online documentation to learn about the Cadence products and how they work together to achieve a design objective. Online books document design flow methodology, aspects of design, and details on how the tool performs specific design tasks. You can view and print the books independently of Cadence applications.

Types of Information

The following types of information are available:

- Overviews: Design methodologies and concepts
- Command Reference: Definitions, syntax, and procedures associated with each command
- Command Browser using the `helpcmd` command: Access to complete selection of keyboard commands and any associated online documentation
- Technical Definitions: General terminology associated with development and design

The topic-based, application-specific, context-sensitive Cadence® Help online documentation system enables you to navigate, search, and display the contents of multiple document libraries. Cadence Help supports application types that let you run Flash-based multimedia in Web browsers or media viewers and PDF files in Adobe Acrobat.

Printing Documentation on Unix

If you are using a UNIX workstation, you may have to set the PRINTER variable in your environment file to the desired printer; for example:

```
setenv PRINTER <printer_name>
```

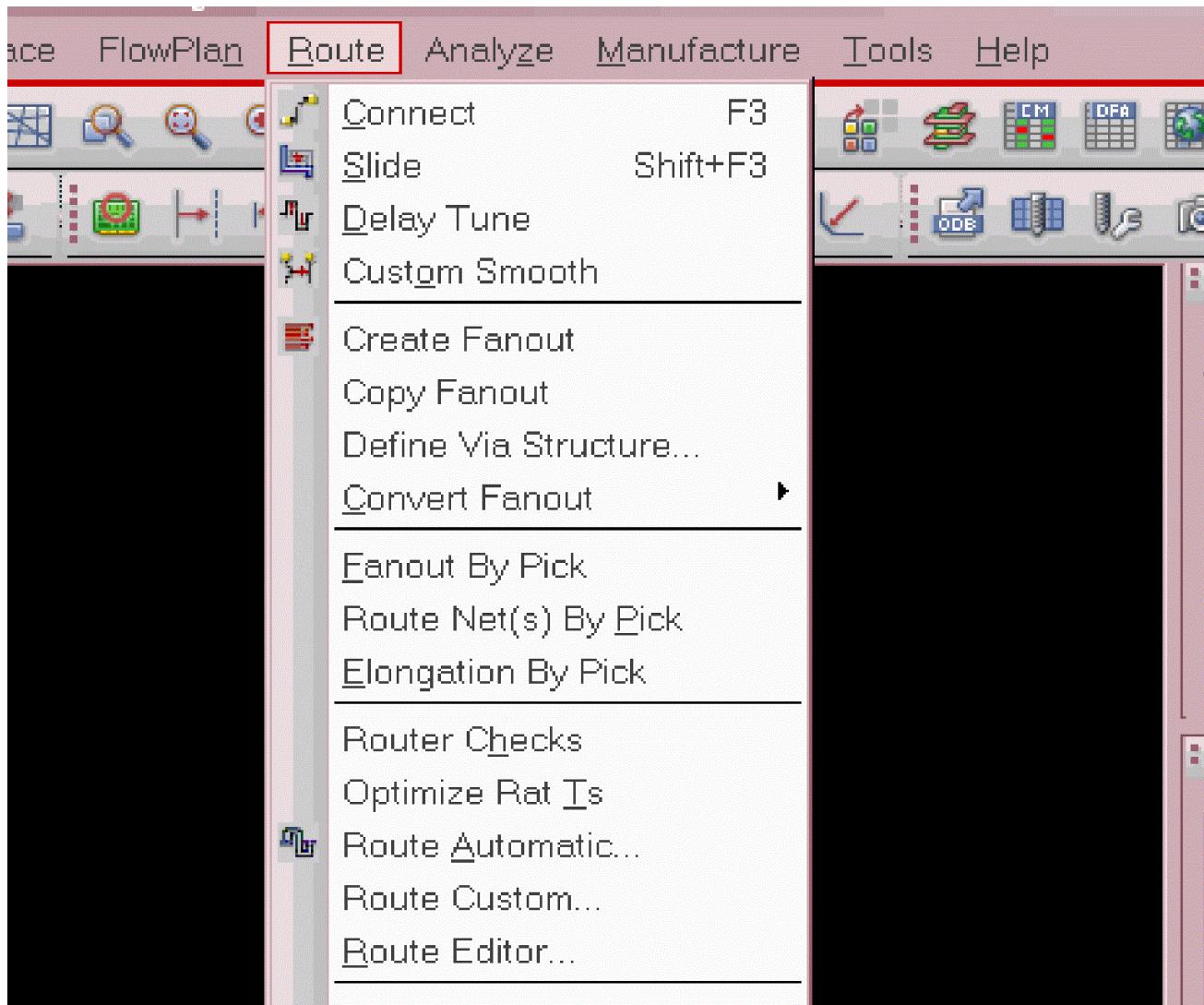
Your system uses that printer for all print jobs.

User Guide Conventions

This user guide employs certain conventions to describe the Allegro user interface.

Running Commands

The user interface uses toolbars (menu items and icon buttons) to run commands. It also provides a console window prompt (>) from where you can enter commands. This user guide documents the menu selection followed by the console command that activates the command. In most cases, the console command is formatted as a hot link that, when you click on it, brings up procedural information on the command in the *Allegro X PCB and Package Physical Layout Command Reference*. For example , the convention used in this document is: choose Route – Gloss – Parameters ([gloss param](#) command).





This means that you choose the menu selection in the user interface or type in the command name as it is written in the user guide at the tool command prompt.

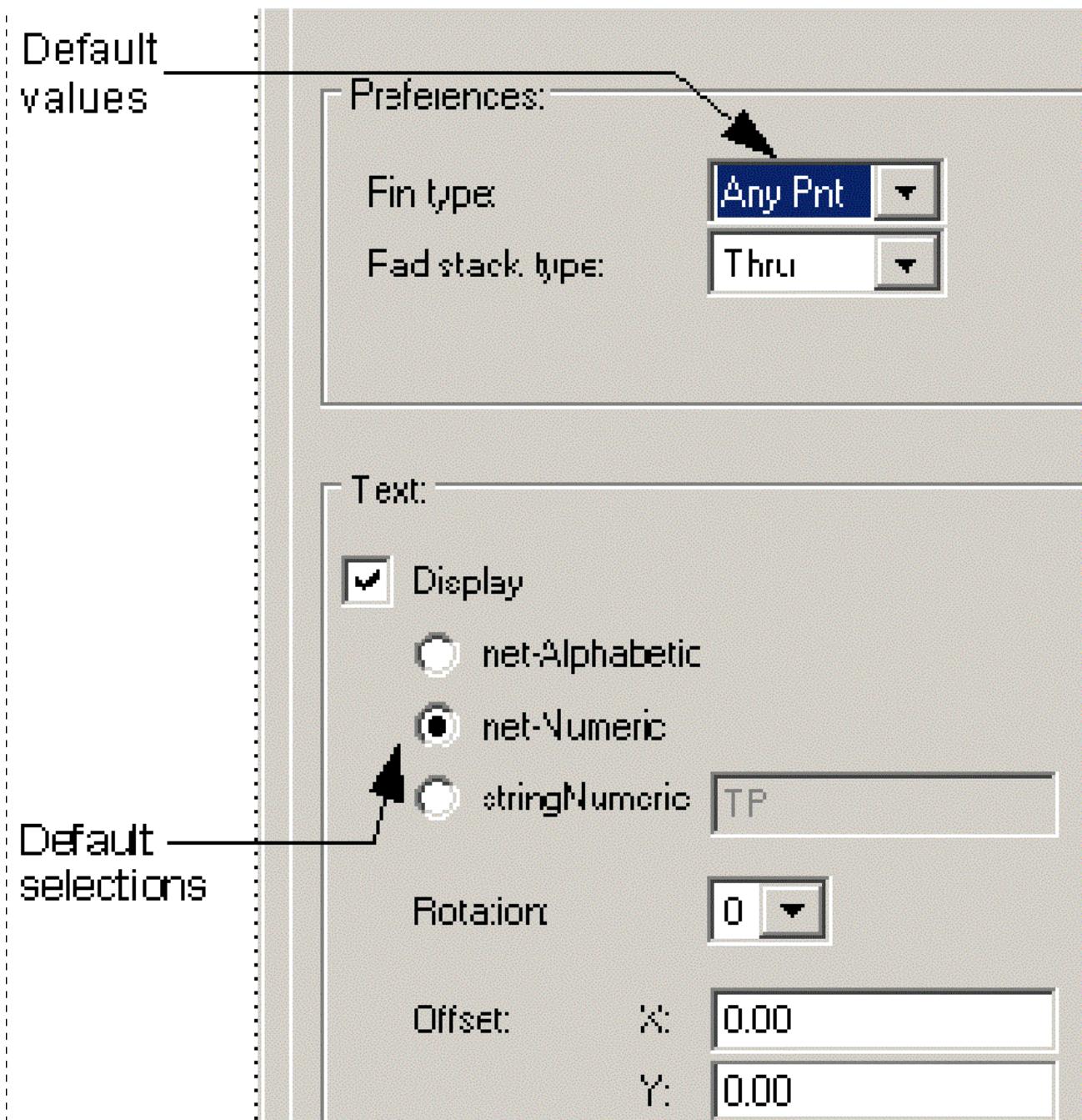
Default Values in Dialog Boxes

Dialog boxes are usually shown with the system default selections:

- Filled buttons are the default selections
- Filled-in fields are the default values

Figure 1 shows the Test Prep Parameters dialog box.

Default Values in Dialog Boxes



Command Syntax Conventions

Although you can run most commands by menu selection from the user interface, you must enter some commands in a Run dialog box or on a command line (for example, `extracta`).

This list describes the command line syntax conventions used in this documentation. For

information on command syntax for the Allegro Extension Language (AXL) see the *Allegro User Guide: SKILL Reference*.

Courier	This guide shows all Run/command line names and examples in Courier font.
nonitalic	Nonitalic words indicate keywords that you must enter literally. These keywords represent command (function, routine) or option names.
< variable >	Words in italics indicate variables for which you must substitute a name or value. Names are case sensitive. Angle brackets (< variable >) may also enclose variables. For example Use the switch option - s < symfile > to refresh specified symbols.
	Vertical bars (OR-bars) separate possible choices for a single argument. They take precedence over any other character. For example command argument argument
[]	Brackets denote optional arguments. When used with OR-bars, they enclose a list of choices. You can choose one argument from the list.
{}	Braces are used with OR-bars and enclose a list of choices. You must choose one argument from the list.

...	<p>Three dots (...) indicate that you can repeat the previous argument. If you use the dots with brackets, you can specify zero or more arguments. If they are used without brackets, you must specify at least one argument, but you can specify more.</p> <p>argument ...: specify at least one, but more are possible</p> <p>[argument]...: you can specify zero or more</p>
,...	A comma and three dots together indicate that if you specify more than one argument, you must separate those arguments by commas.

Product Installation Information

Cadence ships installation and licensing instructions with the product.

Late-Breaking Information

Information about Cadence products that becomes available after the product has been shipped may be published in a Release Alert on Cadence Online Support,

<http://www.support.cadence.com>

or the Cadence ftp site. See [Using Cadence Online Support](#) for more information. For customers without access to Cadence Online Support, contact your Cadence Channel partner.

Cadence Customer Response Center

Technical support is available for customers who have a maintenance agreement with Cadence. If you need to report a problem in the software or documentation, submit a request from your Cadence Online Support account, or contact your Cadence Channel partner. Contact Cadence Customer Support at:

<http://support.cadence.com>

Using Cadence Online Support

Cadence Online Support (COS) can be accessed by customers who have a COS account with Cadence. For customers without access to COS, contact your Cadence Channel partner. The details of how to contact Cadence by phone, fax or e-mail from locations throughout the world are available at the Web page:

<http://support.cadence.com/wps/myportal/cos/psa?uri=deeplinkmin:contacts>

You can also use COS to:

- Search the Customer Response Center Solutions Database.
- Create Service Requests directly with the Customer Response Center.
- Check the status of Service Requests and Cadence Change Management System (CCMS) Change Requests (CCRs).
- Get information on current and upcoming releases.
- Read technical application notes.
- Get SKILL code written by application engineers and other customers.

Education Services

Cadence offers many education services for customers including traditional classes and web-based training, and will customize training for specific needs. OrCAD customers can contact their local Cadence Channel Partner for training services offered. For a description of classes and their schedules, visit the [Cadence Education Services](#) web site.

Getting Started

This user guide describes features and user interface functions for the layout editors.

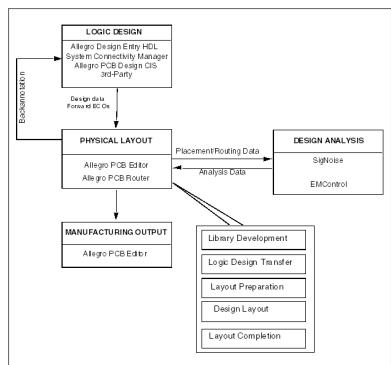
PCB Editor: Design Flow

Cadence's Allegro PCB Editor integrated suites of software tools for systems design help you perform the major tasks of PCB design, including:

- Logic design entry
Create a printed circuit board design based on data from a Allegro Design Entry HDL, System Connectivity Manager, or Allegro PCB Design CIS schematic, or based on a netlist from another CAE system. Then backannotate from the design to the schematic. Update the Allegro PCB Editor designs by performing engineering change orders (ECOs).
- Physical layout
Place design elements and route them, either manually or automatically with Allegro PCB Router.
- Design analysis
Perform design analysis with SigNoise and EMControl.
- Manufacturing output
Generate silkscreens and penplots, and create artwork.

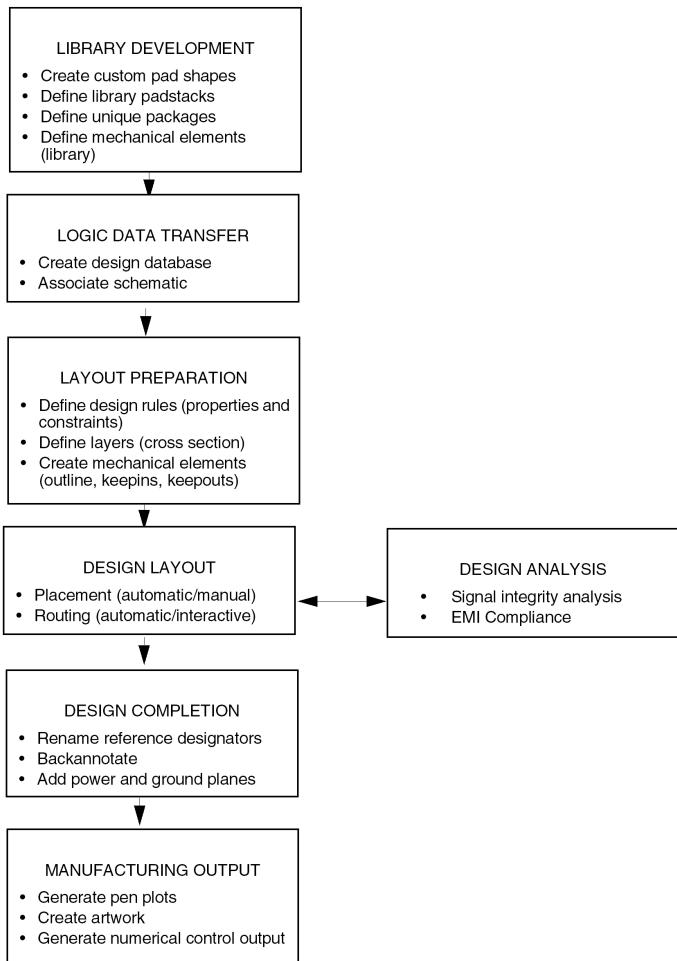
Figure 2-1 shows the functional relationship between Allegro PCB Editor and other Cadence/EDA tools for logic design, physical layout activities, and design analysis.

Figure 2.1: Functional Relationship Among System Design Tools



**Figure 2-2 ** defines the typical PCB design flow process.

Figure 2.2: Design Flow Process



APD+: Component-Design Flow

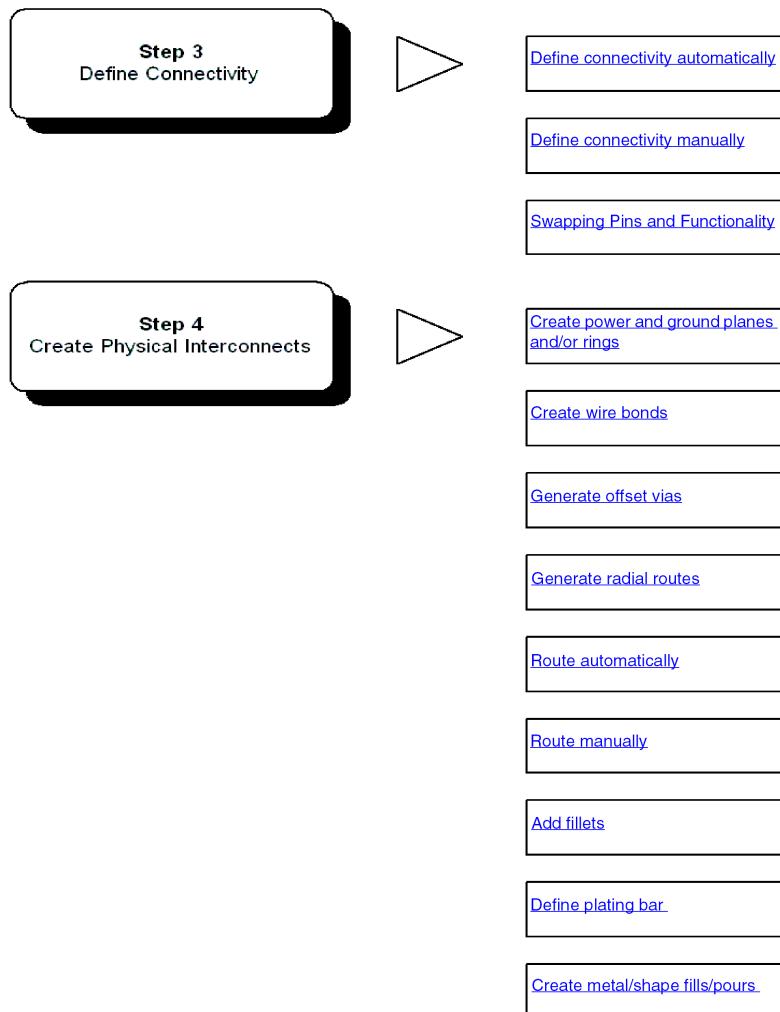
This section includes a general design flow describing the mounting of dies in a component using Allegro Package Designer+ (APD+). The component-design flow covers the design process from the import of data into an empty APD+ design to the export of manufacturing data from APD+. The APD+ physical layout editor imports bare die geometry information from various sources. This information includes the number of die pins, die pin geometry, the die pins' X and Y locations, and the net name associated with each die pin and the die outline extents. The silicon designer can provide this information, ideally in electronic format. APD+ models the component format, and connects traces from the bare die to the component I/O pins.

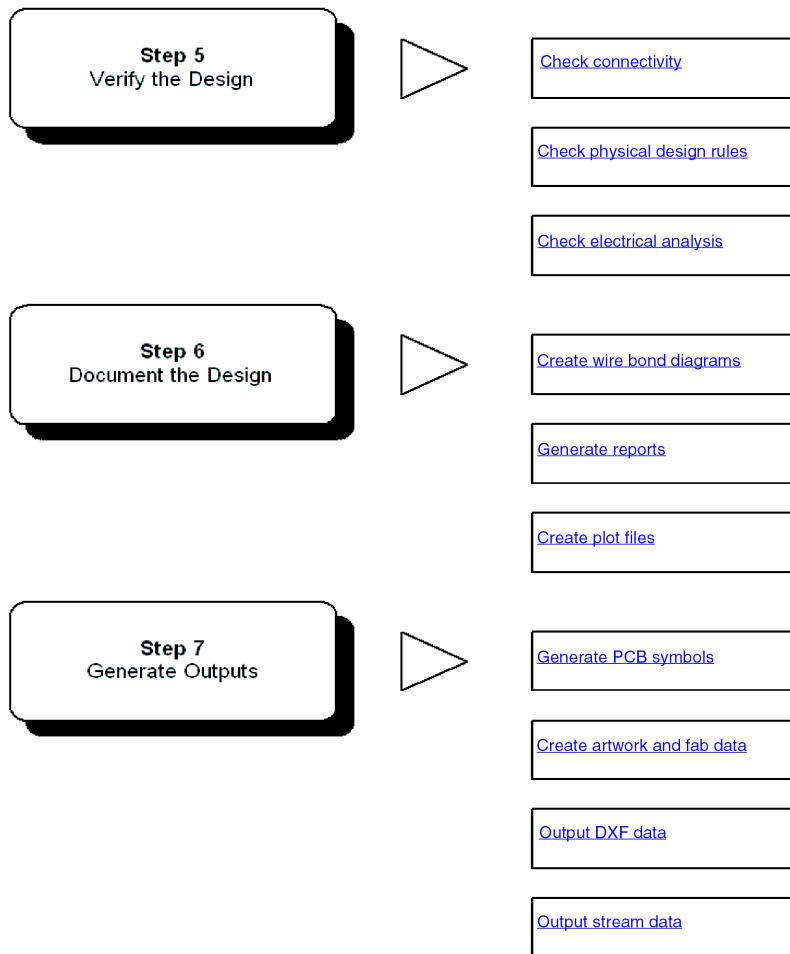
APD+ generates all necessary design data for component fabrication. If the fabricator uses APD+, then the APD+ database can be used as a design-transfer mechanism. For critical or high-speed nets, you can use the Allegro Package SI (AP SI) analysis solution to analyze traces.

Completing a Component-Design Flow Using APD+

Figure 2-3 shows the general steps for creating a component using APD+. Hot links provide access to the descriptions of the specific tasks.

Figure 2.3: Steps for a Component-Driven Flow





Program Suite

When you install the tools on your computer, InstallScape allows you to choose between product tiers. Depending on the tier, different components are available.

A number of command-line utilities are also installed. These programs may display graphical user interfaces when run, or they may require that you enter arguments and options from the keyboard. These programs are documented in the appropriate sections of this user guide.

For more information about other products, see their respective user guides.

Design Workflow

Layout editors also provides design workflow as part of application user interface. The *Design Workflow* pane lists categories of various tasks performed during the design process.

Tool-specific design flows are available in the *Design Workflow* pane of the respective layout editors.

Getting Started with Physical Design

Getting Started--Program Suite

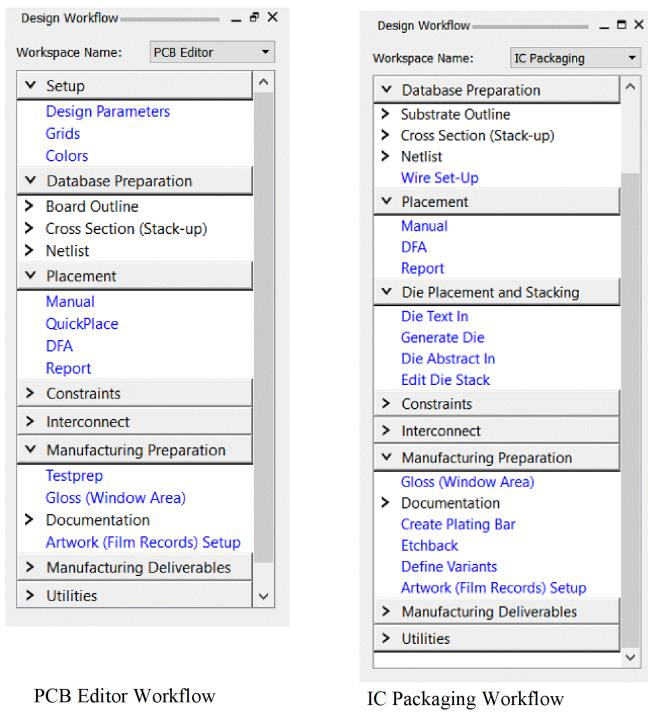
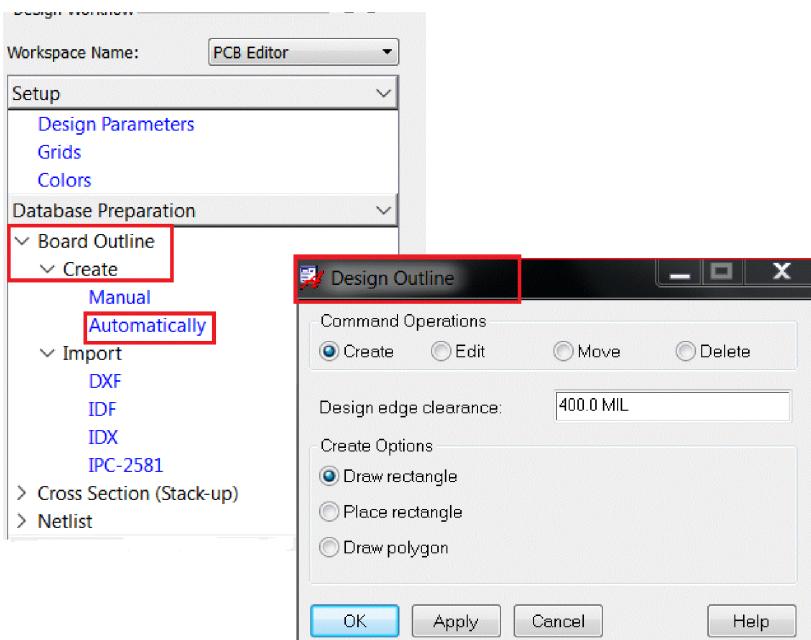


Figure 2-4 Design Workflows in Layout Editors

All the commonly used commands are listed under each task category. Clicking any task in the workflow pane opens the corresponding dialog box.



Layout editors finds the workflow files with its `WORKFLOWPATH` environment variable in the *Paths – Editor* section of the *User Preferences Editor*, available by choosing *Setup – User Preferences* (`enved` command).

Workflow files are XML files, which you can customize according to your requirement. The default workflow files are available at the following location:

`<installation_directory>/share/pcb/text/workflows`

To create a custom workflow, rename the name field and add, change, or remove workflow names and associated tasks in the `<workflow>.xml` file.

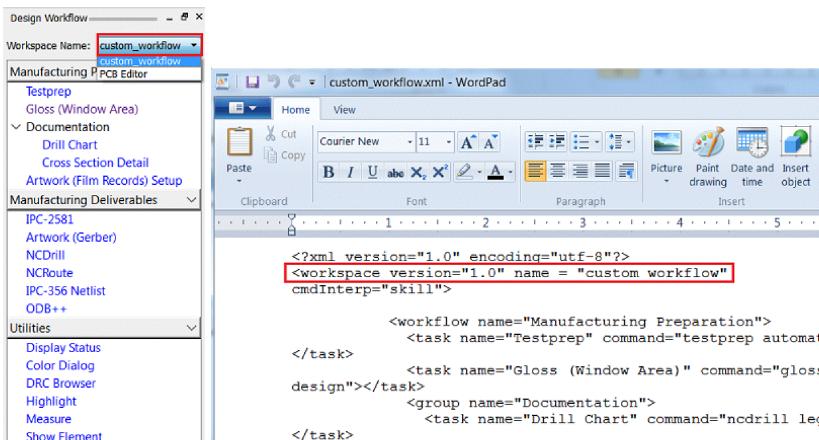


Figure 2-5 Customized Design Workflow

PCB Editor: Design Editing Modes

Allegro PCB Editor contains all the functions required for the layout, interconnect, design rule checking, testability and post processing of a printed-circuit-board design. You can start and run it as a stand-alone tool or as the layout portion of a complete design solution managed with Allegro Project Manager. For further information on Project Manager, see *Getting Started with Allegro PCB Design HDL* and the *Allegro Project Manager User Guide*.

The layout editor's workspace takes many forms — or design editing *modes*—depending on the type of design activity. This affords you the convenience of using a single, variable-mode editor to complete the design. The commands (menu picks and icons) available from the Allegro PCB Editor workspace change to reflect one of the following major design tasks:

- Layout creation and modification
Create the database in this editing mode. Use this mode to perform such tasks as component placement, board or design routing, and other functions.
- Symbol creation and modification
Create symbols for a design in symbol-editing mode. The tool appends the appropriate filename extension when you save a symbol.

You enter a design editing mode by specifying a file type when you choose *File – New* (*new* command) or *File – Open* (*open* command) from the editor. If you are running your layout editor on Windows, you can invoke the file from Windows Explorer (assuming you have set up a file association).

⚠ You must use Pad Designer (a padstack editor) to create or modify a library or design padstack. See [PCB Editor: Design Flow](#) for information on invoking the Padstack Designer.

Application Modes, the Pre-Select and the Post-Select Use Models

The layout editor lets you work in application modes, which provide an intuitive environment in which commands used frequently in a particular task domain, such as etch editing, are readily accessible from right mouse button pop-up menus, based on a selection set of design elements you have chosen.

This customized environment maximizes productivity when you use multiple commands on the same design elements or those in close proximity in the design. Application mode configures your tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the current selection set.

The layout editor comprises two menu use models, post-select and pre-select. The pre-select use model lets you choose a design element (noun), and then a command (verb) from the right mouse button pop-up menu. In the post-select use model, you first select the command and then the design element.

For example, in the pre-select use model the steps to move a symbol are:

- Choose the symbol to move
- Choose Edit - Move

Similarly, the steps to move a symbol in the post-select use model are:

- Choose Edit - Move
- Select the symbol to be moved

By default, the layout editor supports the pre-select use model. This pre-select use model lets you easily access commands based on the design elements you have chosen in the design canvas. The tool highlights the elements and treats them as a selection set, thereby eliminating extraneous mouse clicks and

allowing you to remain focused on the design canvas.

In the pre-select use model, the command ends after the current operation completes. However, the post-select use model lets you perform the command on more than one design element until you choose *Done*.

Application Mode Types

When you initially launch layout editor, it defaults to the general-edit application mode, which lets you perform editing tasks such as place and route, move, copy and mirror.

The pre-selection model, or noun-verb model is enabled by default and allows access to any command provided that no element is currently selected. To work in menu-driven editing mode, that is, to choose a command and then the design element, click in a "black space" area of the design first. Commands not supporting the pre-selection use model ignore the selection set.

Mode Activation

Application mode can be activated in several ways. You can also activate the application mode through the right mouse button on the canvas or in the status bar.

- Choose a menu option:
 - *Setup – Application Mode – General Edit*: General-edit application mode lets you perform editing tasks such as place and route, move, copy and mirror.
 - *Setup – Application Mode – Placement Edit* (PCB Editor only) customizes your environment to fine-tune component placement and alignment. It also allows replication of circuits based on common connectivity and devices. A dockable placement window tab that appears in the *Options* window pane affords immediate access to useful placement options available in the `place manual` command. These include placement by group (that is, refdes, module instances, and so on) and filtering based upon property, room, part number, and net.
 - *Setup – Application Mode – Etch Edit*: Etch-edit application mode customizes your environment to perform etch-editing tasks such as adding and sliding connections, delay tuning, and smoothing cline or cline segment angles.
 - *Setup – Application Mode – Flow Planning*: Interconnect Flow Planner (IFP) application mode customizes your environment to perform route planning for complex (highly constrained, high pin-count) high-speed designs. For example, it enables you to bundle rats and perform bundle flow analysis.
 - *Setup – Application Mode – Signal Integrity* provides quick and easy access to frequently-used SI commands. The SI application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
 - *Setup – Application Mode – Symbol Edit* provides quick and easy access to edit changeable symbols, such as BGAs by adding, moving, deleting, changing, or swapping-pins, in a design. The Symbol Edit application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
 - *Setup – Application Mode – RFEdit* provides quick and easy access to RF command specific to the selected object or group of objects. This application mode configures the tool for a specific task by populating the right mouse button pop-up menu with RF commands that operate on the currently selected RF object or group of RF objects. In addition, the menu also displays some generic RF commands (not specific to the object or objects selected) under a Quick Utilities sub-menu of the RMB.
 - *Setup – Application Mode – Wire Bond Edit* configures the tool for wire bond specific tasks by populating the right mouse button pop-up menu with wire bond commands that operate on the currently selected items.
 - *Setup – Application Mode – Shape Edit* provides quick and easy access to edit the shape boundaries, such as sliding of shape edges with or without corners, multisegment sliding, and adding notches. The Shape-edit application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
- Enter `etchedit`, `generaledit`, `ifp`, `placementeditp`, `symboledit`, `shapeedit`, `signalintegrity`, `rfeedit_Appm`, `wbedit` in the Command Console window pane.
- Choose the appropriate toolbar icon (if added to your toolbar).



Use *Setup – Application Mode – None* (`noappmode` command) to exit from the current application mode and return to a menu-driven editing mode, or verb-noun use model, in which you choose a command, then the design element.

You can also use the `appmode` environment variable to control the application mode that launches on startup, which defaults to the application mode used on previous invocation of the tool.

Mode Verification

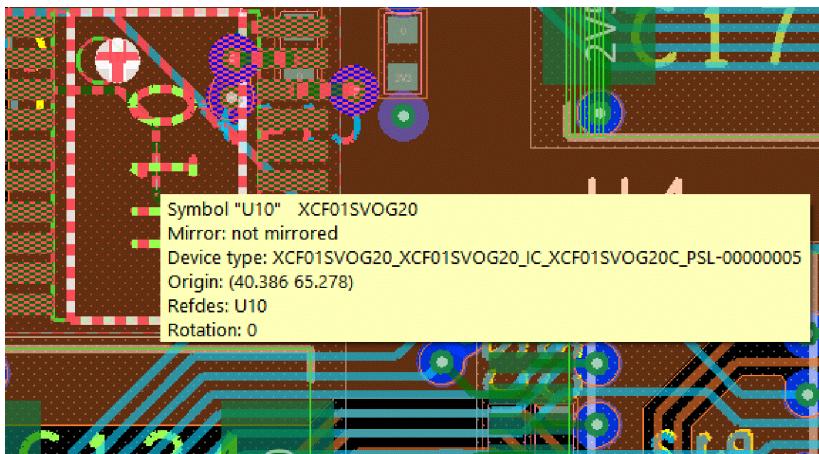
You can quickly check to see which application mode is active by hovering your cursor over the application box in the status bar.

Figure 2.4: Application Mode Status in Status Bar



Design Element Selection Model in Application Mode

As designs become denser, discerning a particular design element in a dense design may be difficult. To help you choose the correct element, hovering your cursor over an element highlights it and, a context-sensitive datatip that identifies the element appears next to the cursor. You can make datatip to appear above the Console Command window pane by setting the variable `datatips_fixedpos`.

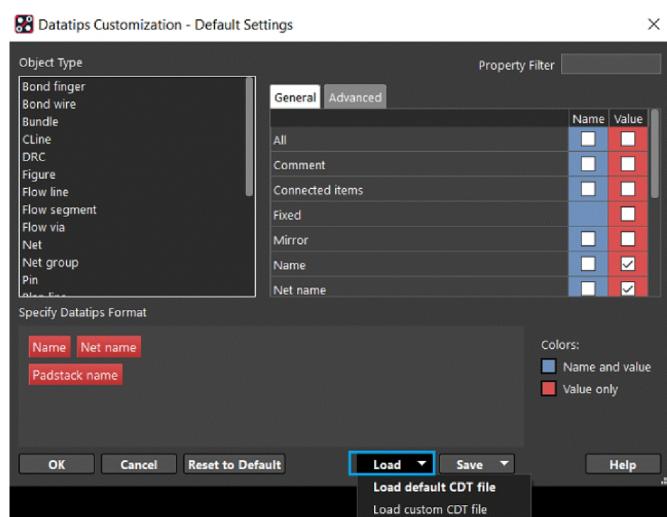


To disable the display of datatips, use the environment variable `disable_datatips`.

⚠ You can set the environment variable for datatips from the *Datatips* category in the *Display* section of the *User-Preference* dialog box.

Customizing Datatips

You can control the information that displays in a datatip for various objects like clines, nets, symbol instances, pins, vias, DRCs and so on by using *Setup – Datatip Customization* ([custom datatips command](#)).



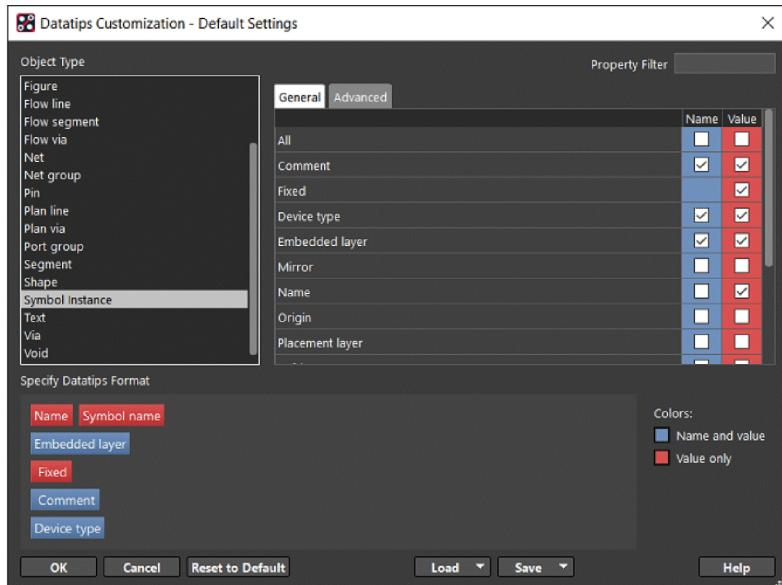
To save the datatip customization use *Save – Save default CDT file* button in the Datatips Customization - Default Settings dialog box. A datatip configuration file `custdatatips.cdt` is saved in the local `pcbenv` directory. You can import the local default CDT file using *Load – Load default CDT file* button.

You can reset the datatip customization to default by using *Reset to Default* button. It will load the .cdt file from the `<installation_directory>/share pcb/text`.

However, you can also create a customized .cdt file for a particular design, by saving customized datatips settings using *Save – Save custom CDT file* button. A datatip configuration file `custdatatips.cdt` is saved in the desired directory. You can then import the `custom.cdt` into another design using *Load – Load custom CDT file* button.

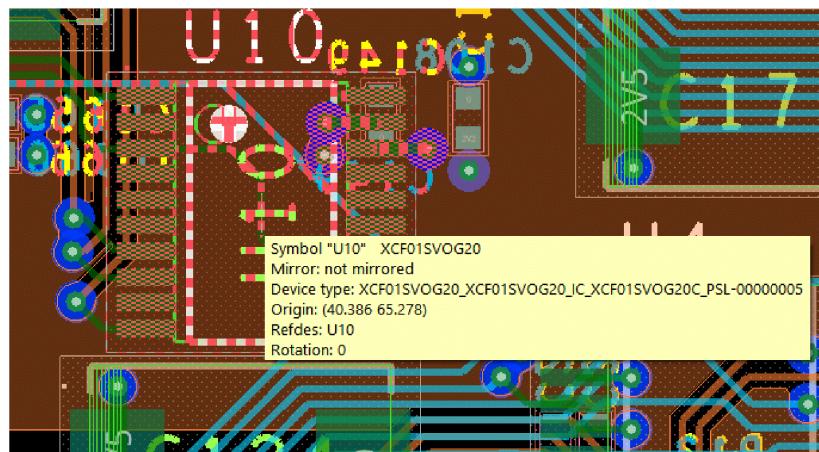
The *General tab* lists the information, available for display, about the element chosen in Object Type. The selected properties are added to the *Specify Datatips Format* field. The red labels indicate that only the *Value* will be displayed in the datatip and the blue labels will display both *Name and value* for the selected Object type.

Figure 2.5: Datatips Customization General tab and Specify Datatip Format



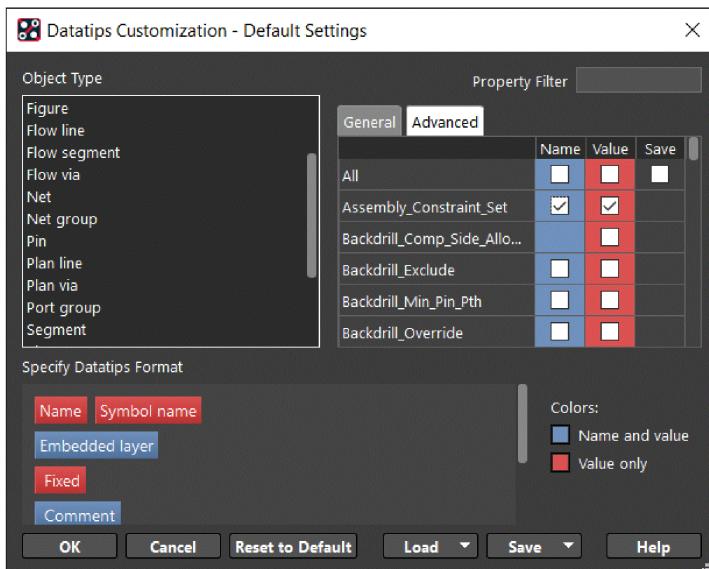
Following image displays the symbol instance datatip after customization:

Figure 2.6: Customized Datatip



The *Advanced tab* displays all properties applicable to the selected element and matching the string in property filter to available for inclusion in the datatip.

Figure 2.7: Datatips Customization Advanced Tab



The Save box appears next to user-defined attributes; select the save box to include the attributes in the .cdt file on saving it.

By default, a datatip disappears 250 milliseconds after the cursor leaves an object. You can, however, set the delay for datatip disappearance by setting the variable `datatips_delay`. For a custom datatip, you can set variable `custom_datatip_remove_delay`. In order to access the datatip features, move towards the datatip otherwise datatip will be removed.

You can turn the datatips on or off using icon in the Display toolbar.

Figure 2.8: Datatip toolbar icons



The dynamic display of datatips can be controlled by setting the variable `focus_followmouse`. The possible options are

Blank	Shows datatips only when the design canvas is in focus, that is, only when the design window is active.
Allegro_derived	Shows datatips when the sub-window is in focus, such as show element, reports and so on.
Anywhere	Shows datatips regardless of the focus. For example, datatip appears on pointing to an element on the design canvas even if any other application window is active.

Navigating Design Elements

While base elements such as cline segments, pins, and vias cannot be parents of other elements, they are the building blocks of hierarchical elements such as nets, clines, and components of which they are made. A pin is a child of a net, as well as that of a symbol and a function. Similarly, a cline could be a child of a symbol and a net. For a symbol with a shape containing a void, for example, the hierarchy may span five levels. The segment comprising the void has a hierarchy of Other Seg – Void – Shape – Symbol – Component.

If you enable more than one base or hierarchical element in the *Find* window pane, the base element determines the hierarchical elements you may choose. You navigate through the hierarchy by using the following or any other pre-defined hot keys:

- **Ctrl-Tab** for all base elements

⚠ The **Ctrl-Tab** key is unavailable when you select by window.

- **Tab** for all hierarchical elements

⚠ The **Tab** key is unavailable when you select by window, which chooses only top-level hierarchical elements.

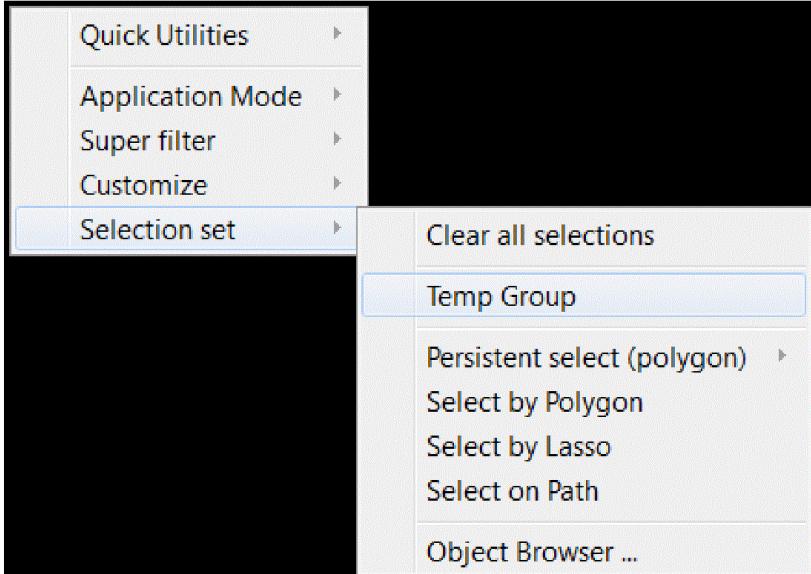
- The *Find* window pane to disable unwanted elements

The base element you initially chose remains highlighted in a different highlighting scheme.

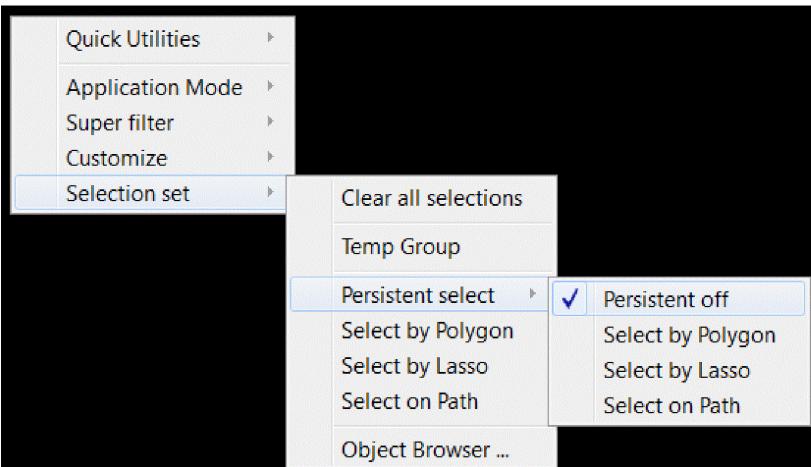
Using the Selection Set

The tool highlights design elements you've chosen in the design canvas as a selection set. Commands that operate on this selection set then appear on the right mouse button pop-up menu. If no elements are selected, when you right-choose and choose *Selection Set*, only the first five options appear; otherwise, the following depending on the number of item types selected and the cursor location (black space or hovering over elements):

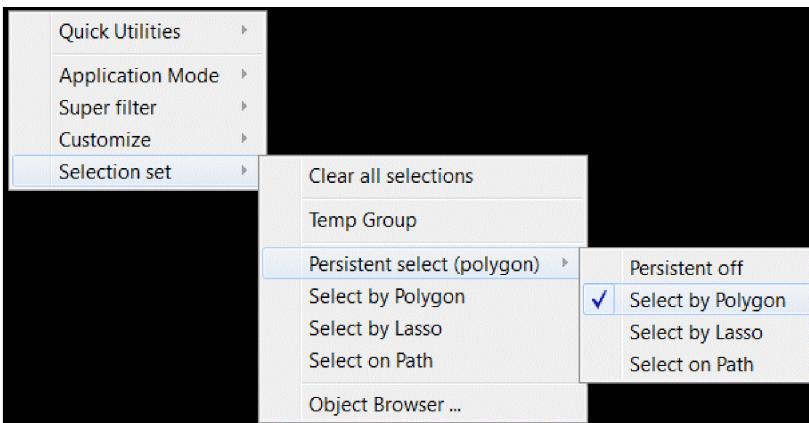
- *Clear All Selections* empties the selection set.
- *Temp Group* lets you choose the elements you want to group together. Each element you choose is highlighted. When you choose all the elements, right-click again to display the pop-up menu and choose *Complete*. All the selected elements are added to a preselect buffer.



- *Persistent select* lets you specify the selection mode (*Select by Polygon*, *Select by Lasso*, and *Select on Path*) for selecting multiple objects



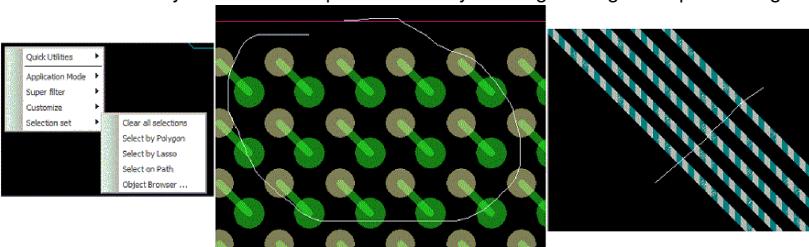
Enabling a mode for persistent select changes the menu and highlights the selected mode.



The active persist select mode remains unchanged during the session and applies to all the commands that access selection modes using right-click pop-up menus.

To turn off the persistent select mode, enable *Persistent off*. By default, the *Persistent select* is disabled.

- *Select by Polygon* lets you choose multiple elements by drawing a polygon around the elements during an editing session.
- *Select by Lasso* lets you choose multiple elements by drawing a open loop path around the elements during an editing session.
- *Select on Path* lets you choose multiple elements by drawing a straight line path during an editing session.



- *Object Browser* lets you search for elements by name or by property.
- *Select* appears only if elements are available to choose at the current mouse position.
- *Narrow Select* lets you refine your selection when multiple elements have been chosen during an editing session.
- *Toggle Select* lets you further refine the elements in the selection set after you select by window.
 - Clicking an element with a minus sign next to it removes it from the selection set.
 - Clicking an element with a plus sign adds it to the selection set.

You modify the elements in the selection set using the following.

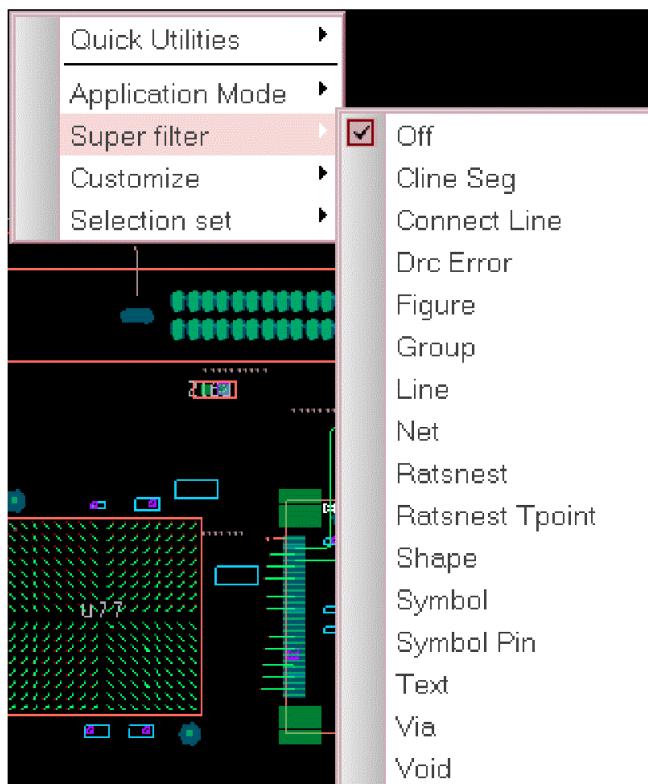
Click (single select)	Clears previous selection set and adds highlighted element at the mouse location to the selection set. If nothing is selectable at this location, clears previous selection set.
Shift + click (extend select)	Adds highlighted element at the mouse location to the selection set.
Ctrl + click (toggle select)	Adds the highlighted element at the mouse location if not already in the selection set. Removes the highlighted element from the selection set if the selection set already contains it.
Selection by window	Clears previous selection set. Adds elements enabled in the <i>Find</i> window pane and that overlap the window to the selection set.
Shift + Select by window	Adds elements enabled in the <i>Find</i> window pane and that overlap the window to the selection set.

Ctrl + Select by window	Removes elements: <ul style="list-style-type: none">• enabled in the <i>Find</i> window pane• overlapping the window• already in the selection set
-------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Choosing Design Elements with the Superfilter

The Superfilter lets you choose a particular element type to refine your selection set and temporarily disable all other elements from the right-mouse button pop-up menu rather than the *Find* window pane. By default, the Superfilter is set to *Off*. This means that all objects in the design are selectable (selection is unfiltered).

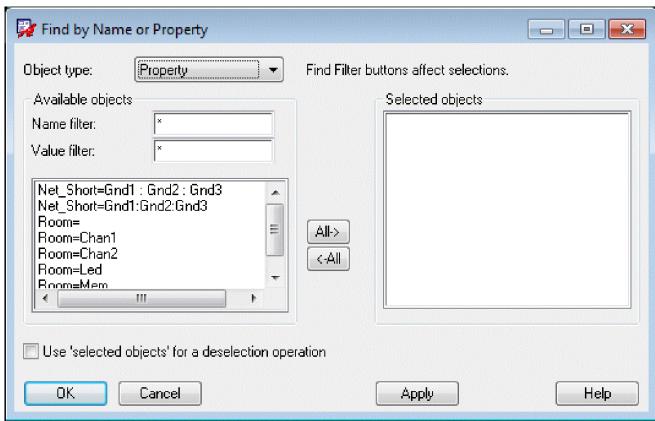
Figure 2.9: Superfilter in PCB Editor



Choosing Design Elements with the Object Browser

The Object Browser lets you select or de-select specific objects in the design by type, name or value. This selection method is particularly useful if the objects you want to operate on are difficult to see or are located on different layers of the design. To access the Object Browser right-click and choose *Selection Set – Object Browser*.

Figure 2.10: Object Browser



Default Hover-over Selection

In general-edit application mode, the highest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when your cursor hovers over an element.

In etch editing and IFP application mode, the lowest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights and becomes selectable when you hover the cursor over it. This is because the lowest level elements are most frequently used. Use the *Tab* key to navigate to other hierarchical level elements.

In placement-edit application mode, the lowest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when you hover the cursor over it, unless the element is a child of a symbol, or if a symbol is a member of a group. In these cases, the group assumes priority over the symbol, and the symbol assumes higher priority over the child element. Use the *Tab* key to navigate to other hierarchical level elements.

In shape-edit application mode, the lowest level hierarchical element (or segment) enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when you hover the cursor over it. In these cases, segment is a child of a shape. Use the *Tab* key to navigate between the segment and shape.

Context-sensitive pop-up menus

Application-mode commands are accessible from a right mouse button pop-up menu based on the current selection set. The commands that populate the pop-up menu depend on:

- Current application mode
- Design elements already in the selection set
- Design elements selectable at the current mouse position

Hovering your cursor over...	...populates the pop-up menu with
an element already in the selection set	commands applicable to the selection set.
an area where nothing is selectable, such as black space in the design	commands and functions independent of the selection set contents, which appear under <i>Quick Utilities</i> , such as <i>Undo</i> , <i>Design Parameters</i> , and <i>Grids</i> , as well as <i>Application Mode</i> , <i>Customize</i> , and <i>Superfilter</i> , depending on the current application mode. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> ⚠ In <i>Shape edit</i> application mode, four additional commands <i>Add Polygon</i>, <i>Add Rectangle</i>, <i>Add Circle</i>, and <i>Delete Islands</i> are available for creating shapes. </div>

You can further filter all elements chosen during the current editing session by right-clicking and choosing *Select Set* from the pop-up menu, then *Narrow Select*. This is useful, for instance, when both base and hierarchical elements comprise the selection set, and you want to only include one of the hierarchical elements, such as symbols, in a particular area.

Common Options on the Pop-up Menus

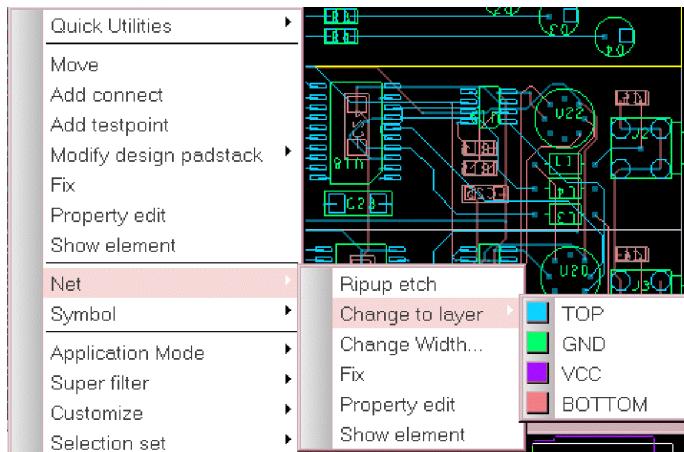
The right mouse button pop-up menus let you perform additional functions, and the available options vary:

- **Quick Utilities** allows access to frequently used functions, such as *Undo*, *Design Parameters*, *Grids*, *Global Shape Parameters*, and *Change Active Subclass*. These options are independent of the selection set contents.
- **Application Mode** lets you choose General Edit, Interconnect Flow Planner (IFP), Etch Edit, Placement Edit customized environments, or none.
- **Superfilter** confines your work to a particular element type, such as all nets, and disables the *Find* window pane.
- **Customize**
 - *Enable Single Click Execution* lets commands execute with a single rather than double-click, such as `add connect` in etchedit application mode
 - *Disable Automatic Drag Operations* initiates select by window rather than `slide` functionality
 - *Enable Shape Selection through Shape Fill*
 - *Reset to Defaults*
- **Selection Set**
 - *Clear All Selections* empties the selection set.
 - *Temp Group* lets you choose the elements you want to group together.
 - *Persistent select* lets you specify the selection mode (*Select by Polygon*, *Select by Lasso*, and *Select on Path*) for selecting multiple objects. By default, the *Persistent select* is off. The active persistent select mode remains unchanged and applies to all the commands that access selection modes using right-click pop-up menus. Select by Lasso
 - *Object Browser* lets you search for elements by name or by property.
 - *Select* appears only if elements are available to choose at the current mouse position.
 - *Narrow Select* lets you refine your selection when multiple elements have been chosen during an editing session.
 - *Toggle Select* lets you further refine the elements in the selection set after you select by window. Clicking an element with a minus sign next to it removes it from the selection set; clicking an element with a plus sign adds it to the selection set.

To work on a single element, hover your cursor over that element and then choose *Select – Select – <element>* from the pop-up menu, which also clears all previous selections.

If the selection set contains a mix of elements, the right mouse button pop-up menu displays pop-up submenus containing commands applicable to those elements.

Figure 2.11: PCB Editor: Selection Set Elements Determine Right Mouse Pop-up Menu Contents



If a command executes on a subset of the whole selection set or on hierarchical parents, corresponding elements append to the selection set and the others are ignored.

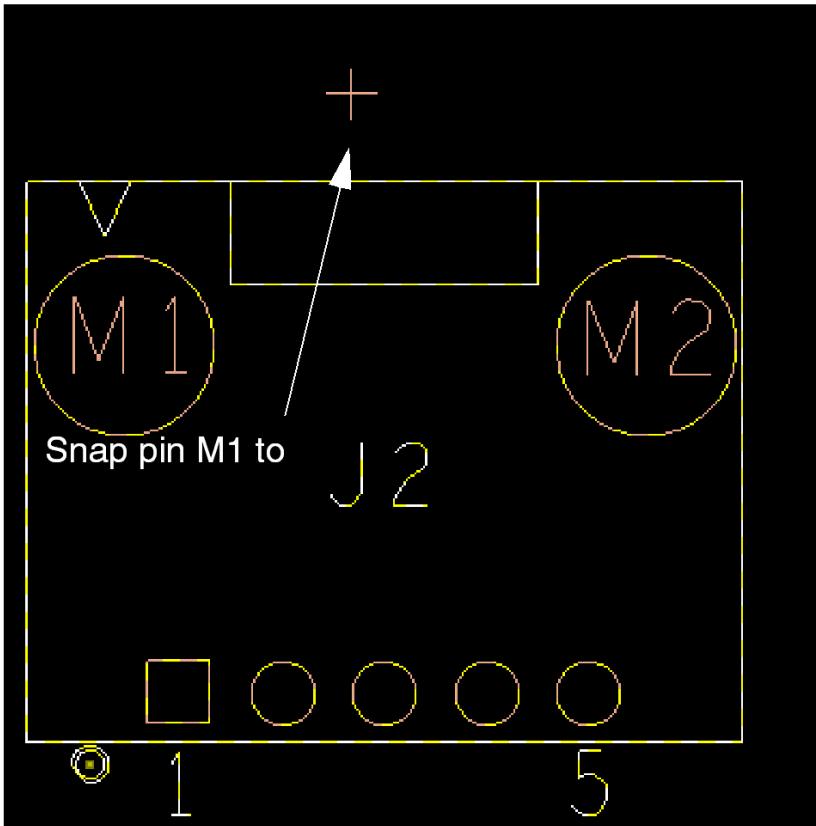
- **Snap pick to**, available in all application modes, lets you choose a snap mode from a list of options found on the right mouse button pop-up submenu when in an interactive editing command, for instance *Move* and *Copy*. On demand snapping lets you control both the starting pick point as well as the destination. The snap to point reaches from the current mouse position to the chosen snapping mode and is based on the chosen snapping mode, visibility, and object type. These parameters supersede the settings in the *Find* filter. It detects the nearest object in close proximity to the current mouse position. If no objects are available, snapping fails. A message appears in the command prompt window indicating that snapping was unsuccessful. The snapping mode is in effect for one pick event only and does not save the current snap mode for future sessions.

- Supported snapping modes include:

Snap Mode	Description
Persistent snap	Lets you select any of the <i>Snap pick to</i> options which will retain snapping for all further selections. Each pick of an operation will snap to the persistent option. The active snap mode applies to all the commands and application modes that performs snapping. By default, this option is off.
Segment vertex	The pick point snaps to the nearest vertex of any of the supported object types.
Segment Midpoint	The pick point snaps to the nearest mid-point of the supported segment type.
Segment	The pick point snaps to the nearest point on the segment.
Intersection	The pick point snaps to the intersection of the segments.
Shape Center	The pick point snaps to the center of a shape.
Arc/Circle Center	The pick point snaps to the center of an arc segment or a circle.
Symbol Origin	The pick point snaps to the origin of the symbol.
Symbol Center	The pick point snaps to the center of a symbol.
Pin	The pick point snaps to the nearest pin.
Via	The pick point snaps to the nearest via.
Figure	The pick point snaps to the nearest figure.
Pad Edge Vertex	The pick point snaps to the nearest vertex of the pad edge.
Pad Edge Midpoint	The pick point snaps to the nearest mid-point of the pad edge.
Pad Edge	The pick point snaps to the nearest point on the pad edge.
Off-grid Location	The pick point directly returns without snapping.
Grid point	The pick point snaps to the nearest grid point.
Snap Offset(0.00 0.00)...	<p>The pick point snaps at a given offset to one of the selected snap destinations. You can define the offset from a snapping destination in two ways by setting either:</p> <ul style="list-style-type: none"> ■ X and Y coordinates ■ or distance and angle <p>If <i>Persistent snap</i> is enabled, the <i>Snap Offset</i> values are remembered throughout the editing session and for all the commands that uses snapping.</p>

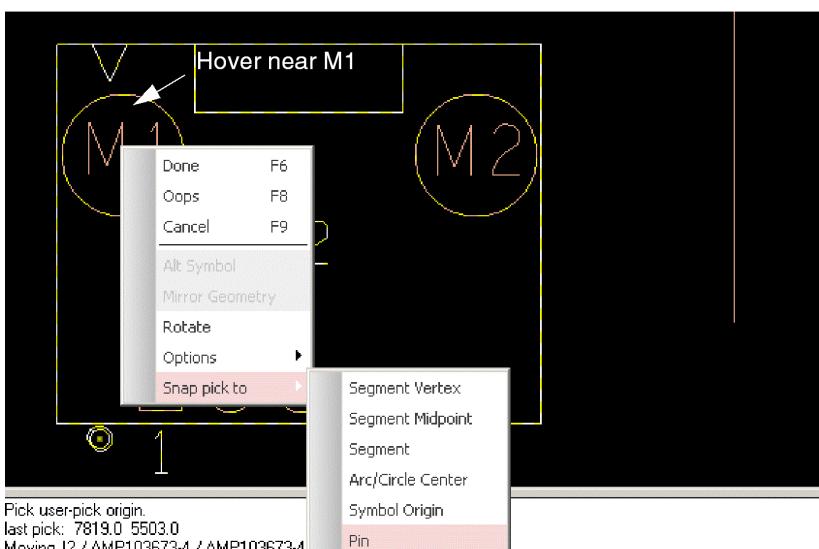
On demand snapping involves the following basic steps for all interactive commands. This example uses the *Move* command to illustrate snapping a mechanical pin associated with a connector to a cross hair target. The *Intersection* option represents the snap object for the cross hair ([Figure 2-14](#)).

Figure 2.12: Snapping To a Cross Hair Target (PCB Editor)



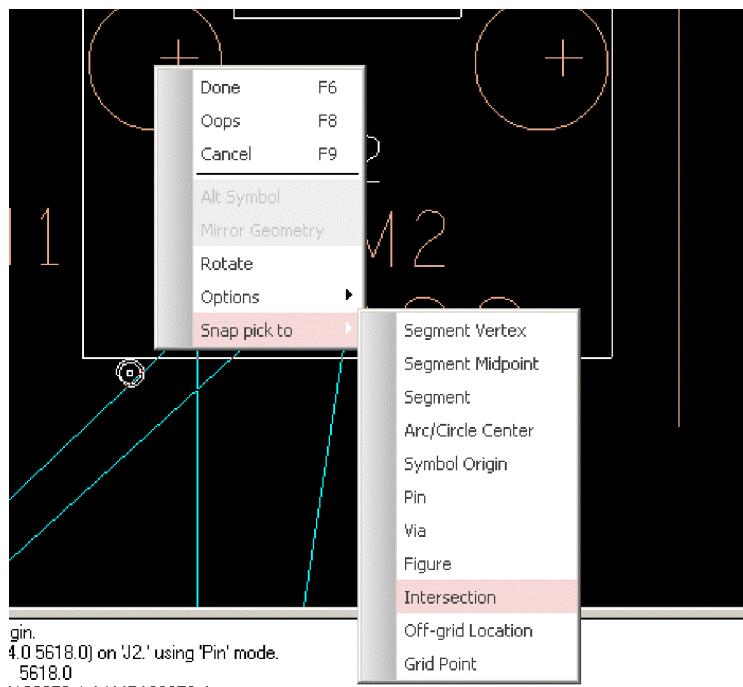
Selecting the mechanical pin (M1) as the initial snap object requires a setting change associated with the *Move* command. From *Edit – Move*, change the rotation point, normally set to *Body Center* to *User Pick*. Next click on the connector when the message: `Pick user-pick origin` appears in the command window prompt. Hover over `M1` and then use the right mouse button popup menu option *Snap pick to – Pin* ([Figure 2-15](#)).

Figure 2.13: Snapping a Mechanical Pin to a Cross-Hair Target (PCB Editor)



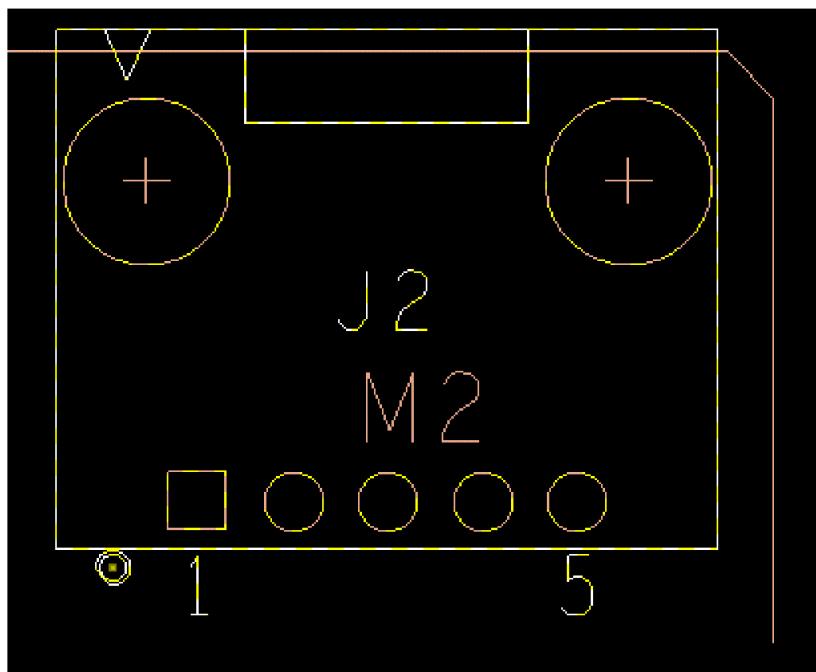
Move the connector with your cursor locked on M1 towards the location of the cross hair target and then use the right mouse button pop-up menu option *Snap pick to – Intersection* to snap the mechanical pin to the target ([Figure 2-16](#)).

Figure 2.14: Snap Pick to Intersection (PCB Editor)



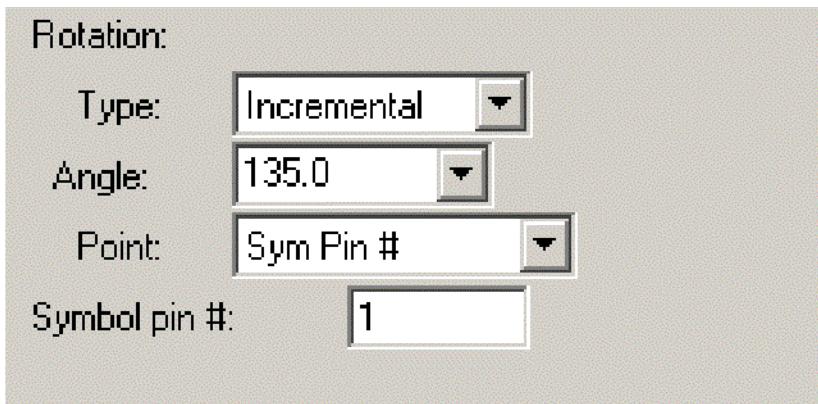
The end result is the center of the mechanical pin on the connector snaps to the intersection of the lines of the crosshairs. A message appears in the command prompt window indicating that snapping is complete

Figure 2.15: Completed Snapping Operation



You can use the same procedure as in this example to snap any pins on a connector to the target. For example, to snap Pin 1 of the connector to the target, change the rotation point associated with the Move command from User Pick to Sym Pin # and then enter a value of 1 in the Symbol pin # field (Figure 2-18).

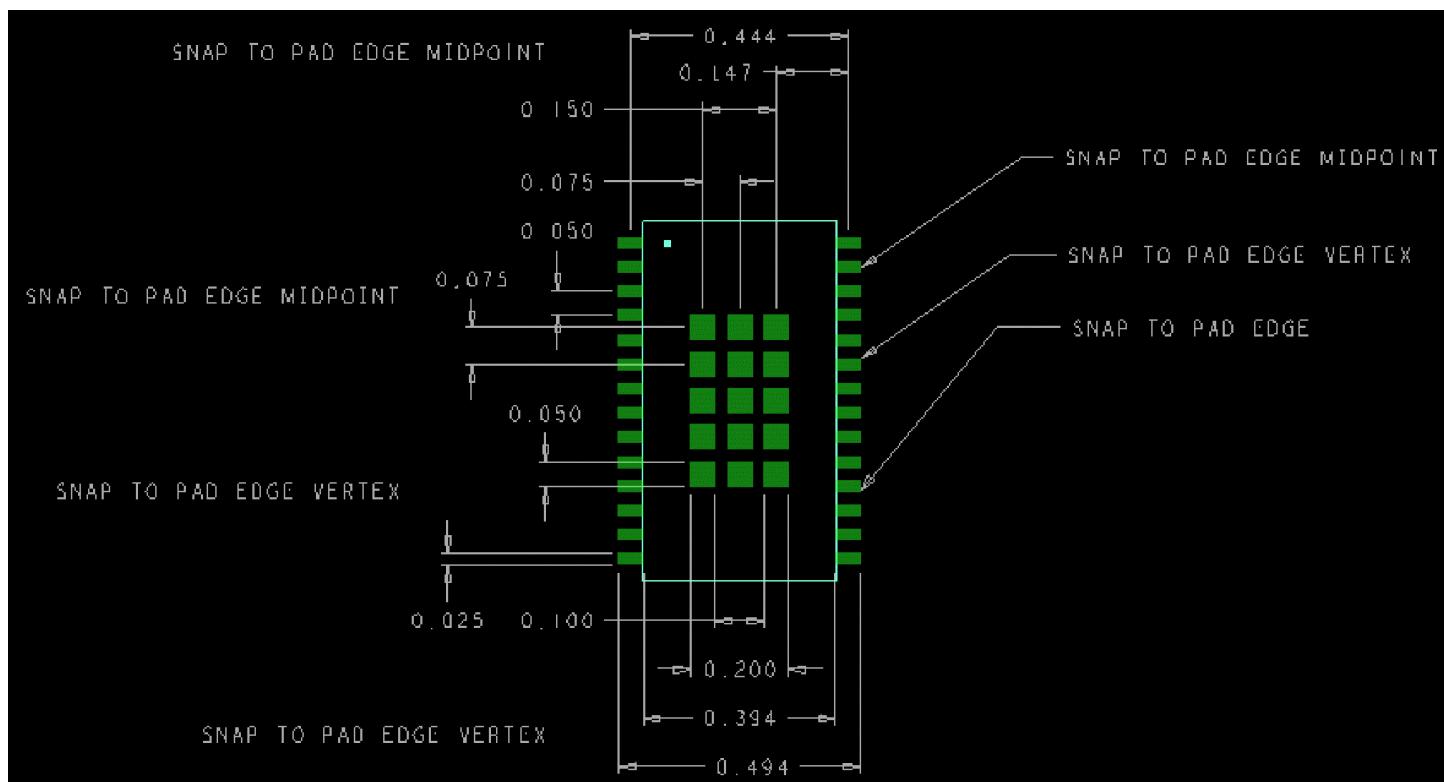
Figure 2.16: Changing the Rotation Point



Example

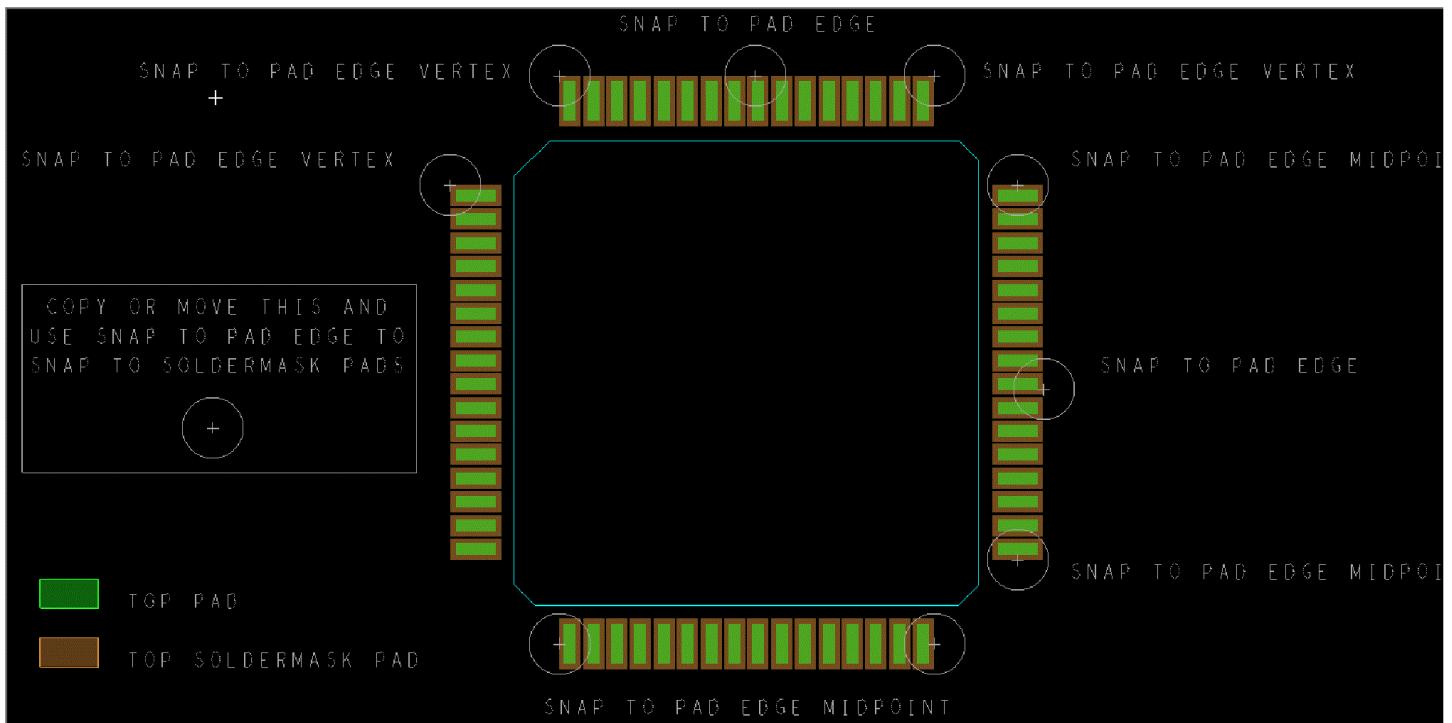
Snap Pick to Pad Edge options in Dimensioning Environment

Dimensioning using *Snap Pick to Pad Edge*, *Snap Pick to Pad Edge Midpoint*, and *Snap Pick to Pad Edge Vertex* snaps to objects that are on the conductive subclasses.



Snap Pick to Pad Edge options with Move and Copy commands

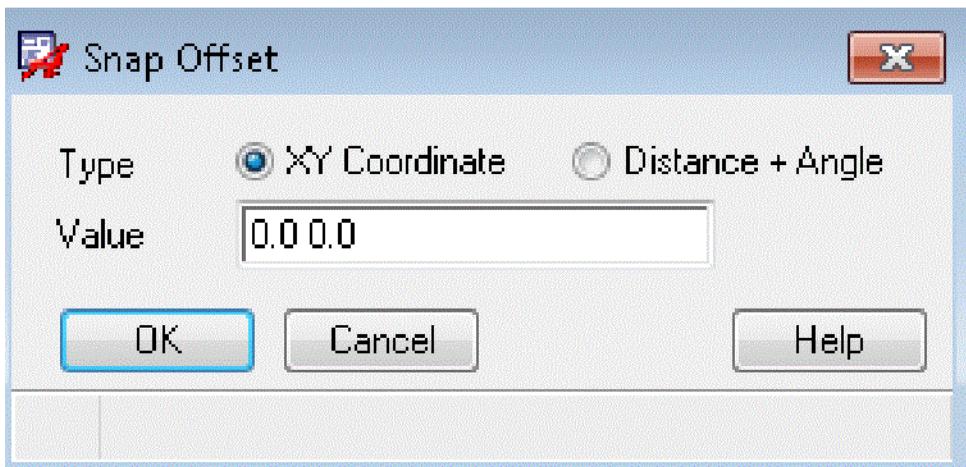
You can use *Snap Pick to Pad Edge*, *Snap Pick to Pad Edge Midpoint*, and *Snap Pick to Pad Edge Vertex* to snap objects to pads on any subclass. In the following figure, change active class and subclass before executing the `copy` and `move` commands, to select pad edges on Pin class.



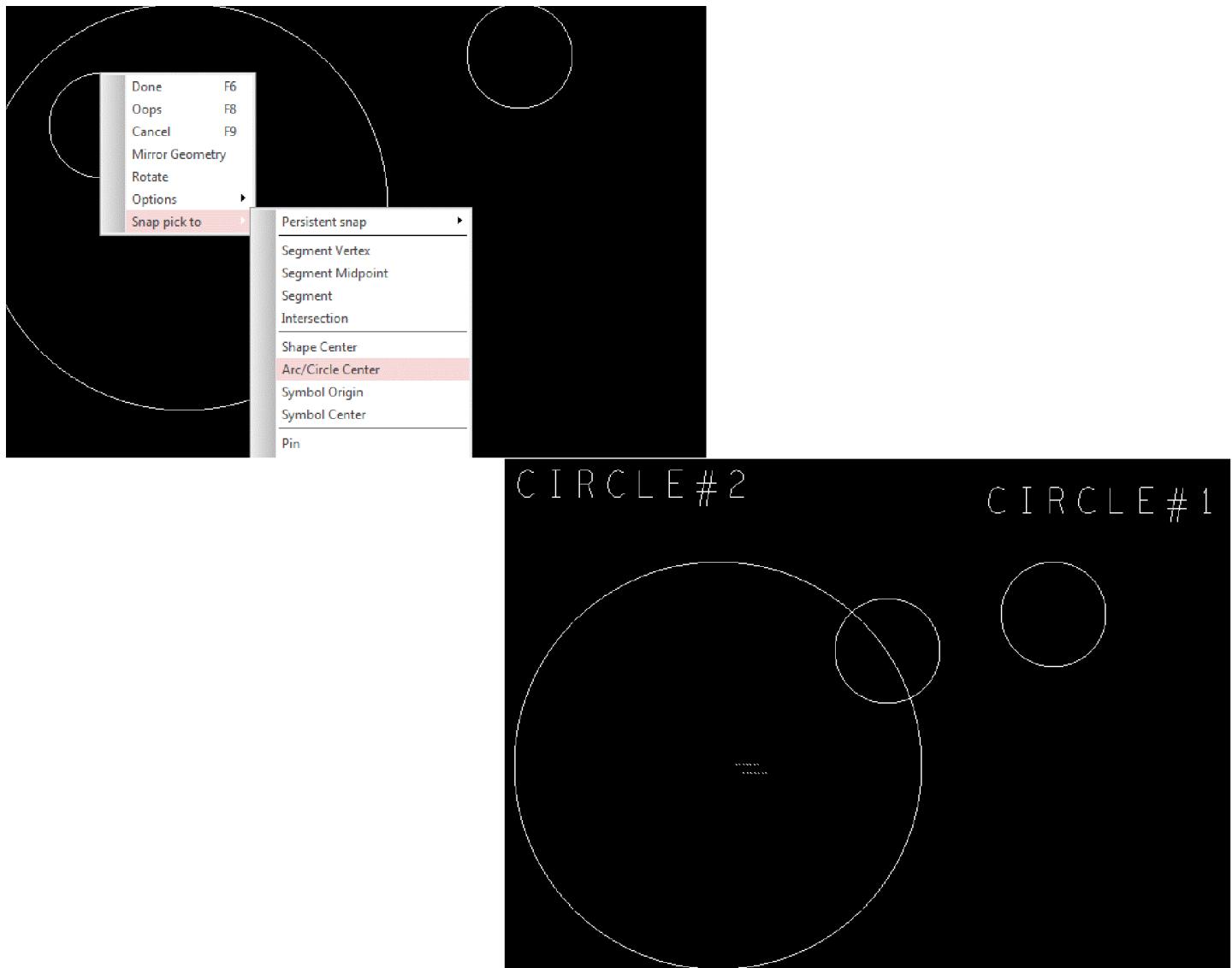
Snap Object at an Offset

You can use *Snap Pick* to along with *Snap Offset* to snap objects to a specific destination and at a defined offset. You can define the offset either by setting X and Y coordinates or by setting a distance and an angle.

For example, copy a circle (circle#1) and snap it to the center of another circle (circle# 2) at an offset. Use `copy` command to copy circle#1, right-click and choose snap mode as *Snap Offset*. Set the offset of (1000.0 1000.0).



Place the cursor at the circle#2, right-click and choose the *Snap pick* mode as *Arc/Circle center*.



Application Mode Default Command Execution

Depending on the current application mode, you can automatically execute a default command with a click, double-click, or a drag-and-drop operation on an element.

Default commands that execute with a click, double-click, or drag-and-drop operation depend on the following:

- Element chosen with the double-click.
- Current application mode; for example, when in etch-edit mode, single clicking a pin executes *Route – Connect* (`add connect` command).

Etch-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements. Single-click execution is enabled by default, which you can disable by right-clicking and choosing *Customize* from the pop-up menu.

Element	Drag	Shift Drag	Ctrl Drag	Shift Ctrl Drag	Double-click
Group	move	move	copy	none	none
Symbol	move	spin	copy	none	move

Pin	none	none	none	none	add connect
Via	slide	move	copy	none	add connect
Cline*	move	move	copy	none	none
Line	move	move	copy	none	none
Shape	move	move	copy	none	none
Frect	move	move	copy	none	none
Rect	move	move	copy	none	none
Line Seg*	slide	none	delay tune	none	slide
Arc Seg	slide	none	none	none	slide
Figure	move	move	copy	none	none
Text	move	move	copy	none	none
Ratsnest	none	none	none	none	add connect
Rat T	slide	move	none	none	none

*Choosing the midpoint of a cline or line seg invokes the `slide` command; to invoke the `add connect` command from the midpoint of a cline or line seg, right-click and choose `add connect` from the pop-up menu.

General-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements in general-edit application mode. Single click execution is enabled by default, which you can disable by right-clicking and choosing *Customize* from the pop-up menu.

Element	Drag	Shift Drag	Ctrl Drag	Shift Ctrl Drag	Double-click
Cline	move	move	copy	none	none
Cline_seg	slide	none	none	none	none
Component_inst	none	none	none	none	none
Figure	move	move	copy	none	none
Drc_error	none	none	none	none	none
Function_inst	none	none	none	none	none
Group	move	move	copy	none	none
Line	move	move	copy	none	none
Net	none	none	none	none	none
Other_seg	none	none	none	none	none
Ratsnest	none	none	none	none	none
Rat_t	slide	move	none	none	none
Shape	move	move	copy	none	none
Symbol_instance	move	spin	copy	none	none
Text	move	move	copy	none	none
Var_pin	none	none	none	none	none
Via	slide	move	copy	none	none
Void	none	none	none	none	none

PCB Editor: IFP Application Mode Automatic Command Execution

The following commands execute by default with a double-click or drag-and-drop operation on these design elements in IFP application mode.

⚠ This feature applies only to Allegro PCB Editor.

To...	Position your cursor here...	Press and hold this key...	Use this mouse action...
Insert and position a new flow vertex.	Over a flow line segment	n/a	Depress the left button and drag-and-drop
Slide an existing flow line segment.	""	Shift	""
Insert a new flow via.	""	n/a	Double -click
Move an existing flow line vertex.	Over a flow line vertex	Shift	Depress the left button and drag-and-drop
Slide an existing flow line vertex.	""	n/a	""
Move an existing flow via.	Over a flow via	Shift	""
Slide an existing flow via.	""	n/a	""
Remove a flow via.	""	n/a	Double -click

Placement-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements in placement edit application mode.

Element Type	Drag	Shift Drag	Ctrl Drag	Single Click
Group	Move	Spin	Copy	Move
Symbol	Move	Spin	Copy	Move
Text	Move	Spin	Copy	Move
RatT	Move			Move

Shape-edit Application Mode Automatic Command Execution

The following commands execute by default with a double-click or drag-and-drop operation on these design elements in shape-edit application mode. Single click execution is enabled by default, which you can disable by right-clicking and choosing Customize from the pop-up menu.

Element Type	Drag	Shift Drag	Single Click
Segment	Move and Slide	Move	Move and Slide
Vertex	Move and Slide	Move	Move
Shape	Move and Slide	Move	Move

Support for the `undo` and `redo` Commands

Using the `undo` command preserves the selection set that existed when you initially launched the command whose results you subsequently have reversed.

About the User Interface

The Allegro layout editor features a task-oriented user interface, with the following components:

- *Start Page*
- *Design Window*
- *Menu Bar*
- *Toolbar*

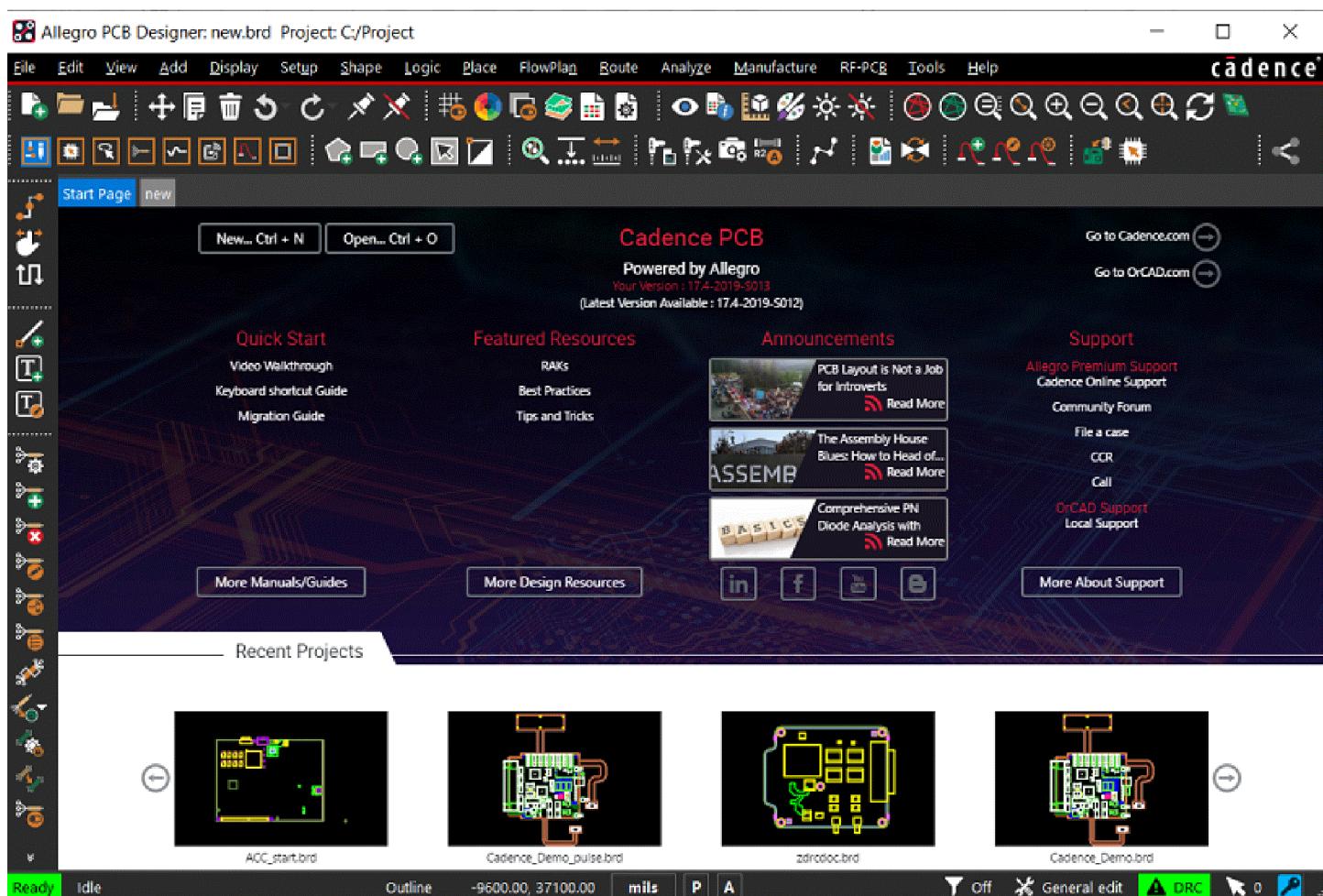
- *Control Panel* with these foldable window panes
 - *Options*
 - *Find*
 - *Visibility*
- *Command* foldable window pane
- *WorldView* foldable window pane
- *Status bar*
- *About Window*

 Wherever user input is required, the interfaces support English (United States) format for numbers.

The design is by default appear in dark theme except for the title bar. To change the theme, set the environment variable *allegro_theme* in the User Preferences Editor and restart the application.

Start Page

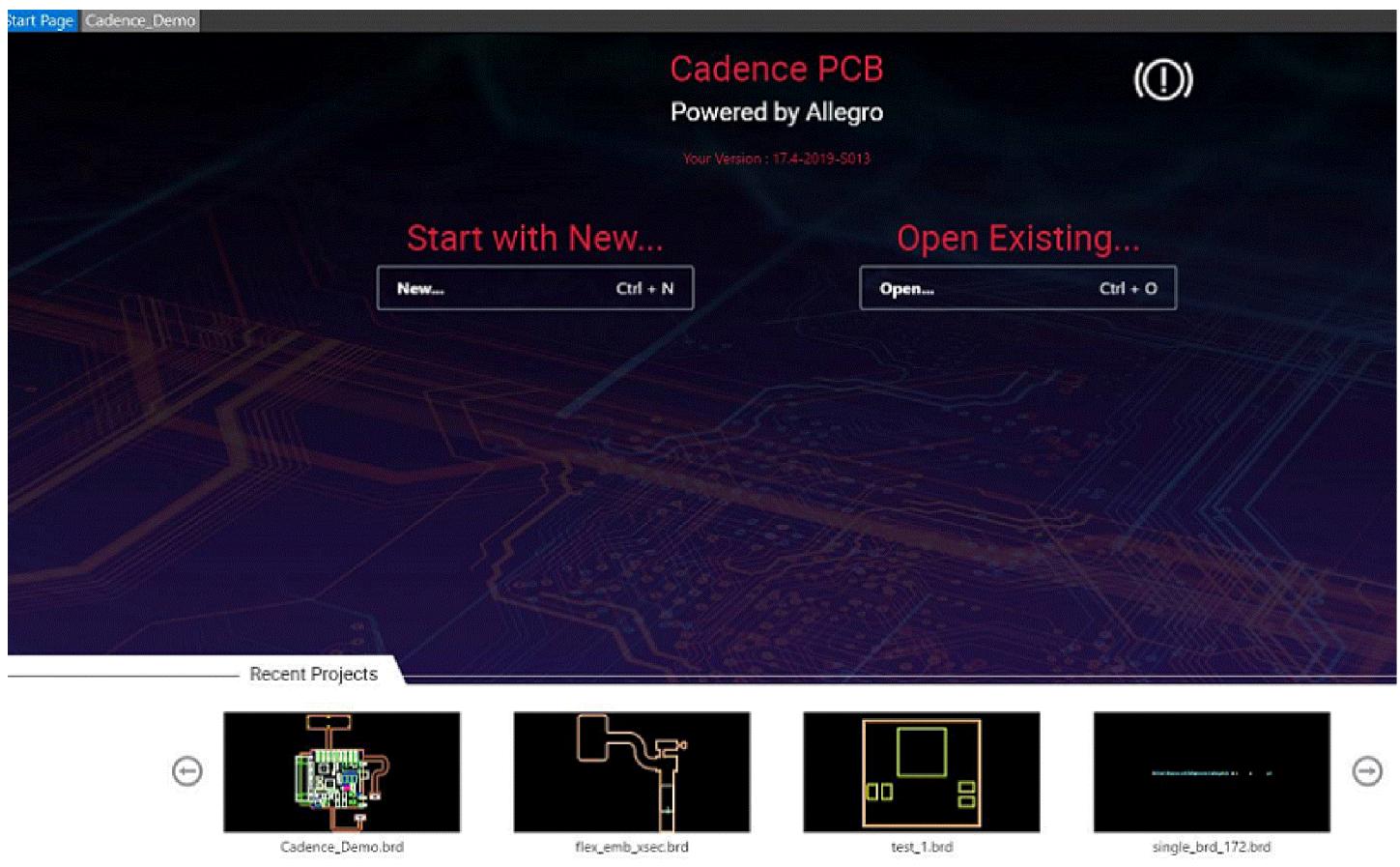
On starting a layout editor, you see a renewed, content-rich, and reorganized start page. Using the start page, you can easily access a variety of information, and projects. From this page, you can read about layout editors, go through brief descriptions of the available features, and access quick start guides and video walkthroughs. You can also access help content, product announcements, and industry news. The page also provides contact information for connecting to Cadence customer support.



The *Start Page* tab provides options to create a new design or to open existing designs. The *Recent Designs* section provides an easy access to recently

opened designs. Click the design name to open it in a new tab. The *Recent Designs* tab displays a quickview of the designs that are saved in the current release. The designs created in earlier releases can be accessed through buttons only and no quickview is available.

When you are working offline, without an internet connection, only static content of the start page is available.

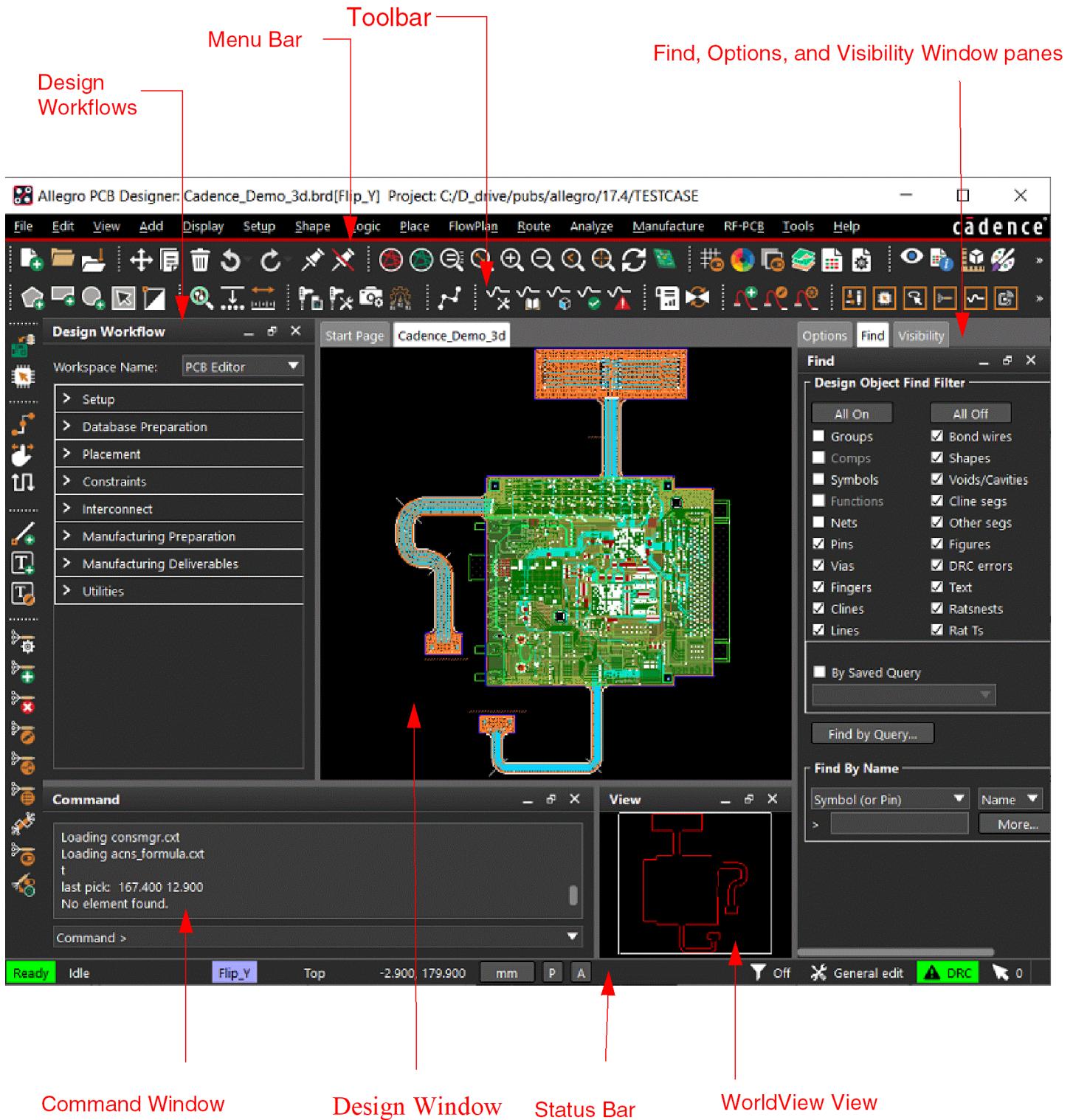


The Design Window

The Design Window is where you create a design.

When you are reviewing logs or reports using the Viewlog and Show Element commands, you can click on coordinate values within these files and zoom center on the corresponding locations in the design window. For additional information, see [viewlog](#) and [show element](#).

Figure 2.17: PCB Editor: User Interface



Panning the Design Window

You can remain at a zoomed-in view, and move the design window across a design in any direction.

You can pan a design using a mouse or arrow keys on the keyboard.

There are several ways to pan in a design:

1. Place the cursor in the editor window. Press and hold the middle mouse button down and slide the mouse to the left, right, up, and down.
2. Use the arrow keys on your keyboard to pan the design. To control the amount of panning using the arrow keys:
 - a. Select Setup – User Preferences
 - b. Select Display/Roam in the Categories section
 - c. Set a value for the *roaminc* environment variable and select *OK*

The Menu Bar

The pull-down menus in the menu bar provide all of the commands that you need to create or modify a design. The menu command sets (*Layout*, *Symbol*) that are available to you depend on the task that you are performing and the tool that you are running.

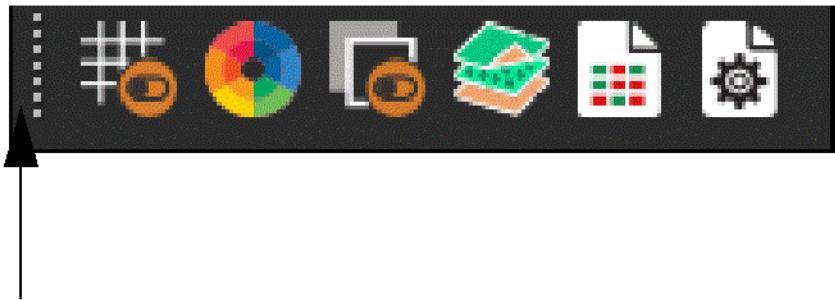
You can also use the accelerator key combinations to execute the command. The key combinations appear in the pull-down menu, to the right of the command.

The Toolbar

The toolbar contains functionally related icons, such as those for routing or placement, to access common commands. To learn a toolbar icon's function, position the cursor over the icon without depressing the mouse button and view its description in the tool tip that appears. Icons can be customized to suit specific needs.

Dock or undock any toolbar by left-clicking on the small circles, or grippers, next to it and moving it. The size of toolbar buttons are fixed and remain the same when the toolbar is expanded.

Figure 2.18: Grippers



Grippers

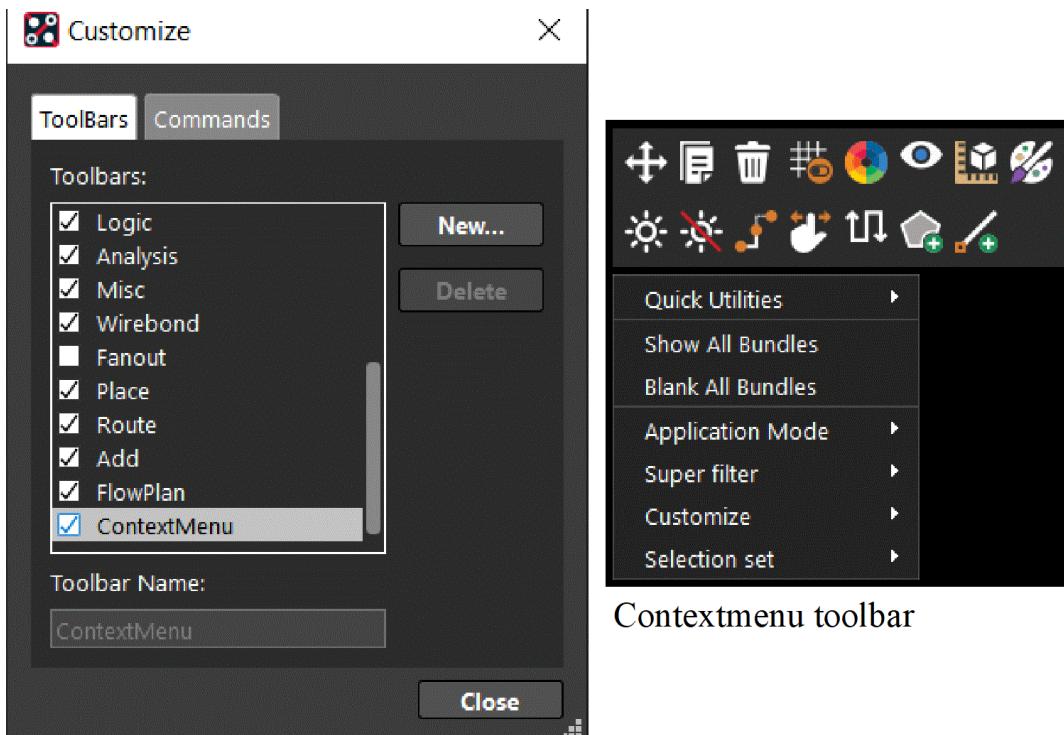
Customizing the Toolbar

You can customize the toolbar with icons or commands of your choice. New toolbars can be created and custom icons can be assigned to commands. Choose *View – Customize Toolbar* to open the Customize dialog box. A *ContextMenu* toolbar is available to assign frequently-used icons. This toolbar is available on right-click context menu. A maximum of 16 icons can be added.

⚠ The *ContextMenu* is not available in Linux.

In the *Toolbars* tab, checked the toolbars you want as part of your workspace.

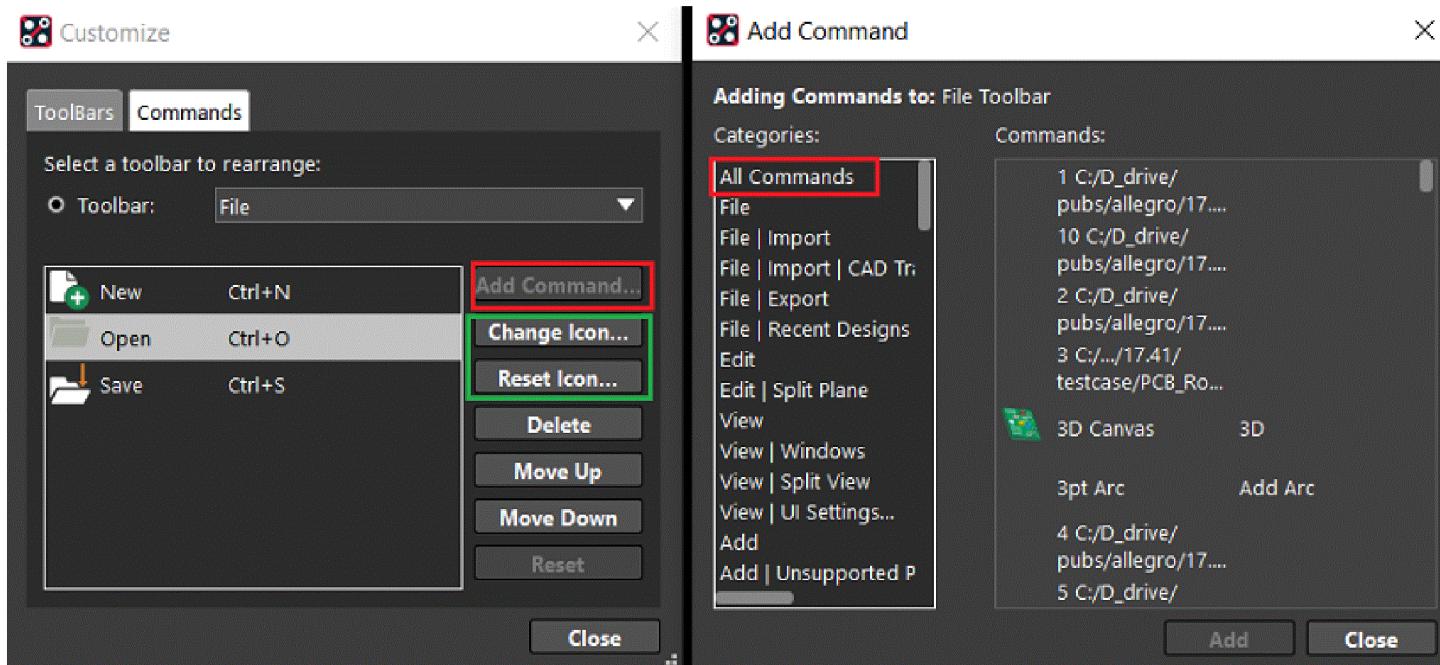
Figure 2.19: Customize Dialog Box for Toolbars



ContextMenu toolbar

In the *Commands* tab, customize individual toolbars. Select the toolbar and then click one of the buttons to add a new command (*Add Command*), delete an existing command (*Delete*), move a command up (*Move Up*), move a command down (*Move Down*) or reset the all changes (*Reset*). The *Add Command* button opens the Add Command dialog box, where you can select a category and a command under that category to add to a toolbar. A complete list of all the commands is available, which is sorted by command name, not by menu or submenu or order in menu.

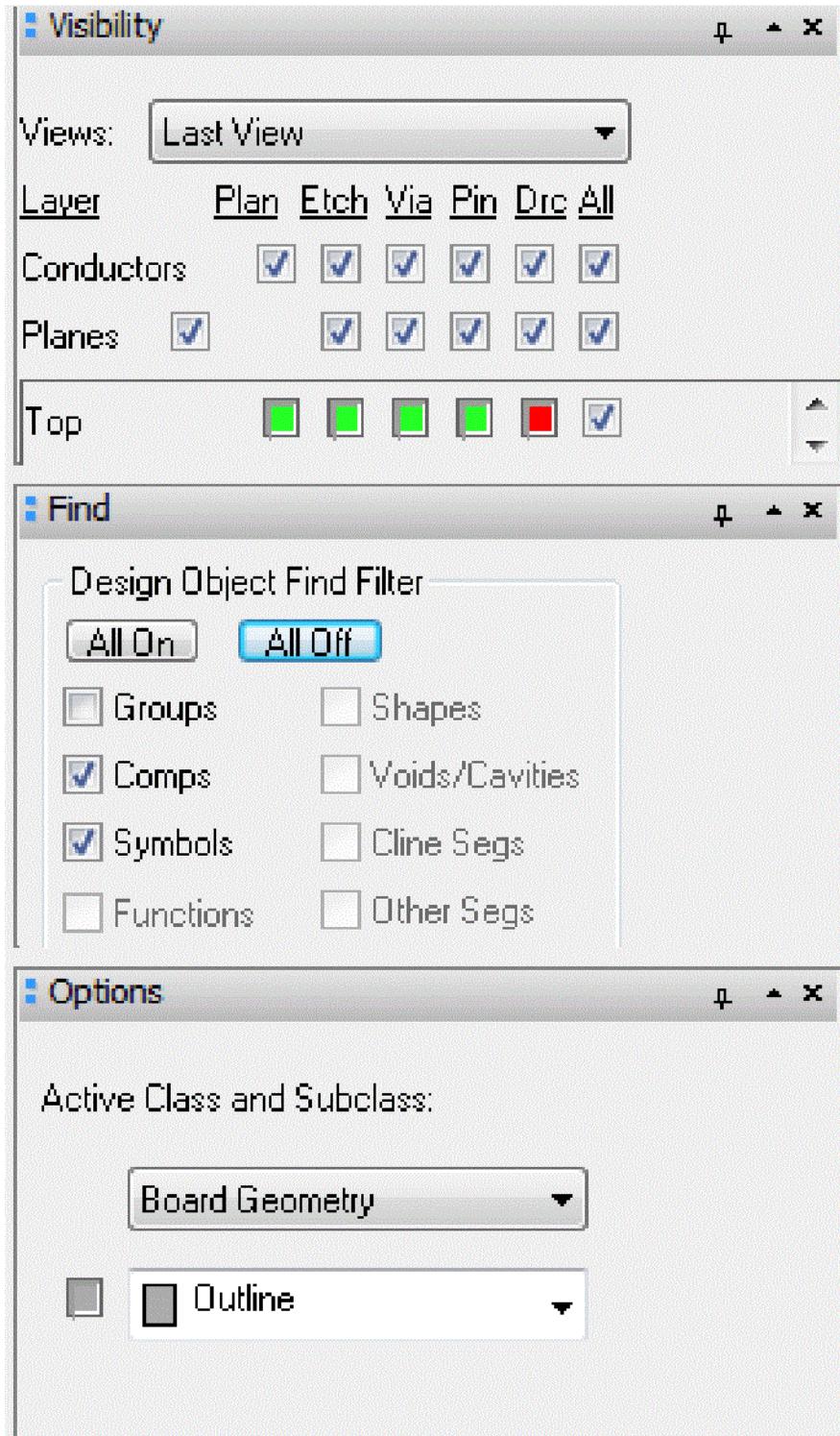
The *Change Icon* button opens a custom icon directory from where you can select a custom icon and assign to any existing or a new command. The path of the custom icon directory is set by the *iconpath* environment variable in User Preference Editor. The *Reset Icon* button restores the default icon applied.



The Control Panel

The Control Panel uses foldable *Options*, *Find*, and *Visibility* window panes that may be quickly resized or relocated to maximize the working design area. Using the pin icon, you can "pin" a window so it remains visible while unpinned windows remain as tabs bordering the design window.

Figure 2.20: Stackable Control Panel Window Pane



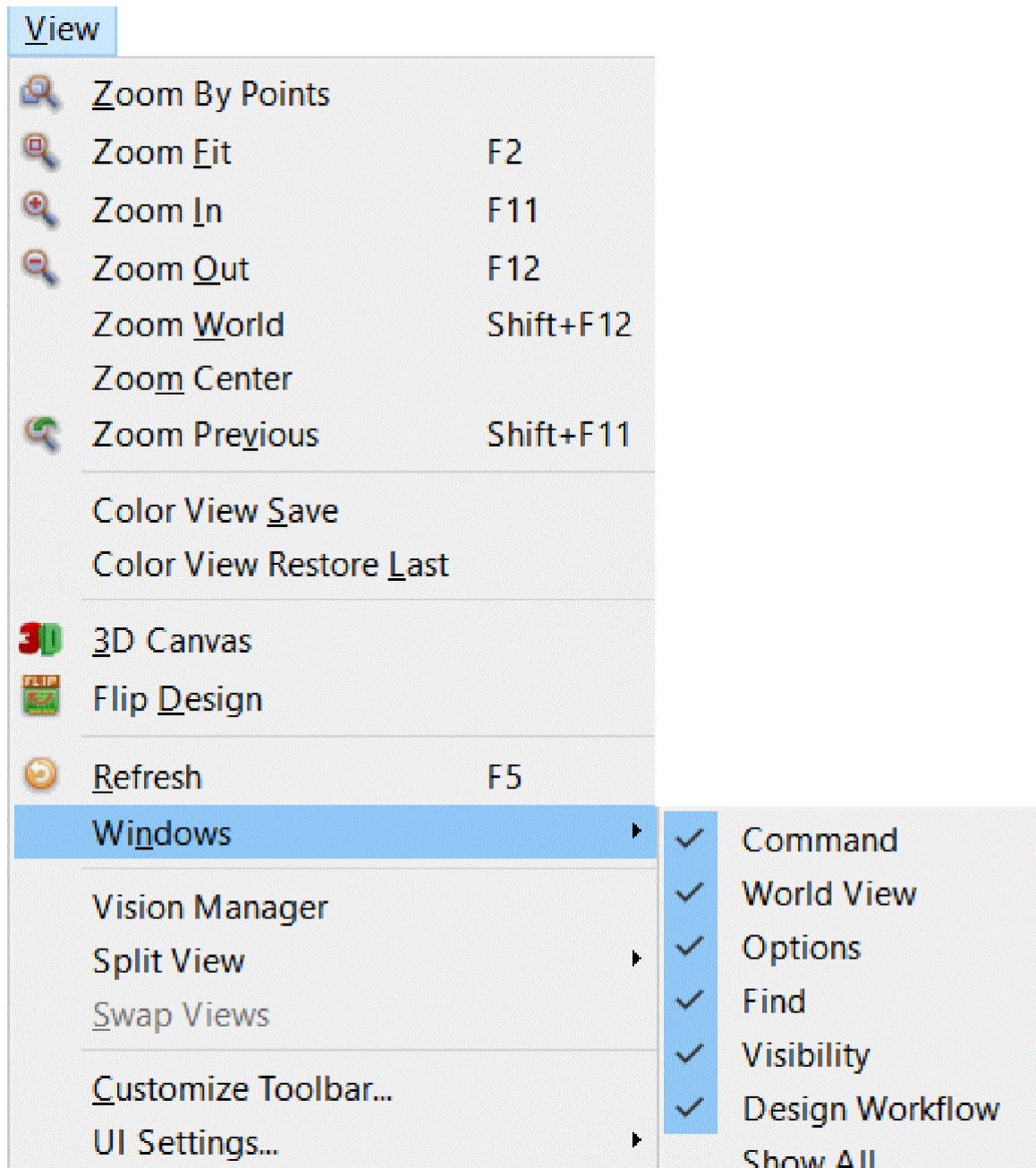
Working with Foldable Windows

The foldable windows are particularly useful on a single monitor setup because they provide more work space, while giving the designer the option of seeing the window-pane information by simply hovering over the tabs bordering the design window. Passing the cursor over any of them quickly unfolds the window pane for viewing or editing, then retracts it.

Click anywhere along the pane name and drag the pane to dock or undock. You can move the panes or windows anywhere within or outside the design window. The available docking area turns light blue. In a dual-monitor system, undocking windows are useful as they can be moved to the second monitor, maximizing the work space.

You control the visibility of these windows by clicking an arrow to expand a docked window pane, clicking the X to hide it, or by using the *View* menu choices to hide or display it.

Figure 2.21: View - Windows commands



The Options Window Pane

The *Options* window pane displays current parameters and values for the active command. Parameters that appear in the *Options* window pane differ according to the active command.

For a command that functions in a pre-select use model, parameters relevant to the command may also be set by right-clicking to display the pop-up menu from which you may choose:

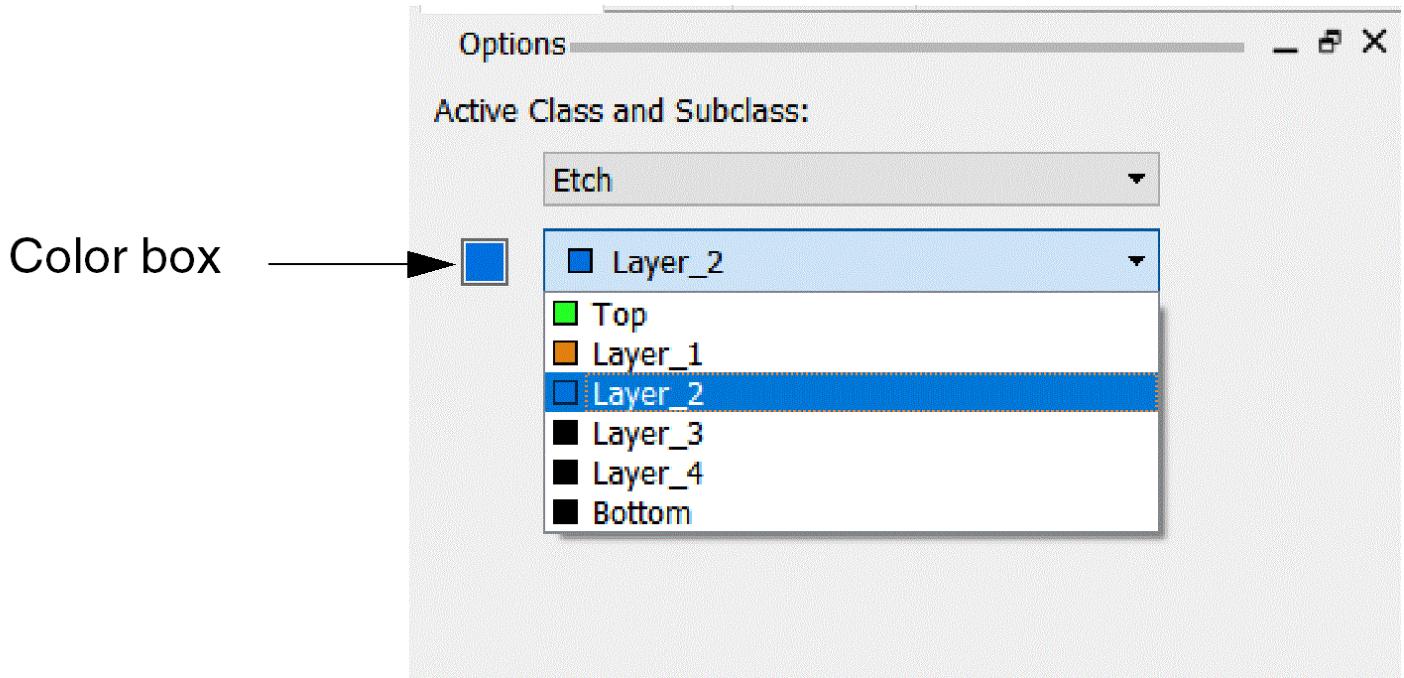
- *Design Parameters* to access the Design Parameter Editor (*Setup – Design Parameters* or `prmed` command)
- *Options*

If a command functioning in a pre-select use model has no parameters that must be set to use the command, *Options* does not appear on the pop-up menu. Changing a parameter using either of these pop-up menu choices automatically updates the *Options* window pane parameters as well.

Dock or undock the window by left-clicking to choose it and moving it anywhere within or outside the design window.

You control the window pane's visibility by clicking an arrow to expand a docked window pane, clicking the X to hide it, or by using *View – Windows – Options* to hide or display it.

Figure 2.22: Options Tab Window Pane (Pinned)



Active Class and Subclass Fields

When you choose a command, the *Options* window pane changes to reflect the appropriate class and the default subclass (the first subclass on the list for that class). For the ETCH/CONDUCTOR class, the subclasses are listed in the order that the layers appear on the design. For non-ETCH/CONDUCTOR classes, the subclasses are sorted alphabetically.

The color swatch to the left of the subclass field indicates the visibility status of the subclass in a design. When in *Visibility* pane, the subclass is enabled the swatch displays the color assigned to the subclass. When the subclass is disabled, the swatch displays the design's background color. You can control the display of a subclass using the *Visibility* pane or the Color Dialog. Choose *Display – Color/Visibility* (`color192` command), described in the *Allegro PCB and Package Physical Layout Command Reference*.

The parameters and values you set in the *Options* window pane take effect immediately and override definitions for the same parameters and values that may exist elsewhere in the tool. For example, the tool looks to the Design Parameter Editor for the rotation and text values. If a different value exists in the *Options* window pane, however, the tool ignores the information in the Design Parameter Editor dialog box.

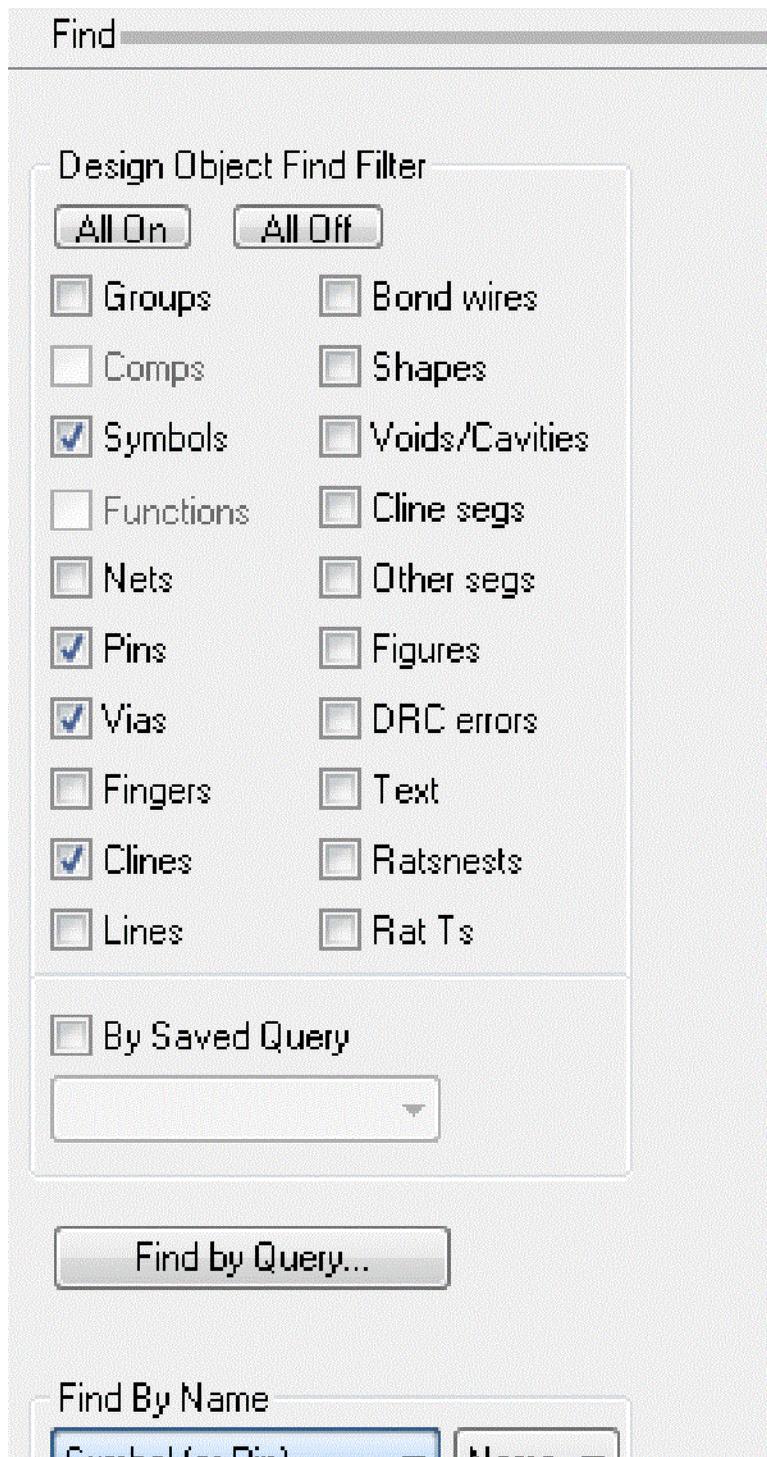
When you update values in the Design Parameter Editor, the values in the *Options* window pane change as well.

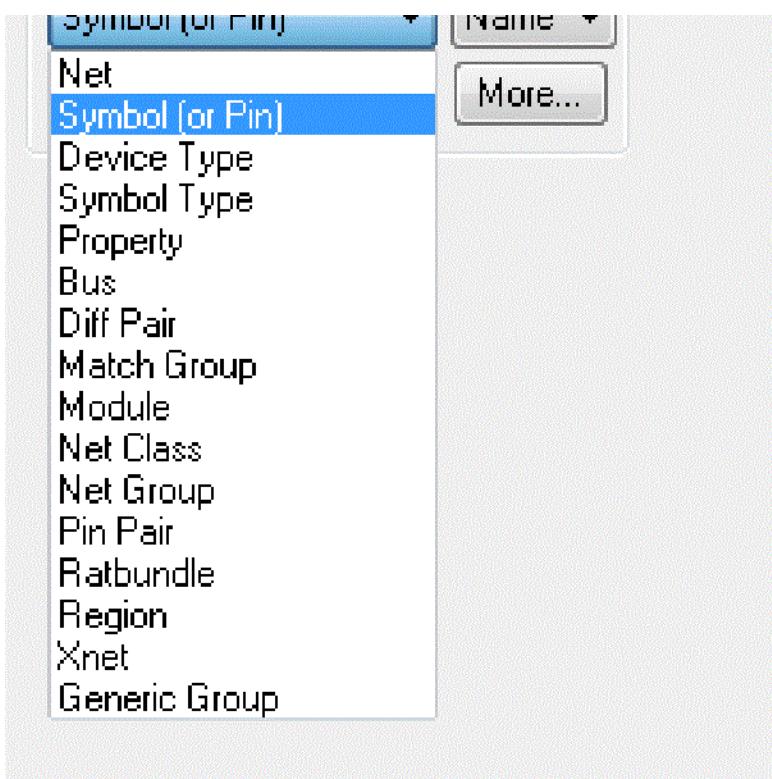
The Find Window Pane

The *Find* window pane lets you specify design elements the active command affects. When you run an interactive command, such as *Edit – Move* ([move](#) command), the *Find* window pane displays the elements the command requires.

To refine your selection set and confine your work to a particular element type, such as all nets, you can also right-click and choose the *Superfilter* temporarily to disable the *Find* window pane.

Figure 2.23: The Find Window Pane (Pinned)





The *Find by Name* section lets you choose elements by name, rather than graphically, or from a text file that contains a list of the names for the design objects. If you choose *Name* from the drop-down menu and click the *More* button, the *Find by Name* or *Property* dialog box appears displaying a list of all available names for the design object you chose.

If you change from *Name* to *List* and click the browse button, a browser window appears that lets you navigate to the directory that contains the specific list file you want.

When using either of these two methods, the layout editor ignores the check boxes in the *Design Object Find Filter* section, unless you use the *Property* pull-down option. For property search, *Find by Name* shows the properties available on displayed objects that are enabled on the *Find* filter.

The Visibility Window Pane

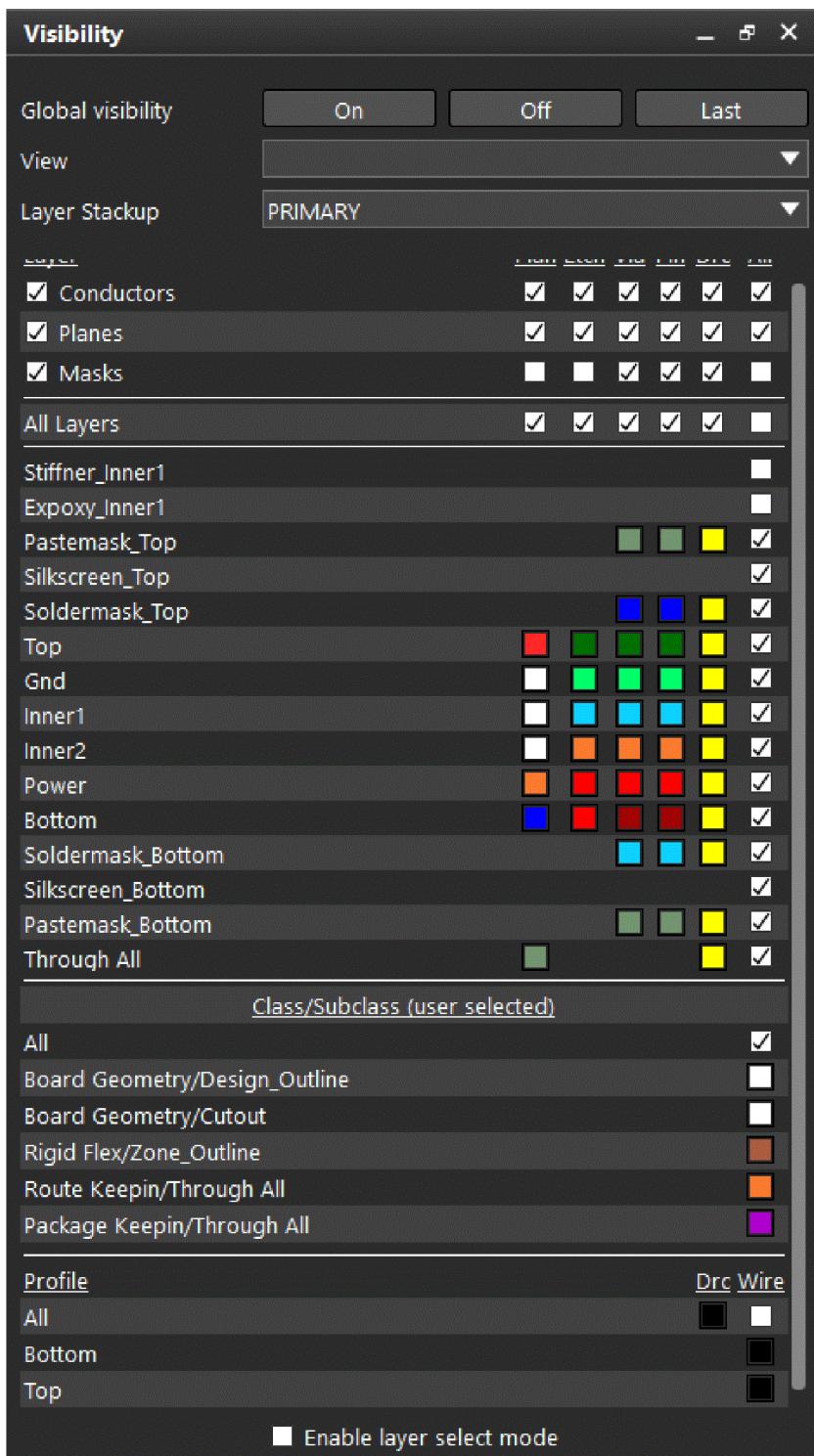
The *Visibility* window pane lets you selectively display or hide conductive elements in a design in a single and multi-stackup designs. The pane lists all design elements, namely, layers, classes/subclasses, profiles, and user-added subclasses present in the design. For example, a PCB layout might list *Conductors*, *Planes*, and *Masks* under *Layer*, whereas a package substrate layout might list *Conductors*, *Dielectrics*, and *Die Stacks*. Same is the case with the other listed items. For easy readability, the rows are colored alternatively.

Once you have assigned colors to each class of design element you can use the *Visibility* window pane to selectively display *ETCH/CONDUCTOR*, pins, and vias on each layer in the design. The *Visibility* window pane displays the color assigned to a design element when that element is visible, and displays the background color of the design window when the design element is invisible.

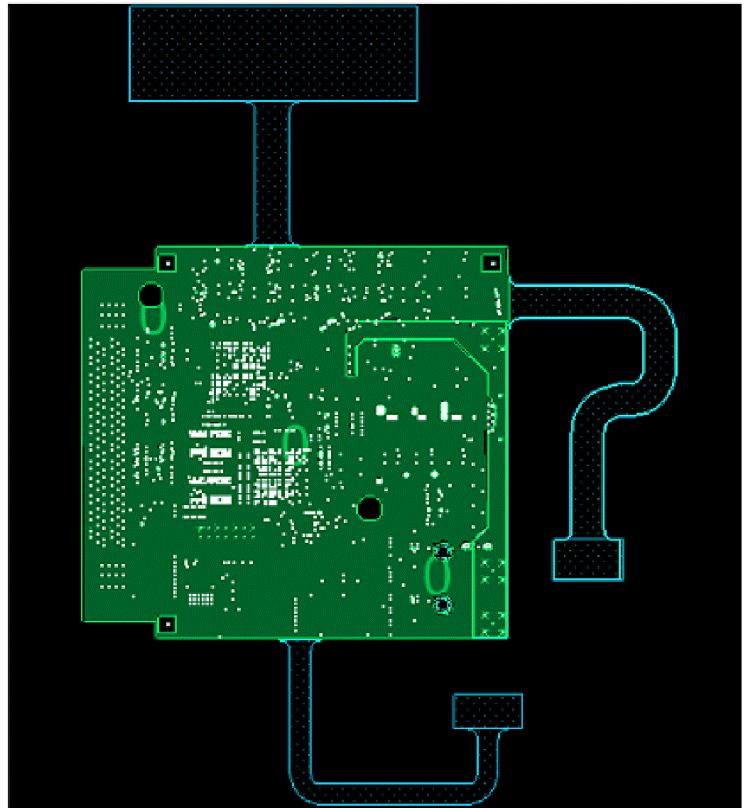
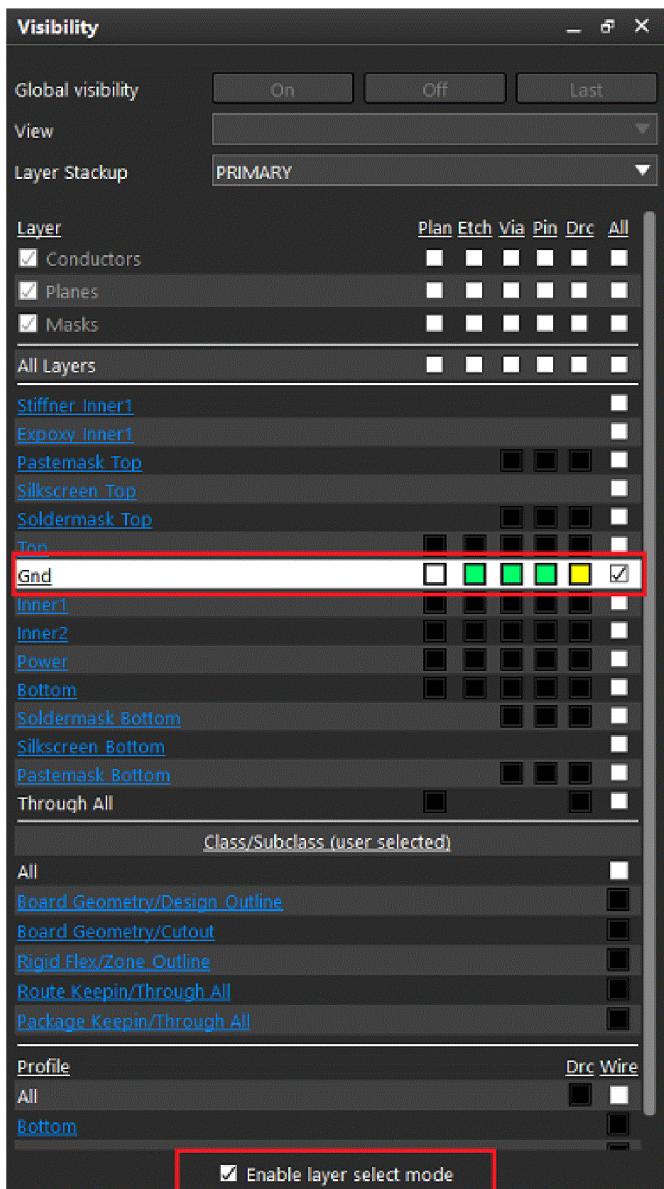
When the button displays the assigned color, visibility is enabled and the design element is visible. When the button displays the background color, visibility is disabled and the design element is hidden. You can quickly control the visibility of all layers by clicking the *All* button associated with the desired design element.

To delete plane layers in the *Visibility* window pane, click the *Planes* checkbox, a convenience if a design has a large number of layers that you might have to scroll through. You can set global visibility from the *Visibility* window pane. You can also turn on or off mask layers.

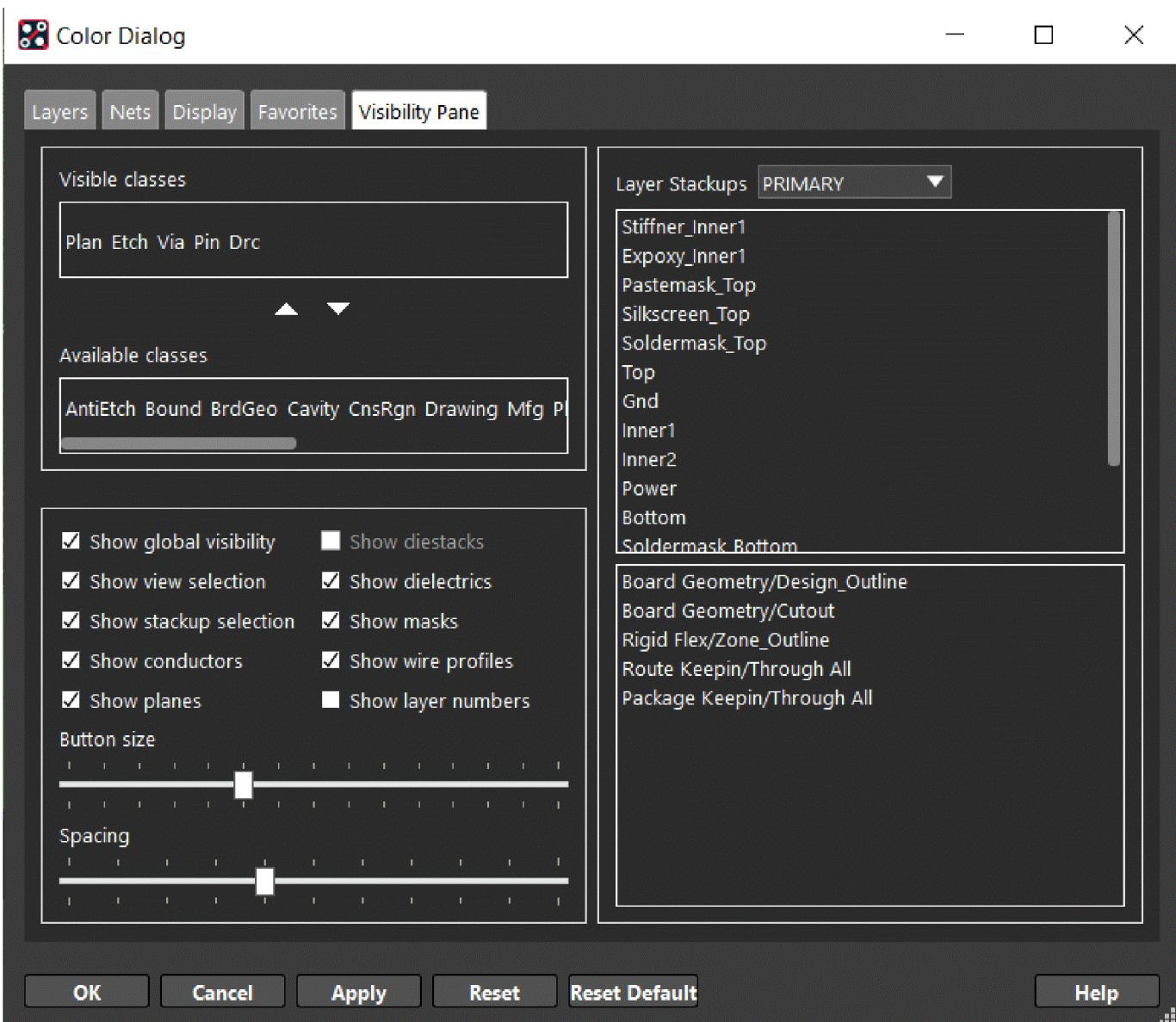
Figure 2.24: The Visibility Window Pane



The *Enable layer select mode* puts the canvas into a single layer view and lets you quickly view or scroll through each available layer. When this mode is enabled, the individual design elements (layers, classes/subclasses, and profiles) changes into active links in blue color that you can click to select. These links allow you to switch between layers. When in the layer select mode, any combination of multiple layers can be selected and viewed using **ctrl** key.



You can customize the Visibility window pane from the Color Visibility dialog box ([The Color Dialog Box](#)). You can enable and disable global visibility, stackup selection, layers, planes, and so on.



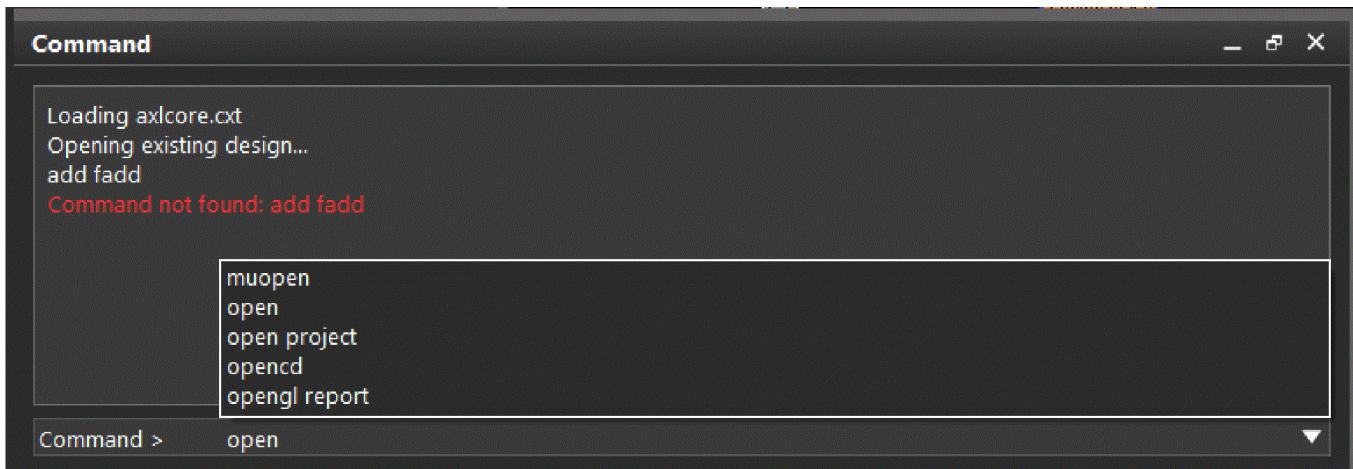
Layers types that do not exist in a design are automatically hidden in the *Visibility* plane even though they are enabled in the Color Dialog.

The Command Window

The Allegro GUI includes a command window that allows you to enter commands while also displaying messages and command output. The command window has two separate sections for input and output. You can increase or decrease the font size in these windows using a mouse with a wheel. Hold the **Ctrl** key on the keyboard and scroll the wheel up to increase the text size or scroll down to decrease the text size. The input section of the command window placed in bottom provides auto-completion of commands. The command line is editable similar to a text editor. The output section shows full history of recently-used commands and the messages. These messages follow a color scheme to easily detect their type:

- Black: Messages stating information are displayed in black color.
- Orange: Messages showing warnings are displayed in orange color.
- Red: Messages showing errors are displayed in red color.

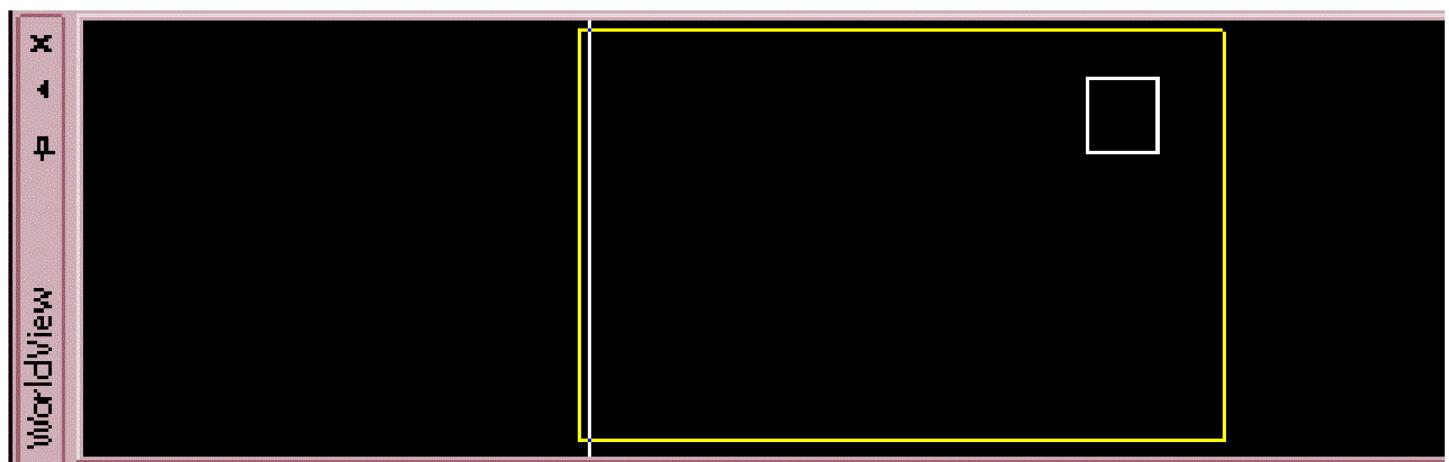
Figure 2.25: The Command Window



The WorldView Window

The *WorldView* window provides a bird's-eye view of your design. Using the *WorldView* window, you can zoom in to display a smaller area of the design outline or zoom out to display a larger area of the design. You can use the *WorldView* window alone with the *View* menu commands and accelerator keys.

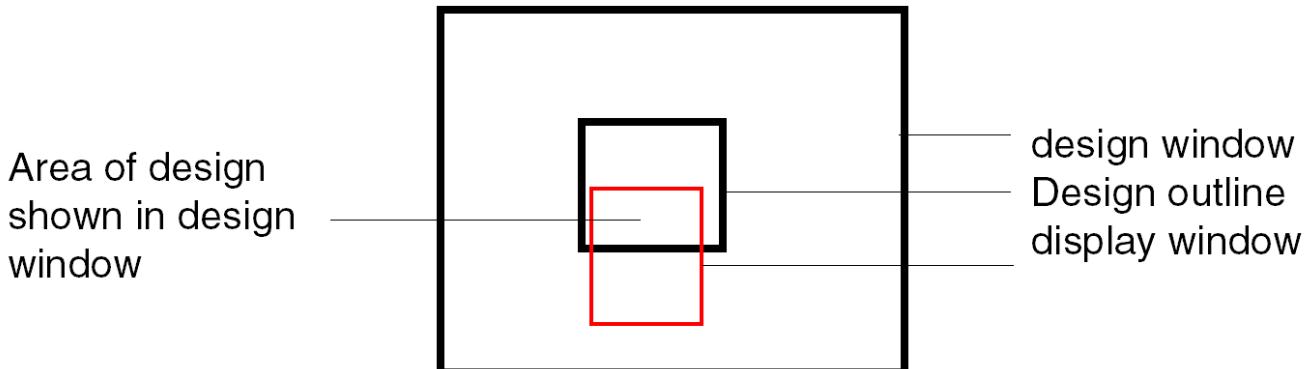
Figure 2.26: WorldView Window Pane



Using the WorldView Window Pane

There are three ways you can control the view of the design using the *WorldView* window:

- To display specific areas of the design
- To scroll through the design
- To zoom in or out of the design



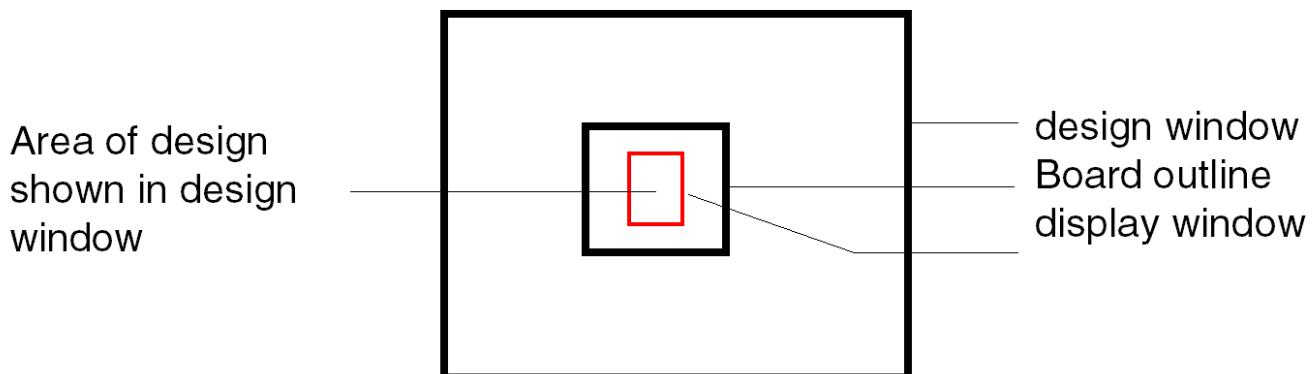
Displaying Specific Areas of a Design

To use the *WorldView* window to display specific areas of a design:

- In the *WorldView* window, left-click and drag-and-drop the display window over the area of the design that you want to display in the design window.

 If you are using a three-button UNIX mouse, the middle button gives you a greater degree of control when performing this operation.

If you size the display window over a small area of the outline (using the left button), the design window zooms in on that area.



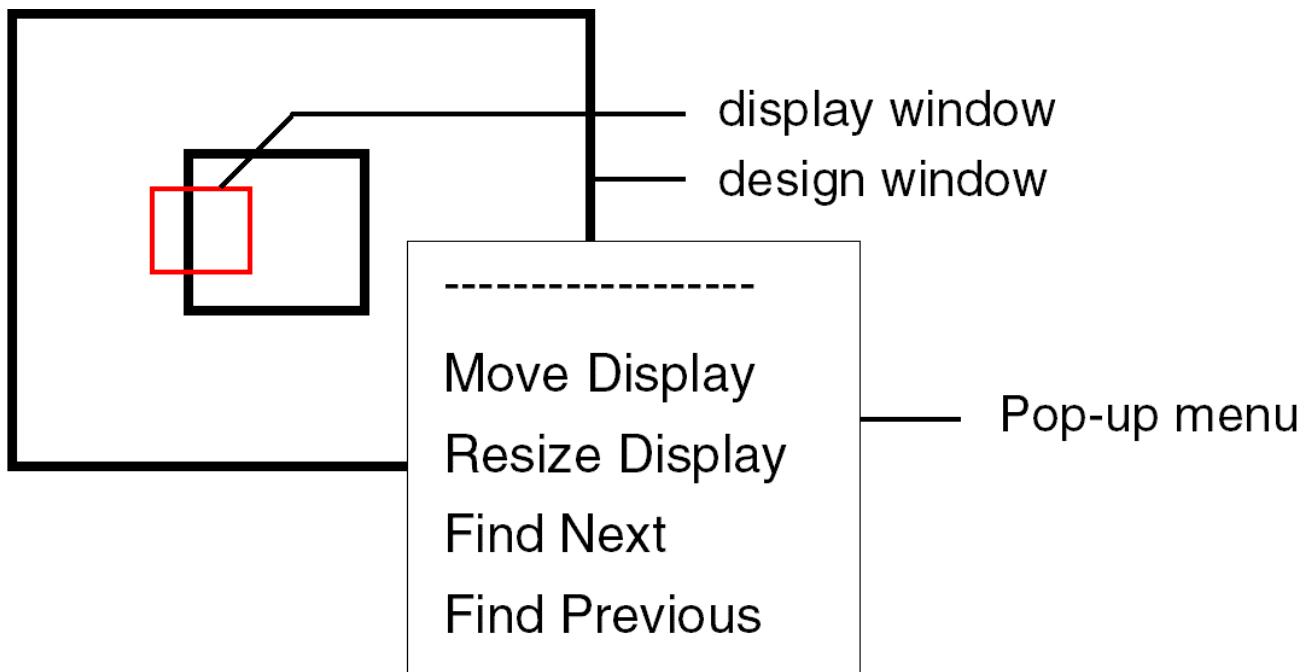
If you size the display window over a larger area of the outline (using the left button), the design window zooms out to display that area.

The WorldView Window Pop-Up Menu

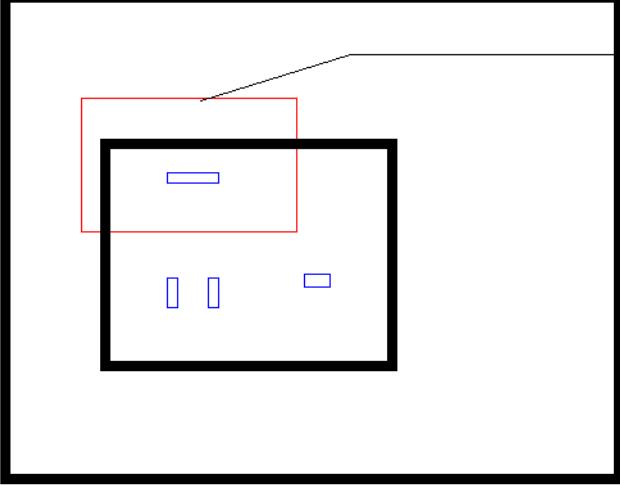
To display the *WorldView* window pop-up menu

- Right-click in the *WorldView* window.

The pop-up menu appears.



Following are descriptions of the options in the *WorldView* pop-up menu.

<i>Move Display</i>	lets you move the display window to select an area of a drawing for display in the design window.
<i>Resize Display</i>	zooms the design window on an area you define by selecting points in the <i>WorldView</i> window. You can also type <code>window center</code> at the console to perform the same function, but you then specify the new window area by selecting its center in the design window.
<i>Find Next</i>	advances through the list of any highlighted items, centering the display on each of them, in the order in which they were highlighted.  <div style="margin-left: 20px;"> display window centered over highlighted item </div>
<i>Find Previous</i>	reverses through the list of highlighted items.

You can continue choosing *Find Next* or *Find Previous* by left-clicking in the *WorldView* window. The click repeats the last command. *Find Next* is the default command in effect with a left click after new elements have been highlighted.

The command window identifies each element as you cycle through the highlighted items in the *WorldView* window. The > symbol indicates that you are advancing to the next element in the list whereas the < symbol indicates that you are advancing to the previous element. For example, after centering on a line with *Find Next*, the message is > *Line*.

The Status Bar

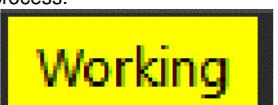
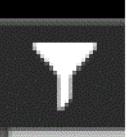
The Status bar shows the active command, subclass, application mode and number of selected objects. These coordinates change as you move the mouse over the canvas. You can also customize the status bar by right-clicking and selecting the options to show or hide the different panes.

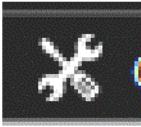
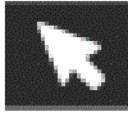
Status Bar Panes

Figure 2.27: Status Bar



The Status bar has the following elements:

<i>App status</i>		 Displays the current state of the application.  process.  Halts execution of the currently active process.	Ready to activate any command or processing action or command that cannot be cancelled. Processing action or command that cannot be cancelled.
Active command		Displays name of the current active command.	
<i>Compatibility Mode</i>		Displays the compatibility mode status, if set.	
Flipboard Mode		Displays the design flipped along Y axis.	
<i>Active Class/Subclass</i>		Indicates active class and subclass. Click this field to display a pop-up menu that lets you choose class and subclass. Switching a class/subclass from the status bar sets the <i>Active class and subclass</i> fields with the same values in the Options tab and makes the layers visible in the design canvas.	
Mouse XY Coordinate		Displays coordinates of the current location of the cursor	
Units		Displays design units. To change unit, click to open design tab of the Design Parameter Editor.	
Pick command		Lets you display a dialog box. When you click this button, and you are in an interactive command, for example, <i>add connect</i> , the Pick dialog box appears and remains displayed until you dismiss it. If the <i>Cmd</i> status is <i>Idle</i> , and you click the <i>P</i> button, the Zoom Center dialog box appears and remains displayed until you dismiss it. You can enter specific or incremental values in these dialog boxes. For additional information, see the Pick dialog box.	
<i>Toggle XY: Absolute/Relative</i>		Toggles the x, y read-out from absolute mode to relative mode. When you are in absolute mode, the x y coordinates location is from the origin of the design. When you are in relative mode, the origin is always from the last pick and the button is labeled <i>R</i> . The layout editor always starts designs in absolute mode.	
Aux/Script Text		Displays the name of the script playing.	
<i>Super Filter</i>		Lets you choose a particular element type to refine your selection set and temporarily disable all other elements from the right-mouse button pop-up menu rather than the <i>Find</i> window pane.	

App Mode		Displays the currently set application mode.
DRC Status		Empty display string indicates design rule checking is disabled.
		If the display string displays DRC it indicates that design rule checking is enabled. Right-click to access the following options: <ul style="list-style-type: none">• Enable On-Line DRC• Enable On-Line DFM checks• Update• Display Status A red color box indicates DRC is out of date or Batch DRC is required. A yellow color box indicates DRC is up to date, but DRC errors exist. A green color box indicates DRC is up to date and no DRC errors exist.
Selected Objects		Indicates the number of selected objects on the canvas. Click this field to display a pop-up menu with commands that operate on the currently selected element(s).  You can resize the status bar fields by setting the environment variable <code>resizable_status_bar</code> from the <i>General</i> category in the <i>Ui</i> section of the <i>User-Preference</i> dialog box.
Pulse Status		Indicates the status of layout editor connection with Pulse platform. Clicking the icon opens Allegro Pulse Status dialog. Three statuses are possible: <ul style="list-style-type: none">• Connection: <i>Disconnected</i>, <i>Connecting</i>, and <i>Connected</i>.• Health: <i>Maintenance mode</i>, <i>Normal</i>, and <i>Error</i>.• Pending Operations: <i>0</i>, or <i>N-many</i>  This icon is available only if Pulse is enabled.

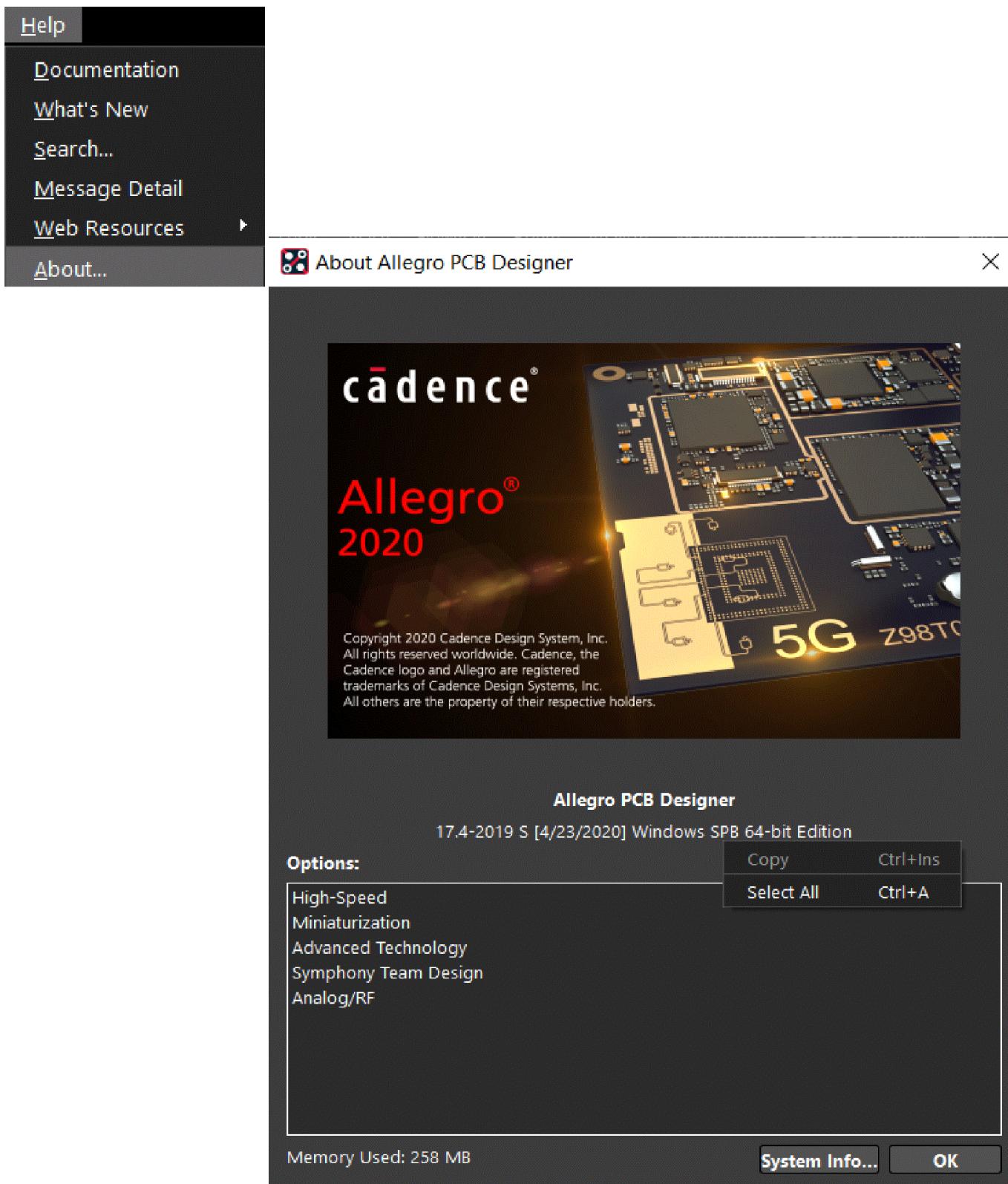
Customizing Status Bar

You can customize the status bar to show or hide the elements or panes. Right-click the status bar and change selection to show or hide the panes.

- ✓ App Status
- ✓ Active Command
- ✓ Compatibility Mode
- ✓ Extra 1
- ✓ Flipboard Mode
- ✓ Active Class/Subclass
- ✓ Mouse XY Coord
- ✓ Units
- ✓ Pick Command
- ✓ Toggle XY: Absolute/Relative
- ✓ Aux/Script Text
- ✓ Super Filter
- ✓ App Mode
- ✓ DRC Status
- ✓ Selected Objects

The About Window

To find out which version of layout editor you have, click *Help – About*. The following illustration displays the *About* dialog box that opens.



Following information can be viewed:

- Product name

- Complete version
- Product options
- Memory used

You can select the text within the *About* window for sharing information about layout editor. Pop-up commands *Select All* and *Copy* are also available to copy the necessary detail.

To view system-level information, click *System Info* button.

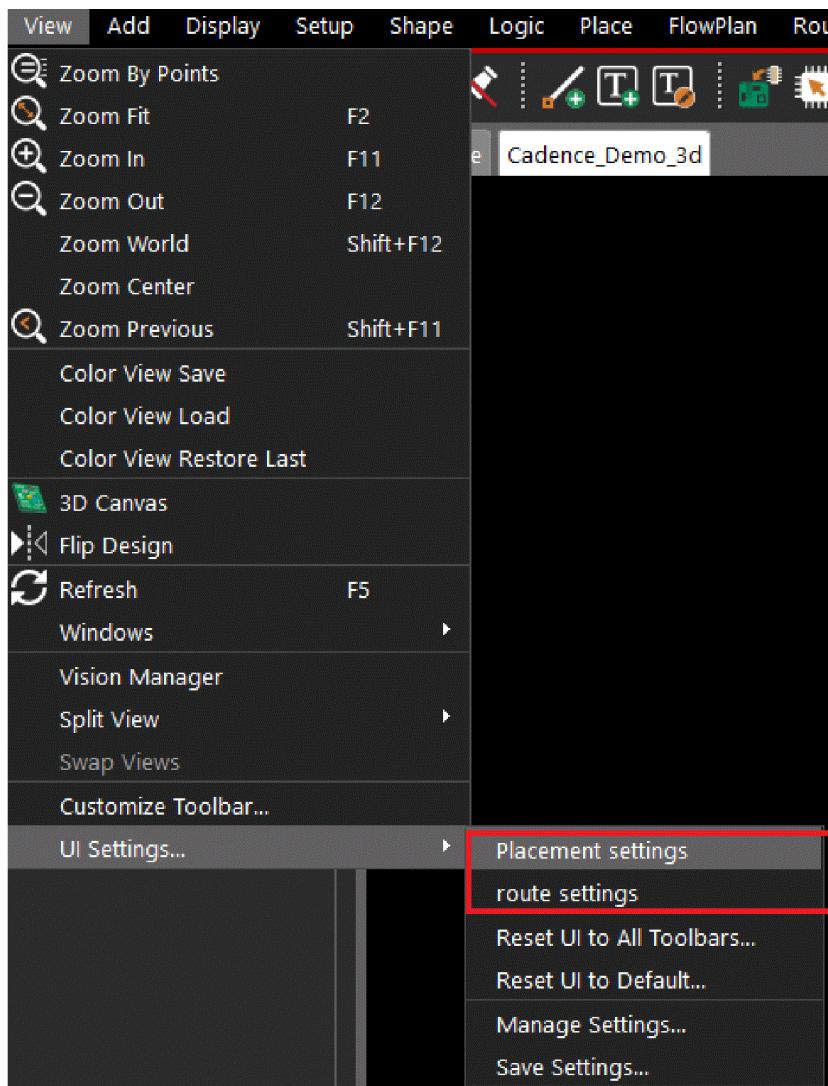
Customizing Design Canvas

Layout editors, by default, shows a minimalistic set of toolbars, icons and pre-defined size and positions of docking windows. The arrangement of toolbars and icons, placement and size of docking panes can be further customized and saved per design requirements. Multiple views can be created, saved and recalled when required. You can also export and import customized settings across systems. These settings can be saved at site level.

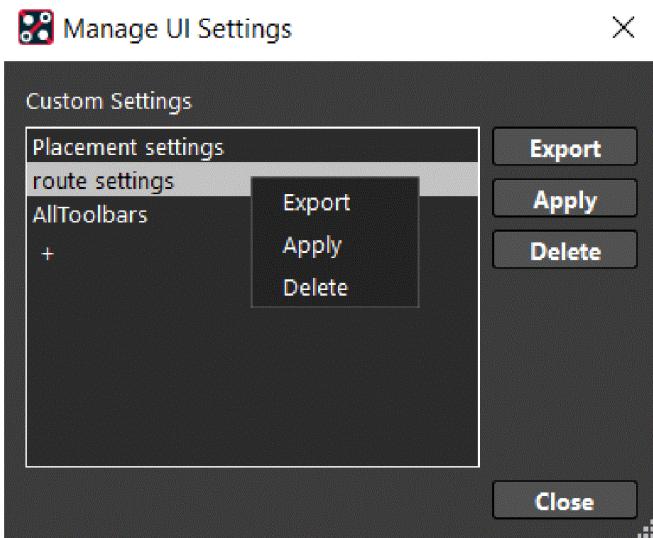
To save a user-defined setting, choose *View – UI Settings – Save Settings*, enter a name and click *Save*.



The setting adds to *UI Settings* menu. The user-defined settings becomes available in the menu and can be selected and applied to the active database.



To export, import, or delete any setting choose *View – UI Settings – Manage Settings*.



A custom setting on export is saved as a configuration (.ini) file in the <HOME>/PCBENV directory. To import an already saved custom setting, click + icon and browse to the location of configuration file (.ini). The selected custom setting becomes available to apply.

To restore the legacy (pre 17.4 release) all toolbars icons view, you can choose *Reset UI to All toolbars* menu. This command brings back default toolbar settings available in the installation directory. It also checks for any site-level UI settings saved at the path specified for the MENUPATH environment variable.

You can also access default configuration settings from the Manage UI Settings. In this window, click + icon and browse to the location of `AllToolbars.ini` file. This file is located at the <installation_directory>/share pcb/text. A new entry `AllToolbars` gets added to the list of *Custom settings*. Select the setting and click the *Apply* button.

The customized configuration can be saved and used as default. Set the operating-system variable CDS_SITE, which overrides the default site location:

```
<cdsroot>/share/local
```

See [Site Customization](#), which describes the options available with CDS_SITE.

You can also use *Reset UI to Default* menu to go back to the conventional window and toolbar positions and display settings.

Padstack Designer

The Padstack Designer lets you create or edit library padstacks, including:

- Defining the parameters of padstacks
- Creating blind and buried via padstacks
- Adding padstack layers
- Copying padstack layers
- Deleting layers in a padstack

A library padstack defines pad data for all layers. You must define padstacks before you create any package symbols, because each pin in a package symbol must have an associated padstack.

When you double-click the Pad Designer icon (in Windows) or type `pad_designer` at the UNIX system prompt, the Padstack Designer appears.

For information on the Padstack Editor, see "[Using the Padstack Designer](#)" in "[Library Padstacks](#)".

Maintaining Databases

The DBDoctor program checks the database for errors and other problems and reports them as they occur. DBDoctor supports .brd, .mcm, ..mdd, .psm, .dra, .pad, .sav, and .scf databases. DBDoctor can:

- Analyze and fix database problems.
- Eliminate duplicate vias.

- Perform batch design rule checking (DRC).
- Uprev databases more than one revision old.

Running DBDoctor

To verify the integrity of a drawing database at any time during the design cycle, run DBDoctor at regular intervals but always after completing a design and prior to creating an artwork file. For specific procedures, see *Tools – Database Update (dbdoctor command)* in the Allegro PCB and Package Physical Layout Command Reference.

You can run DBDoctor to verify work in progress, or from a terminal window outside the layout editor, perhaps to check multiple input designs in batch mode by using wildcards and various switches. You do not have to run the layout editor to use DBDoctor.

During processing, DBDoctor generates `dbdoctor.log`, which records check summaries and detailed information on records that contain errors, as well as names of symbols and nets and x.y coordinate information. If DBDoctor finds an error, then it adds the `dbdoctor.log` to the design as an attachment. The layout editor only saves the log file from the last run of DBDoctor that found an error.

DBDoctor uses the input file name by default and copies it as `<boardname>.brd.orig` or `<>.mcm.orig` in the same directory, thereby permitting you use wild cards. If you use wildcards with the input file, then each design you enter is copied under `<boardname>.brd.orig` or `<>.mcm.orig` unless you choose the *No Backup* field on the dialog box that appears when you launch DBDoctor externally or use the `-no_backup` switch, in which case, the tool replaces the original design.

Partial Versus Full Database Consistency Checks on Saving

When you save a design, the tool executes a partial database consistency check by default, in essence, a quick check.

The `dbsave_full_check` environment variable indicates to the database save utility when to do a full check rather than a quick check. A number of 1 or 0 specifies that each time a design is saved, execute a full check. If you set the variable to 100, then every 100 checks a full check occurs.

For example, to set the `dbsave_full_check` environment variable to do a full check every five saves, at the console window prompt, type:

```
set dbsave_full_check = 5
```

If the tool detects errors, it saves the file as `<name>.SAV`.

 A full database check may considerably lengthen the time required to save large databases.

 For information on opening a design saved in a previous version of the layout editor in the current version, see [Uprevving](#).

Database Revision Support

The oldest database revision support for uprev on a platform depends on when the layout editor initially supported that platform. The following table lists the older database that you can uprev.

Platform	Allegro Physical Database
Linux	14.0
Windows	11.0

Databases older than 11.0 require you to maintain 16.6 on a Sparc Workstation to update the design to revision 16.6 before accessing in on Windows or Linux.

Upreving Differential Pairs from Release 14.x to Release 15.x

When upreving a Release 14.x database to Release 15.x, the layout editor shifts the differential pair primary gap from the spacing rule set to the physical rule set assigned to the differential pair.

Since you can associate a physical rule set with nets tied to different spacing rule sets, the tool takes the value of the new 15.x gap from the first instance of the differential pair information found.

Differential Pair Log

When you uprev a design containing differential pairs, any problems with migrating the differential pairs appear in the `uprev_diffpair.log`, which you can scan using *File – Viewlog (viewlog command)*, described in the *Allegro PCB and Package Physical Layout Command Reference*. The tool only creates the log if problems occur.

The `uprev_diffpair.log` file lists the discrepancies for all other nets that share the physical rule but had different spacings for the differential pair.

These warnings are a guide that you can use in recreating differential pair constraints through ECsets or new physical rule sets in Release 15.x. You can set the gap values to the original values once data is moved into Release 15.x.

Information appears in the `uprev_diffpair.log` in this format:

```
Warning: Already seeded gaps for the physical rule PAIRS found.
```

```
DPI requires new physical rules.
```

```
Original Primary gap on TOP was 8.0.
```

```
Original Primary gap on BOTTOM was 10.0.
```

Additional information that cannot translate to Release 15.x rules occurs when Release 14.x databases contain differential pair data specific to spacing rule sets.

Since constraint areas no longer apply to differential pairs, you should carefully review the differential pairs in Release 15.x. Updating the DRC, in this case, shows problem areas within constraint areas. You can then apply the smallest gap spacing found in constraint areas for differential pairs to the new physical constraint value for *DiffPair neck gap* in the appropriate constraint set for the differential pairs.

Also, data uprevved to Release 15.x has spacing rule sets that you may not need. You can delete them if they only apply to differential pairs.

 Cadence recommends recreating differential pair constraints at the differential pair object level rather than on individual nets.

For additional information, see the *Creating Design Rules* user guide in your documentation set.

Removing the DIFFERENTIAL_PAIR Property

During the uprev process, the layout editor removes the DIFFERENTIAL_PAIR property (obsolete in 15.x releases) from the nets in the pair and places the nets in a differential pair group object. The object group name is the same as the property value. Differential pairs appear in the Assign Differential Pair dialog box, available by choosing *Logic – Assign Differential Pair* ([diff pairs](#) command).

If more than two nets have the same DIFFERENTIAL_PAIR property value, the tool randomly uses two of the nets to create the differential pair group. It skips the remaining nets, and a warning appears in the `uprev_diffpair.log`.

Converting Spacing Constraints

The differential pair spacing constraints, which are now electrical constraints, convert as shown in the following table:

14.x Diffair Spacing Constraint	Converted to this 15.x Electrical Property	Notes
Length Tolerance	DIFFP_PHASE_TOL	The DIFFP_PHASE_TOL property replaces the old DIFFP_LENGTH_TOL property. Delay percentage is no longer supported.
Primary Max Sep	DIFFP_PRIMARY_GAP	
Secondary Max Sep	—	This constraint is obsolete.
Secondary Length	DIFFP_UNCOUPLED_LENGTH	The DIFFP_UNCOUPLED_LENGTH property replaces the old DIFFP_2ND_LENGTH property.

In the case where a property on the net overrides an old spacing constraint, the most conservative value (the lowest value) converts to the new electrical property.

For those instances when nets have different values assigned to their differential pair constraints, including any assignments for constraint areas in the Spacing Rule Set Assignment Table, the most conservative value converts to the new electrical property for both nets. This is true, even when the value is zero.

 The layout editor flags any converted properties that result in a value of zero for the 15.x property in the `uprev_diffpair.log` file.

Differential pair properties placed on nets automatically bubble up to the differential pair group. The 14.x spacing constraint set name is kept on the nets, along with any non-differential pair constraints. The tool does not create a new electrical constraint set containing the new electrical constraints for the nets. Consequently, during uprev the same properties connect to each net in the pair, through the differential pair group.

Converting a DRC Mode

The DRC modes for the 14.x *Length Tolerance* and *Secondary Length (Max Len over Prim Sep)* spacing constraints on the nets convert into one 15.x DRC mode for the pair called *All Differential pair checks*.

If the 14.x modes differ, the layout editor assigns the mode based on this order of precedence: *Always/On, Batch, Never/Off*.

The tool converts the modes as follows:

14.x DRC Modes	Converted to this 15.x DRC Mode
Any mode is set to <i>Always</i>	<i>On</i>
A variety of <i>Batch</i> and <i>Never</i> settings	<i>Batch</i>

Converting Environment Variables

The *drc_diff_pair_override* and *drc_diff_pair_primary_separation_tolerance* environment variables are retained in Release 15.x only for uprevving purposes. You can no longer set these variables. During migration, they convert to new DIFFP_COUPLED_PLUS and DIFFP_COUPLED_MINUS electrical properties that define the coupling tolerances around the primary gap for the differential pair. For details about these properties, see [DIFFP_COUPLED_PLUS](#) and [DIFFP_COUPLED_MINUS](#) in the *Allegro Platform Properties Reference*.

The layout editor converts the *drc_diff_pair_override* environment variable as follows:

14.x drc_diff_pair_override Value	Converted to these 15.x Properties*
0 or blank	Nothing done
100	DIFFP_COUPLED_PLUS = 1 DIFFP_COUPLED_MINUS = 1
200	DIFFP_COUPLED_PLUS = 2 DIFFP_COUPLED_MINUS = 2

*These values are in database units using the specified accuracy (both settings are in the *Design* tab of the *Design Parameter Editor*). Use *Setup – Design Parameters* ([prmed](#) command) to access the Design Parameter Editor. For example, for a *drc_diff_pair_override* value of 100, if the *User Units* are [mils](#) and the *Accuracy* is 2, these become the 15.x property values:

DIFFP_COUPLED_PLUS = 0.01 MIL

DIFFP_COUPLED_MINUS = 0.01 MIL

The *drc_diff_pair_primary_separation_tolerance* environment variable can specify optional minimum and maximum values. The tool converts these values in the following ways:

14.x drc_diff_pair_primary_separation_tolerance Values	Converted to these 15.x properties
blank	Nothing done
minimum value specified (example: 10 mil)	DIFFP_COUPLED_MINUS = 10 MIL
maximum value specified (example: 20 mil)	DIFFP_COUPLED_PLUS = 20 MIL

⚠ If both the *drc_diff_pair_override* and *drc_diff_pair_primary_separation_tolerance* environment variables are set, The tool only converts the *drc_diff_pair_primary_separation_tolerance*.

Setting Up a UNIX Environment

The tool set operates in a windows environment on UNIX workstations. UNIX contains two major shell families: [csh](#) and [sh](#). The [csh](#) family includes [tcsh](#) while the [sh](#) includes [ksh](#) and [bash](#).

Depending on default login shell you need to perform either of the following steps described in the [csh](#) or [sh](#) sections to access the Cadence Allegro tools.

To identify login shell, type following command in a terminal:

```
echo $SHELL
```

Using csh environment to access Cadence Allegro tool set

If you are working in a C shell, you must source the `.cshrc` file to initialize your environment before starting your tool. You can do this in two ways:

- Source the `cshrc` file each time you start the layout editor.
- Copy the contents of the `cshrc` file into your own `.cshrc` file.

To source the `cshrc` file:

- At an operating-system prompt, type

```
source <install_dir>/tools/bin/allegro_cshrc
```

The `install_dir` is the directory in which the layout editor was installed.

⚠️ In release 17.0 location and name of the Cadence provided `cshrc` file has changed. It has been updated to add a default Cadence PATH variable. The presence of Sigrity tools may require an additional location.

Copying the contents of the `cshrc` file into your own `.cshrc` file

1. To install it in your `.cshrc` file, at an operating-system prompt, type:

```
source <install_dir>/tools/bin/allegro_cshrc
```

Replace `<install_dir>` with the root location of the Allegro release.

⚠️ This file is normally hidden. To view the file, in your home directory type following in a terminal

```
ls -a
```

Using sh/ksh Environment to access Cadence Allegro

If you are working in a korn shell, you must incorporate the layout editor's `profile` file into your environment before starting the tool. You can do this in two ways:

- Source the `profile` file each time you start the tool.
- Copy the contents of the `profile` file into your own `.profile` file.

To incorporate the `profile` file into your korn shell environment:

- At an operating-system prompt, type:

```
. <install_dir>/tools/pcb/bin/allegro_profile
```

The `install_dir` is the directory in which the layout editor was installed.

⚠️ In release 17.0 location and name of the Cadence provided `profile` file has changed. It has been updated to add a default Cadence PATH variable. The presence of Sigrity tools may require an additional location.

Copying the contents of the `profile` file into your own `.profile` file

1. To install it in your `.profile` file, at an operating-system prompt, type:

```
source <install_dir>/tools/bin/allegro_profile
```

Replace `<install_dir>` with the root location of the Allegro release.

⚠️ This file is normally hidden. To view the file, in your home directory type following in a terminal

```
ls -a
```

Starting the Layout Editor from an Operating-System Prompt

When you start a tool from the operating-system prompt, you have the following options:

- Type an Allegro command and do not include a drawing name, or include the name of a new design.
- Type an Allegro command and include the name of an existing padstack, symbol, or layout (without an extension) to be opened.

To start the layout editor from an operating-system prompt:

- Type one of the following Allegro commands:

To open padstacks:

```
padstack_editor [-s  
script ][-p startdir ]<padstack (.pad) filename >
```

The arguments for the Allegro command `allegro` are

<code>- product product_name</code>	Determines the product tier that is run.
<code>-s script_name</code>	Runs a specified script file.
<code>-o journal -j journal</code>	[Default] Starts a journal file that records your Allegro PCB Editor work session. The name of the file is <code><program>.jrl</code> .
<code>-p start_directory</code>	Lets you specify a startup directory. If you start the layout editor with a drawing name that includes a path to the drawing (for example, <code>/home/joe/pcb/designs/layout_name</code> (.brd or .mcm), other files created during processing (.log and .jrl files) are created in the directory you specified and not the directory in which the drawing is located.
<code>filename</code>	Specifies a design file. You do not have to include the file type (extension).
<code>-product license_filename</code>	Starts the product based upon the name of the product license file.
<code>-proj cpm_file</code>	Reads the HDL-indicated .cpm file at startup.
<code>-mpsXXX</code>	Standard Cadence mps argument support (This is not typically required.)
<code>database_name</code>	Starts the product with the indicated database name.
<code>-version</code>	Prints the version of the product, then exits.
<code>-nographic</code>	Runs the layout editor in a non-graphic mode but still requires an X server. UNIX operating systems only.

For example:

```
allegro -product <  
product_name  
> -s <  
scriptfile>  
<  
filename>
```

```
apd -product <  
product_name  
> -s <  
scriptfile>  
<  
filename>
```

If you do not include a design name, the tool displays the editor you selected and opens a default file called `unnamed.pad`, `unnamed.dra`, `unnamed.brd`, or `unnamed.mcm`. You can then use the `open` or `new` command to open an existing or new drawing from the user interface.

If you have previously opened sessions of the layout editor, the last saved design in the previous session opens, based on information written to the `master.tag` file.

The `master.tag` file is a text file automatically generated when you launch a session of your layout editor. The file contains the name of the last database that you saved before ending a session. The tool reads this file when you next launch a session and opens the database of that name.

If, for any reason, you do not want the tool to open to the last saved database, you can move or delete the `master.tag` file. The tool then opens a new, unnamed design file. To locate `master.tag`, open the initialization (`.ini`) file, located in your `pcbenv` directory. Search on `directory=` to locate the file.

Starting Allegro PCB Editor

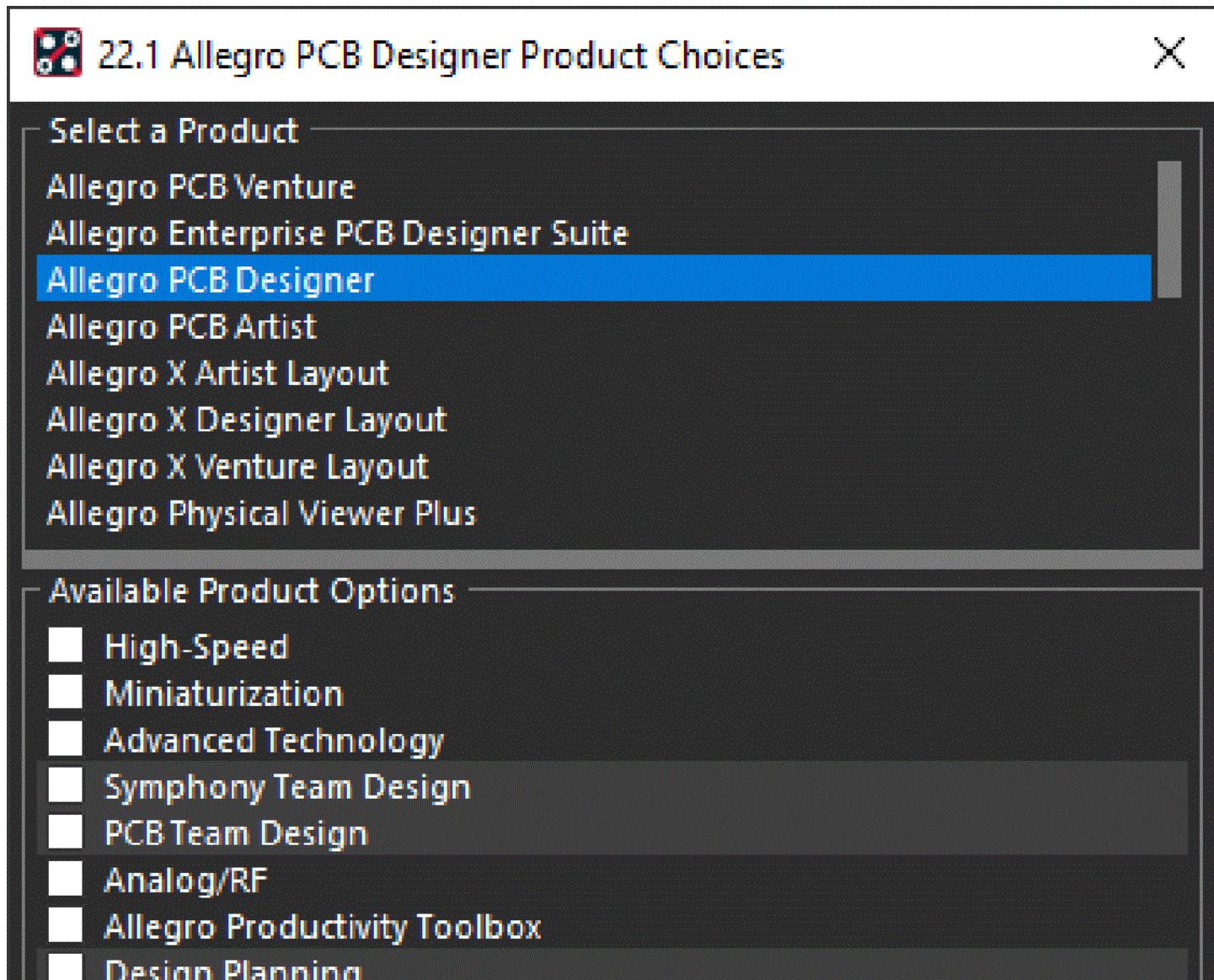
You can start Allegro PCB Editor in one of the following ways:

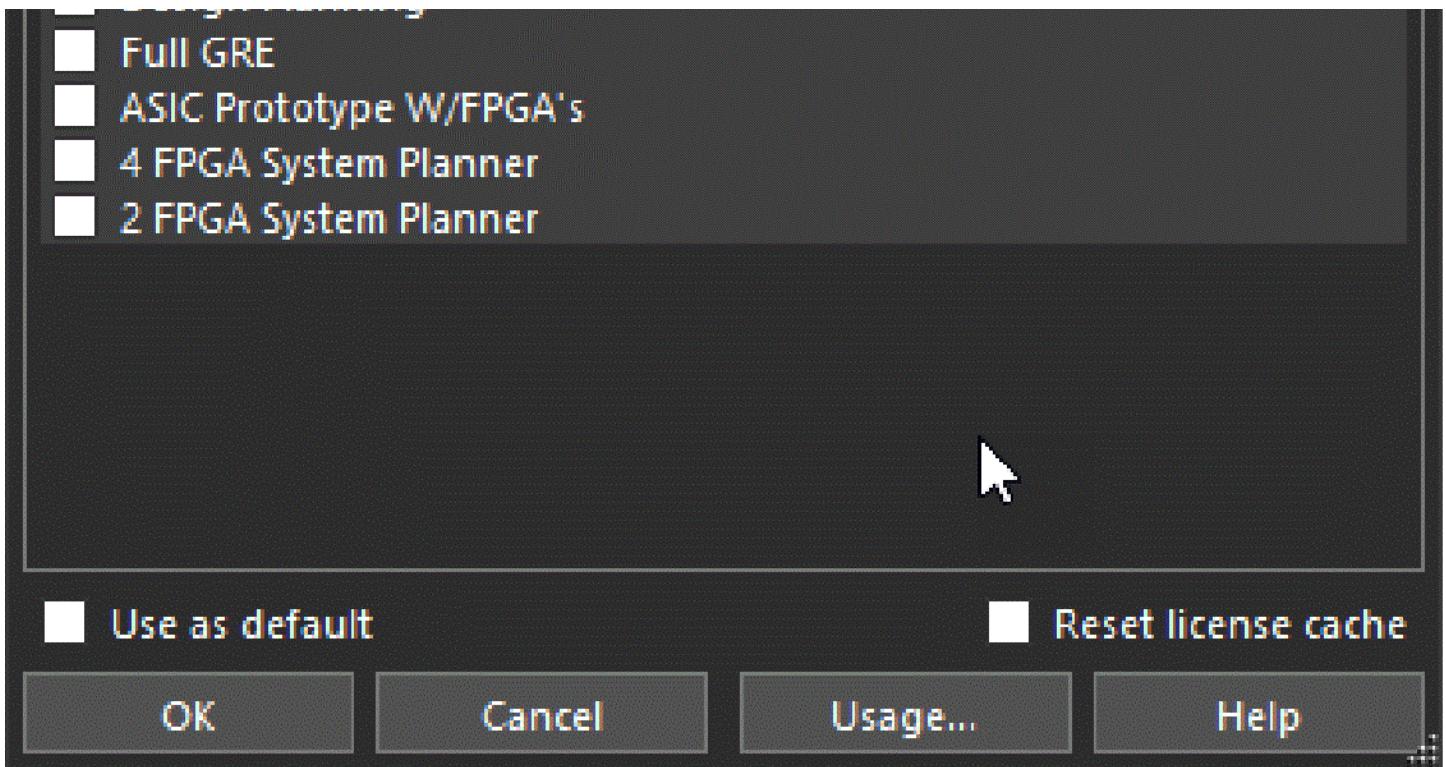
On Windows

- From the Windows Start menu, select *Cadence PCB 2022 – Allegro PCB Editor 2022*.
- Or
- At the Windows command prompt, navigate to the install directory, `<install dir>/tools/bin`, type `allegro` and press `Enter`.

On LINUX

1. At the shell prompt, type `allegro &` and press `Enter`.
The *Cadence 22.1 Allegro Product Choices* dialog box opens listing all the available product options.
2. Select the required product and product option, and click *OK*.





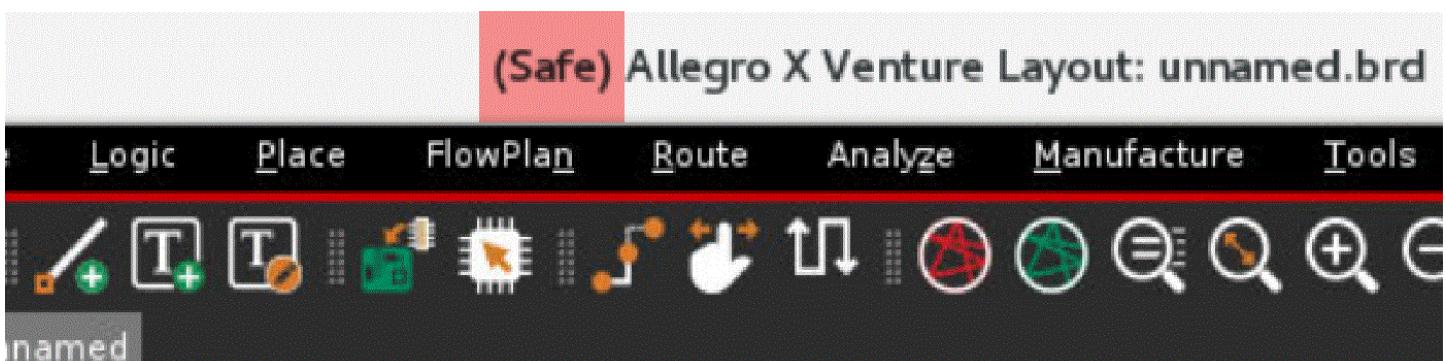
To stop this dialog box from appearing each time you launch the tool, select the *Use As Default* option, and click *OK*.

Starting Allegro PCB Editor in Safe Mode

If you experience any issue with user customizations or extensions, you can run the tool in the safe mode to debug the issue.

- To start Allegro PCB Editor in the safe mode, run the following command:

```
allegro -safe
```



i The safe mode should only be used to help diagnose issues. It should not be used for production work. Some applications may access local customizations as a prerequisite.

When launching the layout editor in the safe mode, the following are not loaded on startup:

- Local env file (<HOME>/pcbenv/env)
- cds_site configuration data
- Any user SKILL code

- Pre-registered scripts
- The `ini` file which stores window size and position information
- Most Recently Used files (MRU)
- Remembered Windows positions (`.geo` files)

Setting up a pcbenv Directory for Windows or UNIX

The layout editor creates a `pcbenv` directory with the `env`, `allegro.ini`, and `allegro.geo` startup files at a location determined by the value of the environment variable `HOME`. The `pcbenv` directory stores your window and toolbar preferences. Do not edit these files. Instead, use the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* (`enved` command) to make changes. For additional information, see [The User Preferences Editor](#).

If your initial default directory is inaccessible, you cannot save any of your preferences.

If you have not explicitly set a `HOME` variable, the tool uses a combination of the `HOMEDRIVE` and `HOMEPATH` variables to generate the home directory (`HOMEDRIVE:\HOMEPATH`) on Windows. If the `HOMEDRIVE` and `HOMEPATH` variables do not exist, the tool uses `c:\`.

The layout editor also lets you set the `ALLEGRO_PCBENV` environment variable to override the default location of the `pcbenv` directory. You must set the `ALLEGRO_PCBENV` variable before starting the tool, so that the Allegro tool looks for the startup files in the new location.

The `ALLEGRO_PCBENV` must be set at the operating-system level. On UNIX, add it to your `.profile` (`sh/ksh`) or to your `.cshrc` (`csh/tcsh`). On Windows, add it to your user environment variables using the same technique as adding a `HOME` variable, described below. Adding it to your environment file will not work.

Creating or Changing the HOME Variable

The `HOME` variable is used to locate the `pcbenv` environment file as well as other required user-specific files. By default, it is not used to store design data. Starting in Release 15.5, you can also set `ALLEGRO_PCBENV` (see above) to modify the location of the `pcbenv` directory, and if the `HOME` variable is not set, the default is the standard Microsoft *My Documents* location. On most Windows systems, this defaults to:

```
c:\Documents and Settings\<user login>
```

 Earlier versions of Allegro tools require a `HOME` variable to be set to a directory without any spaces.

To create or change the `HOME` variable for Windows:

1. Right click on *My Computer* and choose *Properties*, or choose *Start – Settings – Control Panel – System*.
2. Choose the *Advanced* tab.
3. Choose *Environment Variables*.
4. In the *User Variables* section, either click *New* or *Edit*.
5. To specify a `HOME` directory located at `d:\work`, for example, do one of the following:
 - a. If you clicked *New* in the previous step, add the following in the *New User Variable* dialog box:
 - b. Variable Name = `HOME`
 - c. Variable Value = `d:\work`
 - d. If you clicked *Edit* in the previous step, modify the following in the *Edit User Variable* dialog box:
 - e. Variable Name = `HOME`
 - f. Variable Value = `d:\work`
6. Choose *OK* to save the setting and dismiss the dialog box.
7. Choose *OK* to save and dismiss the *Environment Variables* dialog box.
8. Choose *OK* to save and dismiss the *System Properties* dialog box.

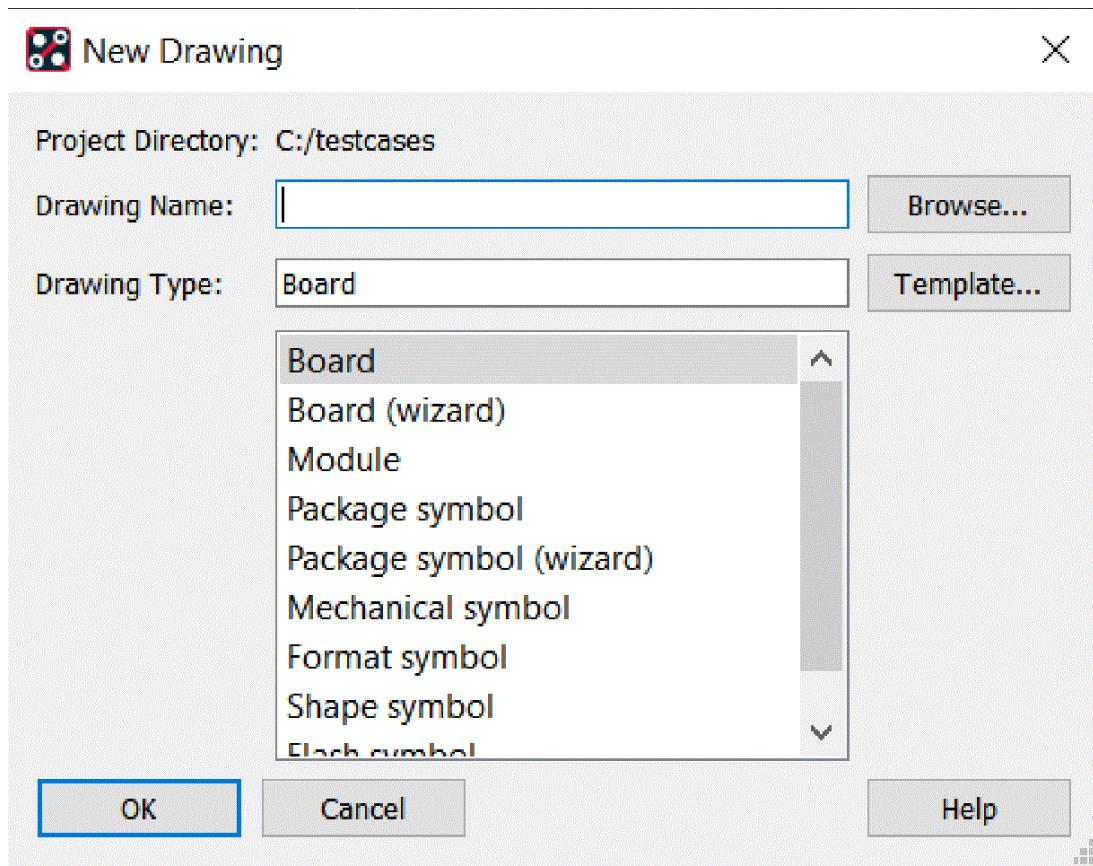
The next time you start your layout editor, the `d:\work\pcbenv` directory is created. The tool looks in this location for startup files (`env`, `allegro.ini`, `allegro.geo`, and so on.)

PCB Editor: Creating New Designs

Once your layout editor is running, you can open new and existing drawings using the appropriate items in the *File* menu. (If you have created designs in previous sessions, the editor opens the last saved design, based on information written to the `master.tag` file, described above.)

When you create a new design file, you must specify the type of design you want to create, using the New Drawing dialog box to select whether you want to create a board file or a symbol file.

Figure 2.28: The New Drawing Dialog Box (PCB Editor)



The choices are:

Layout

Creates a board file (.brd) or design file. You create a design database in this editing mode. Use this file to perform such tasks as component placement, board or design routing, and other functions.

Board (wizard)

The board wizard is designed either to help beginning users create a design in Allegro PCB Editor (board wizard is not available on Allegro Package Designer) or for experienced users who want a quick way to create a basic framework for a design as a foundation for a more complex design database. You can also use the board wizard to import custom design data by way of user-defined templates and technology files.

A template file is an existing user-created .brd file containing customized data. Information that you should include in a .brd template file includes default parameter settings, company-default subclasses, and color-to-layer assignments.

⚠ The template file should *not* contain any data on ETCH/CONDUCTOR, PIN, or VIA classes.

The board wizard accepts the following data from a template file. Board units and board origin are data contained in the template file that can be replaced. The wizard cannot replace the following parameters, but they can be modified after you create a new layout:

- Drawing size
- Board outline
- Spacing constraints
 - Minimum line width
 - Minimum line to line spacing

- Minimum line to pad spacing
- Minimum pad to pad spacing
- Package and route keepins
- Grid definitions
- Cross-section definitions

If the template file contains only two ETCH/CONDUCTOR layers, the wizard lets you add more layers and defines them as routing layers or power planes. If additional layers are defined in the template, this functionality is disabled in the wizard.

If you import data using a template file and a tech file, note that the data in the tech file takes precedence over data brought in from the template. A tech file template should include constraint (DRC) rules and layer stack-up information. See the *Defining and Developing Libraries* user guide in your documentation set for details on technology files.

Templates and technology files that you can import into the design database should contain the following default parameter settings:

- Company-default subclasses
- Color-to-layer assignments
- Constraint (DRC) rules
- Layer stack-up information
- Mechanical (.bsm) symbols

If you choose not to load data from template or technology files, Board Wizard lets you input the data manually, from the wizard's user interface screens.

For procedural details, see the *Allegro PCB and Package Physical Layout Command Reference*.

Symbol

You create symbols for a design in the symbol editing mode. The tool appends the appropriate filename extension when you save a symbol.

There are two files associated with a symbol. The raw, unprocessed, drawing file has a .dra filename extension. When you choose *File – Create Symbol (create symbol)* command from the symbol editing mode, the .dra file is compiled into the appropriate binary file — Package (.psm), Format (.osm), Mechanical (.bsm), Shape (.ssm), or Flash (.fsm).

The layout editor automatically creates a symbol every time you save a drawing (.dra) when you are in the Symbol Editor. You no longer need to compile the symbol and save the drawing in two separate steps.

Set the environment variable, *no_symbol_onsave* to restore the legacy behavior and allow the layout editor to compile the symbol and save the drawing in two steps.

1. Choose *Setup – User Preferences* to display the User Preferences Editor.
2. Choose *Drawing* and then click the *no_symbol_onsave* environment variable.

See the *Defining and Developing Libraries* user guide in your documentation set for information about symbol files.

The symbol editor lets you create the following types of symbols:

Package Symbol

Creates a new component symbol such as an IC. The tool saves package symbols to the symbol library, by means of *File – Create – Symbol*, and appends the file name that you specify with a .psm extension.

Mechanical Symbol

Creates a drawing symbol such as a card edge connector or a board/design outline. The tool saves mechanical symbols to the symbol library and appends the file name that you specify with a .bsm extension.

Format Symbol

Creates a drawing symbol such as a legend or a company logo. The tool saves format symbols to the symbol library and appends the file name that you specify with an .osm extension.

Shape Symbol

Creates a drawing symbol such as a special shape for a padstack. The tool saves mechanical symbols to the symbol library and appends the file name that you specify with an .ssm extension.

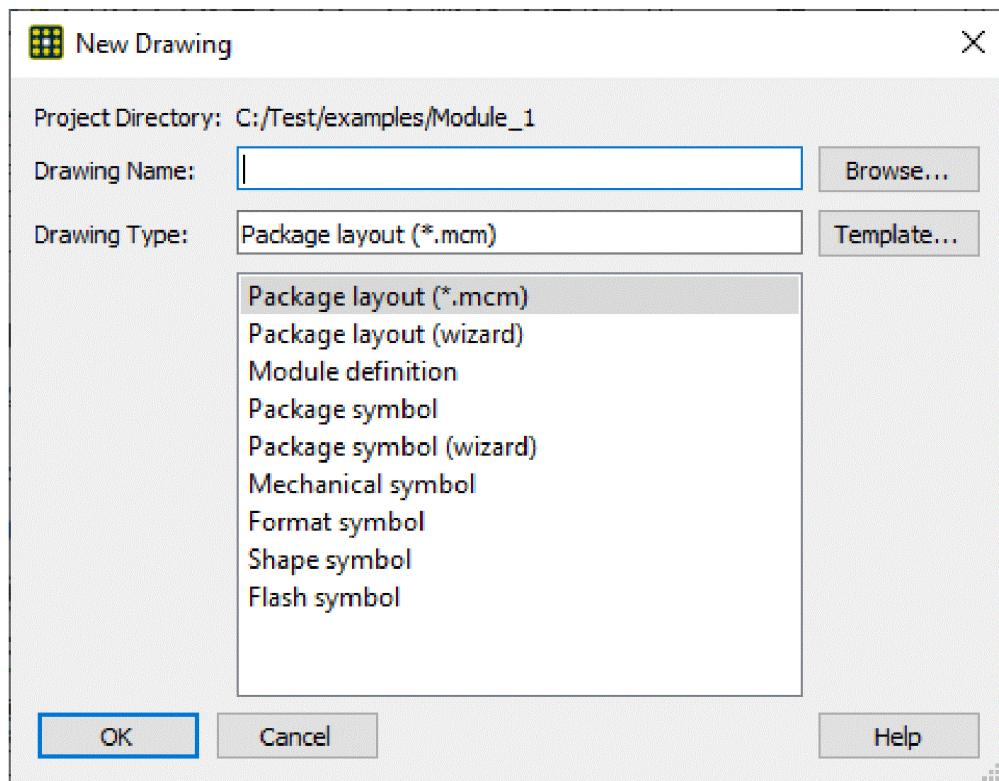
Flash Symbol

Creates a flash symbol such as a thermal pad for Raster formats. The tool saves flash symbols to the symbol library and appends the file name that you specify with an .fsm extension.

APD+: Creating New Design

Design work, which entails symbol and layout creation, occurs within the context of a design that you create. Use the [new](#) command to specify type of drawing that you want to create: component or symbol (Figure 2-31).

Figure 2.29: New Drawing Dialog Box



Drawing types include:

- Component/multi-chip – Creates a design file (`.mcm`). A design file represents the drawing database. Use this file to perform such tasks as component placement, design routing, and other functions.
- Module definition – Creates a module file (`.mdd`). Module files are collections of physical entities (which can include other modules). Modules may or may not have logic (represented by nets, components, and so on) or a block (a collection of schematic information used by a schematic tool) associated with them. They can be permanently stored as module definition databases in library files.
- Symbol – Creates a symbol file. APD+ saves these databases as files with the `.dra` extension. This invokes the Symbol Editor, from which you can create the following types of symbols:
 - Component symbol – Manually creates a new component symbol such as a die or a discrete. APD+ saves component symbols to the symbol library and appends the file name that you specify with a `.psm` extension.
 - Component symbol (wizard) – Creates a new component symbol such as a die or a discrete using the Package Symbol Wizard.
 - Mechanical symbol – Creates a drawing symbol such as a card edge connector or a design outline. APD+ saves mechanical symbols to the symbol library and appends the file name that you specify with a `.bsm` extension.
 - Format symbol – Creates a drawing symbol such as a legend or a company logo. APD+ saves format symbols to the symbol library and appends the file name that you specify with an `.osm` extension.
 - Shape symbol – Creates a drawing symbol such as a special shape for a padstack. APD+ saves mechanical symbols to the symbol library and appends the file name that you specify with an `.ssm` extension.
 - Flash symbol – Creates a flash symbol (`.fsm`) used in various artwork processes.

Figure 2-32 shows the New Drawing Configuration dialog box that appears after you name your drawing and choose a drawing type. You choose the component configuration and accept the design parameter defaults in this dialog box. You can override these defaults. See [Setting Drawing Parameters](#).

Figure 2.30: New Drawing Configuration Dialog Box

Opening Existing Designs

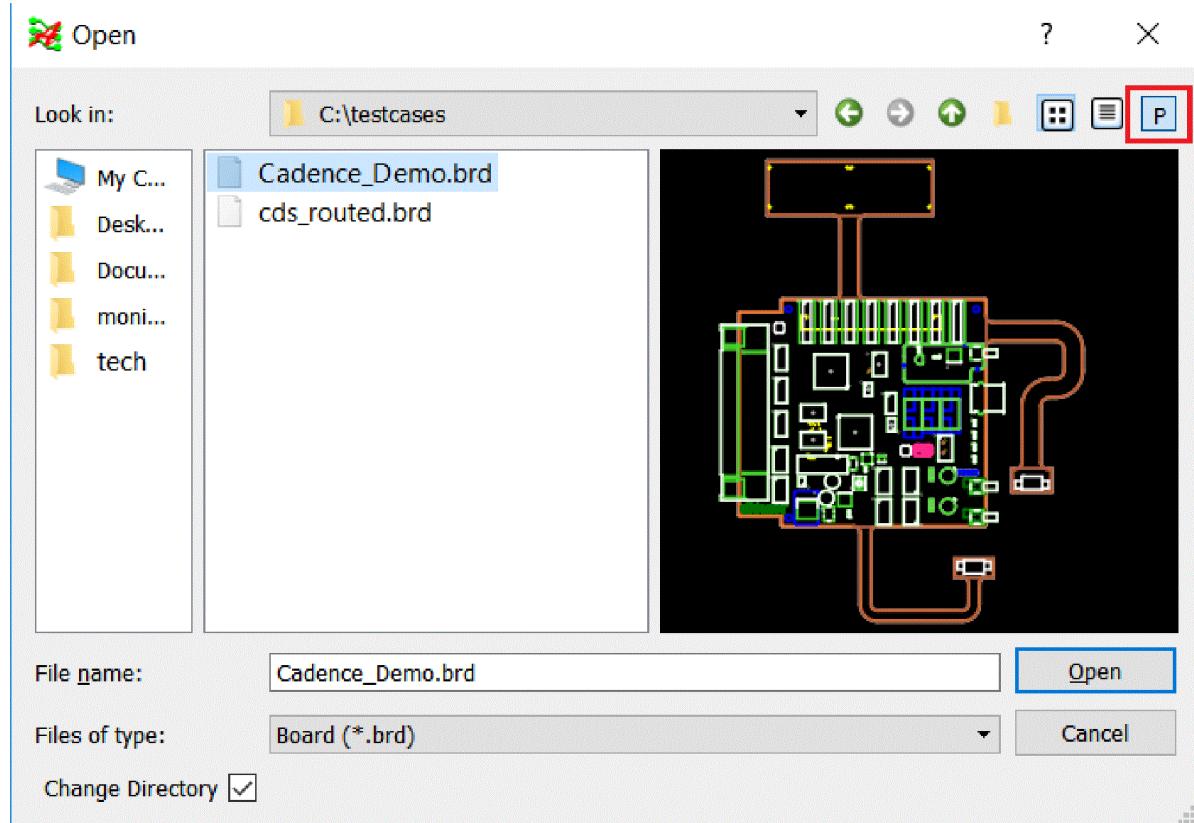
You can open existing drawings in the following four ways:

- From an operating system prompt, as described in [Starting the Layout Editor from an Operating-System Prompt](#)
- From Start page, as listed in the *Recent Designs* tab
- From within the layout editor using *File – Open (open)* command
- Drag and drop design file on the layout editor canvas

 You are prompted to save any changes made to an open design before opening a new file, but may be prohibited from doing so if the database has been locked. For details on database locking, see [Protecting Files with Edit Locks](#).

You can display information for an existing drawing before opening it by using the Quickview window in the Open dialog box. Quickview provides a high-level graphic overview or a summary of properties of the database you select from the list. The information that appears is based on the icon you press in the dialog box. [Figure 2-33](#) is an example. Use preview button, located at the top right corner of the file browser dialog, to toggle the display of design quick view.

Figure 2.31: Quickview in the Open Dialog Box



For additional information on Quickview, see "[Using Data Browsers](#)" in Using the Layout Editor.

Saving Automatically

The layout editor lets you automatically save an active design or symbol at regular intervals when you set the *autosave* environment variable. When the tool saves a design, it automatically generates a file named `AUTOSAVE.brd` (a symbol is saved to a file named `AUTOSAVE.dra`) and places it in the directory that was active when you opened the tool. If you change directories, the tool saves the file to the original working directory. The saved file is kept after you have closed and saved the design or symbol and exited the software.

⚠ The autosave option is automatically disabled if you invoke the database locking command, `file_property`. For details on this feature, see [Protecting Files with Edit Locks](#).

If the autosave time is reached when a command or non-filled shape is active, the tool displays a message that reads "Save Pending." The save executes when the command is completed or when the shape is filled. If you have not executed a command since the last autosave, the tool does not resave the design.

Activating the Autosave Utility

- Set the `autosave` variable in the Environment Editor by choosing *Setup – User Preferences* (`enved` command). See [Managing Environment Variables](#) for details on environment variables.
or
- Before opening a design, execute the following command from the console window prompt:

```
set autosave
```

You can specify the interval at which checkpoint saves are made by using the `set` command and the `autosave_time` variable as follows:

```
set autosave_time = <time>
```

The `<time>` can be set from 10 to 300 minutes. The default is 30 minutes.

Changing the Default Name (AUTOSAVE) of the Generated File

- Running the `enved` command to display the User Preferences Editor dialog box and entering a new value for `autosave_name` in the *Autosave* Category
or
- Using the following command and the `autosave_name` variable

```
set autosave_name = <filename>
```

The tool lets you specify whether a database check is performed when a design or symbol is saved with the autosave facility.

Enabling a Database Check

- Set `autosave_dbcheck` in the User Preferences Editor dialog box
or
- Execute the following command from the console window prompt:

```
set autosave_dbcheck
```

Note that enabling the database check during autosave requires additional processing time. The default is `disabled`.

Disabling the Autosave Facility

- Uncheck `autosave` from the User Preferences Editor dialog box
or
- After opening a design, execute the following command from the command line:

```
unset autosave
```

Suppressing the Overwrite File Confirmers

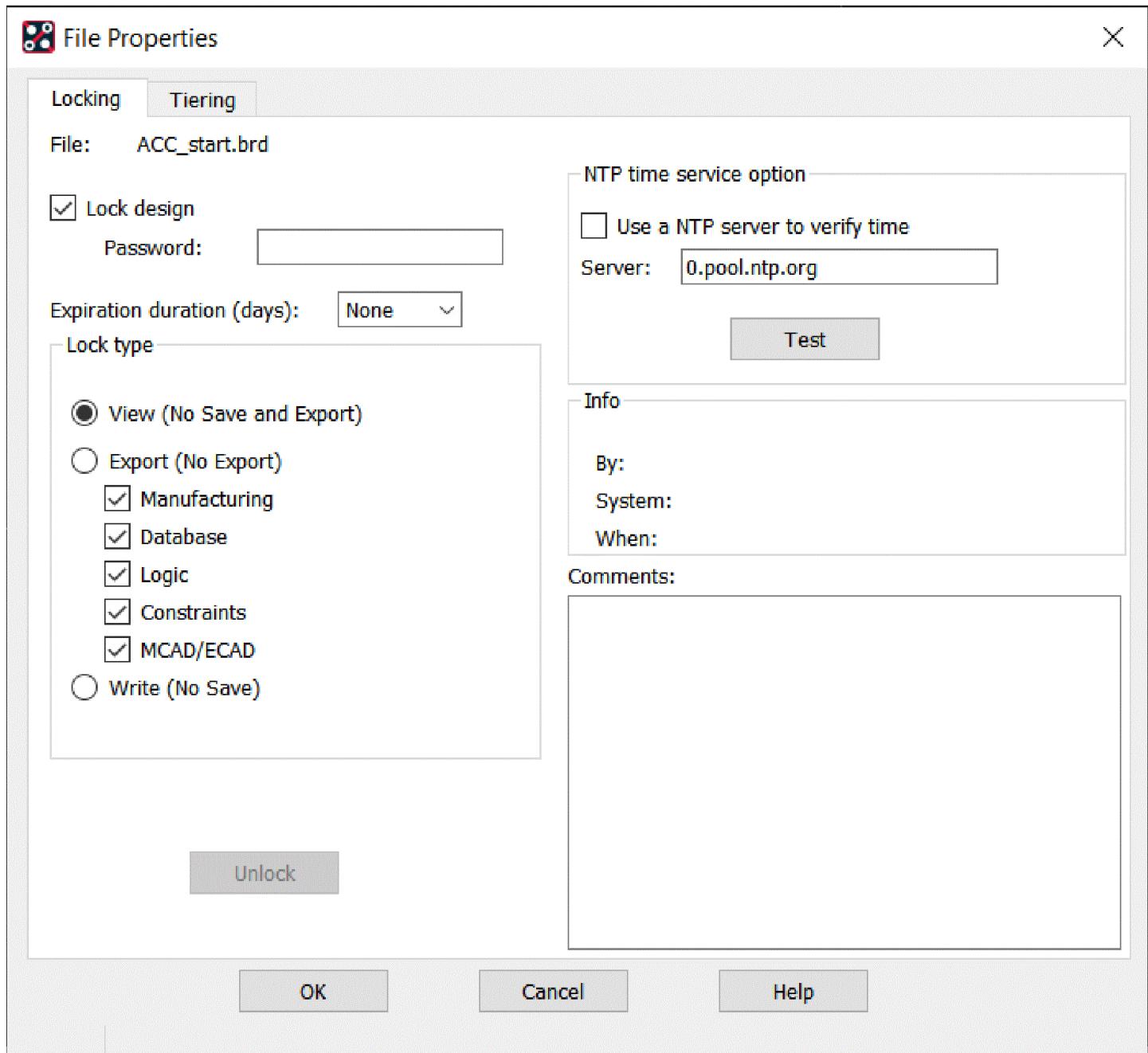
To disable the overwrite confirmers that automatically appear when you save an existing file, disable the `noconfirm_savedb` environment variable in the *Drawing* category of the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* (`enved` command). No warning message displays even if saving will overwrite an existing database.

Saving to an Earlier Version

The databases are backward-compatible with their major version number (the number to the left of the dot). This means that databases created in or upreved to any revision within a major version (for example, to 14.1) can migrate between revisions of that version. You *cannot* save any major version to an earlier one, such as 15.x to 14.x, 14.x to 13.x, and so on.

Protecting Files with Edit Locks

You can secure any design database file by choosing *File – Properties* ([file_property](#) command) to set an optional password-protected database lock. Doing so marks the file as read-only in the database (as opposed to on the platform's operating system). This ensures that the design is not accidentally replaced by you or an unauthorized user when attempting to save over the file.



In addition, you can set database locking to disable the export of design data such as writing techfiles, exporting libraries, and creating modules. A set of export options are available as check boxes with *View* and *Export* locks. These options create different groups of export commands. You can select none, some or all the options. If an option is selected, the export commands belonging to that group are disabled.

Database locking also turns off the autosave environment variable. The locking mechanism does *not* prohibit you from performing an uprev of the database in batch mode; however, batch programs that open databases for writing, such as `netrev` and `netin`, are unable to perform their operations when the database is

locked.

When a database lock has been set, editing the file results in an error message, warning the user that the database has been locked for saving. (Edit locking will *not* inform you if another user has the file open.) The lock can be disabled only by entering the password established when the file was locked or, if a password was not set, by unlocking it in the File Properties dialog box or through the `dbdoctor` command. For procedures on locking files through the user interface or at the system prompt, see *File – Properties* (`file_property` command) or *Tools – Database Update* (`dbdoctor` command), respectively, in the *Allegro PCB and Package Physical Layout Command Reference*.

Tip: *It is extremely important that you record any passwords used to lock databases. Cadence does not support the recovery of databases in a locked state due to forgotten passwords.*

Because a design might be legitimately opened for updating by any number of users in a large, networked system environment, the File Property dialog box displays the name of the user who locked the file, when it was locked, and on which system it was locked. A comment field allows you to provide additional information. These comments, as well as the option for prohibiting design data export, cannot be altered when the file is locked.

File Types

The layout editor automatically attaches the appropriate extension to the base filename that you specify. These extensions indicate the following file types:

Extension	File Type
.art (default)	Artwork files. ⚠ You can change the default file extension of <code>.art</code> for artwork film filenames by setting the <code>ext_artwork</code> environment variable in the User Preferences Editor, available by choosing <i>Setup – User Preferences</i> (<code>enved</code> command).
.brd	Board file that represents the drawing database
.bsm	Library file that stores drawing or board symbols
.cio	Specifies a file containing a co-design die.
.cml	Used in component design, the Condensed Macro Library (.cml) file type stores the LEF data for those pins of a macro that impact your component design.
.dat	Data files.
.def	Used in component design, the Design Exchange Format (.def) file type is an industry-standard format developed by Cadence Design System for representing digital IC implementation data.
.dfa	Design for Assembly file.
.dpf	Design Partition file.
.dpm	Design Partition file.
.dps	Design Partition file.
.dra	Drawing file. You must create one of these before you create a symbol file. Later, this file is compiled into a binary symbol file.
.drl	NC drill output files. ⚠ You can change the default file extension of <code>.drl</code> for NC drill output filenames by setting the <code>ext_drill</code> environment variable in the User Preferences Editor, available by choosing <i>Setup – User Preferences</i> (<code>enved</code> command).
.fsm	Library file that stores flash symbols.
.jrl	A journal file which contains a record of events — menu picks, keyboard activity, and so on — which are recorded for each session in your layout editor. You can share this data with Cadence Usability staff to help us learn how you use the product, which will assist us in our efforts to improve the user interface.
.ldf	Used in component design, the LEF Definition file type defines libraries and the paths to the LEF files defined in them.
.lef	Used in component design, the Library Exchange Format (.lef) file type is an industry-standard format developed by Cadence Design System for representing digital IC implementation data.
.log	Log file that contains data on processes.

.mcm	Multi-chip module file (APD)
.mdd	Library file that stores module definitions.
.ncr	Output file in Excellon Format 2 for numerically controlled routers.
.osm	Library file that stores format symbols.
.pad	Padstack file.
.psm	Library file that stores package symbols.
.prm	Database parameter file that contains customized parameters exported from one design and imported into another for reuse.
.rou	Output ASCII text file in Excellon format for an NC router based on parameters set in the NC Parameters dialog box, available by choosing <i>Manufacture – NC – NC Route</i> (<code>ncroute</code> command).
.scf	A System Configuration File that specifies the relationship between the <code>.mcm</code> file and the <code>.cio</code> files. The file name of the <code>.scf</code> uses the base <code>.mcm</code> name followed by <code>_codesign.scf</code> . Together, these files are called a design link.
.scr	Script and macro files.
.ssm	Library file that stores shape symbols.
.tap	Output text files that contain NC drill data.
.txt	Text file, such as that used for parameters.

Opening a `.pad` file invokes the Padstack Tool. Opening a `.brd` file starts the Workspace Editor with the layout menu set. Opening a `.bsm`, `.osm`, `.psm`, `.fsm` for `.ssm` file starts the Workspace Editor with the symbol menu set.

When you finish with a `.dra` file in the symbol editor, choose *File – Create Symbol* ([create symbol](#) command). The tool converts the file to a binary, symbol type file.

The layout editor supports the storage of log files, journals reports, and artwork films in a subdirectory under the board file location. Three environment variables control the output locations:

- `ads_sdreport` – report location
- `ads_sdlog` – log file/journal location
- `ads_sdart` – artwork and NC output

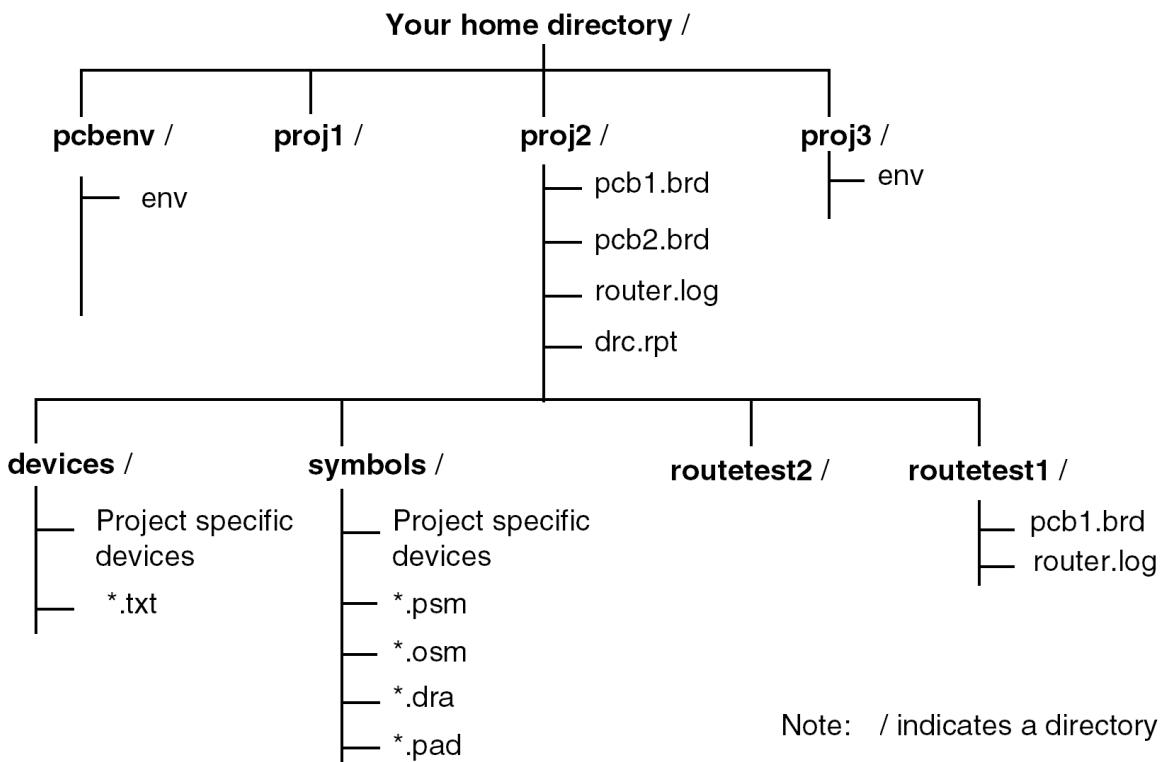
If a directory does not exist, the editor creates one.

You can access these environment variables when you choose *Setup – User Preferences*.

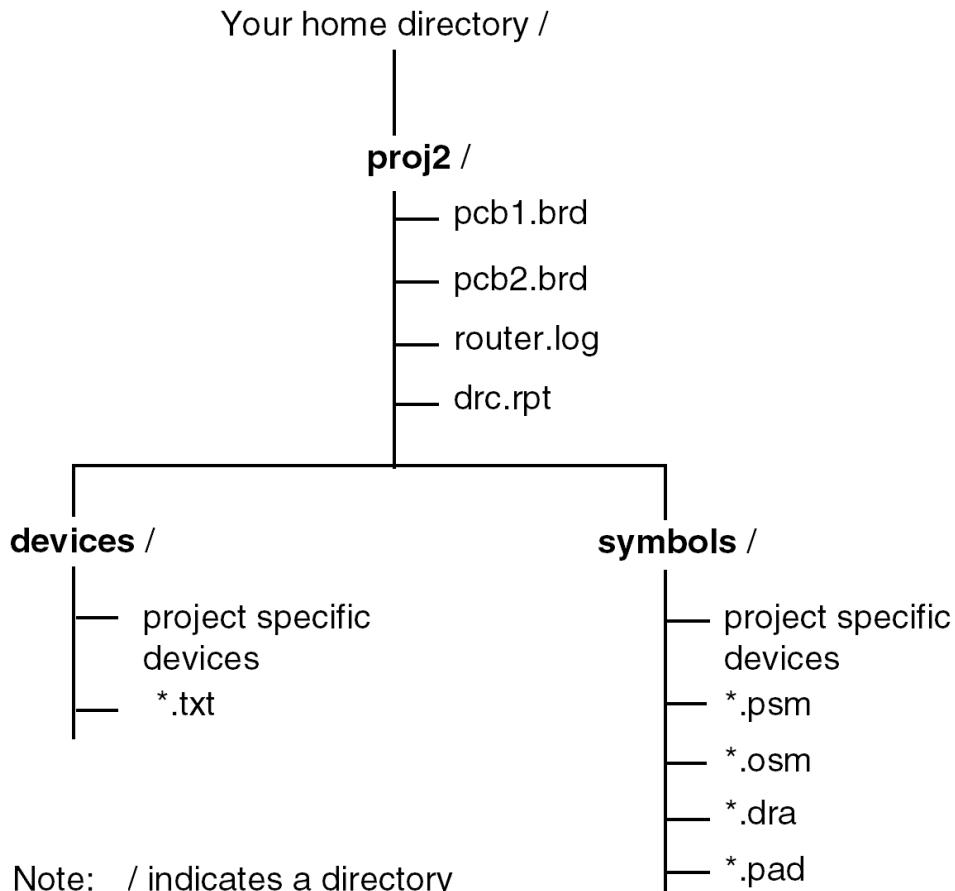
Setting Up a Working Directory Structure

Figure 2-34; shows a suggested directory structure for your projects. This structure lets you have several project directories (for example, `proj1` and `proj2`) and have subdirectories under each project.

Figure 2.32: Suggested Directory Structure for PCB Editor Projects



The symbols and devices directories beneath a project directory contain symbols and devices that are unique to that project. These subdirectories parallel the structure of the library directories supplied by the layout editor in `<install_dir>/share/lib/pcb_lib`.



A project can also contain other subdirectories, such as temporary directories for routing tests that let you run batch routes without replacing log or design files.

Manipulating Design Elements

The Allegro graphical user interface (GUI) adheres to most Microsoft Windows™ standards for pull-down menus, accelerator keys, mouse use, icons, and so on. The layout editor lets you execute commands in one of two methods:

- *command–then–element, or menu-driven editing mode*, in which you first choose a command then elements to be acted upon.
- *preselect use model, or noun-verb*, in which you choose elements to be acted upon, then the command, when you work in an application mode

Using the Mouse

Cadence recommends a three-button mouse. Using a three-button mouse eliminates the need to hold down the Control key while using the right mouse button to pan, zoom in, and zoom out.

Left Mouse Button

Use the left mouse button in conjunction with an active command to select graphic design elements such as lines, pads, and text. The selected feature is highlighted.

⚠ You can move a shape or void with the left mouse button if you enable the *shape_drag_move* design-level environment variable in the User Preferences dialog box, available by running the `enved` command. The shape commands also change the cursor to indicate the legal operation to perform.

You can also use this button to choose commands from menus, tabs, or icons. In dialog boxes with entry fields that list built-in options, the left mouse button can be used in the data field to display and choose these options (for example, the *Options* window pane).

Keyboard Sequence	Functionality
Shift plus left mouse button	Adds elements to the selection set in an application mode
Control key plus left mouse button	Deselects items by pick.
Double-click left mouse button	Extends left mouse click for specific commands: <ul style="list-style-type: none"> • Using <i>Route – Connect</i> (add connect) to add traces, inserts a via. • Using <i>Edit – Vertex</i> deletes a vertex. Double-clicking the left mouse button on any edge of shape also selects it.

Middle Mouse Button

Press and hold the middle mouse button while moving the mouse in the direction you want to pan and use the view (zoom) features (see [Viewing a Design](#)). If you click the middle mouse button, the system either zooms in or out, based on the direction in which you move the cursor. If you move from top left to bottom right, the display zooms out. If you move from bottom right to top left, the display zooms in. In both cases, a rectangle that depicts the new zoom area appears. You can disable the zoom functionality by setting the environment variable `no_dynamic_zoom`. In Windows, for Wheel mouse devices (middle mouse button is a wheel), the middle mouse button must be defined so that the roam function works correctly. Access the Control Panel to open the Mouse Option Control and check the behavior.

Right Mouse Button

Application-mode and pre-select use model commands are accessible from a right mouse button pop-up menu based on the current selection set. The commands that populate an application mode pop-up menu depend on:

- Current application mode
- Design elements already in the selection set
- Design elements selectable at the current mouse position

Keyboard Sequence	Functionality
<code>Shift + right click</code>	Extends the functionality of the active command for two-button mouse devices, and allows roaming.
<code>Ctrl + right click</code>	Executes strokes. Using <i>Edit – Groups</i> (<code>groupedit</code> command), lets you delete an item from the group.
Double-click right mouse button	Choose and select right mouse pop-up if one is available in command. Drag and drop, click and select pop-up and move to item in pop-up. If you set the <code>no_dragpop-up</code> environment variable, then right-drag-click performs a stroke.

Keyboard Shortcuts

Keyboard shortcuts and accelerators let you perform a number of actions without using the mouse, including changing the view of the design and displaying dialog boxes from the user interface.

Select by Window

Create a selection rectangle by clicking the left mouse button to pick a corner for the rectangle, then holding the left mouse button and dragging the mouse. All applicable items with the rectangle are selected.

Select by Group

While using a command in the menu-driven editing mode, rather than the noun-verb (pre-select) use model, click the right mouse button to display the pop-up menu. Choose *Temp Group*. Choose the elements you want to group together. Each element you choose is highlighted. When you choose all the elements, right-click again to display the pop-up menu and choose *Complete*. All objects selected are added to a preselect buffer.

Deselect Support

In the menu-driven editing mode, rather than the noun-verb (pre-select) use model, use the `Control` key and left mouse button to deselect a selected object in temp group mode (in commands that support this option using the right mouse pop-up menu). To complete the selection, choose *Complete* from the right mouse pop-up menu. If you use the `Control` key while holding down the left mouse button, you can deselect multiple objects using a bounding box.

Viewing a Design

The easiest way to zoom in, zoom out, and move across the design workspace is using the middle mouse button. The button gives you access to all the zoom features available from the menu bar or keyboard commands (except `zoom in`, which is integrated into `zoom points`) without the need to make a menu selection or enter a command at the console window prompt. Use of the middle mouse button also enables you to roam or pan across a design.

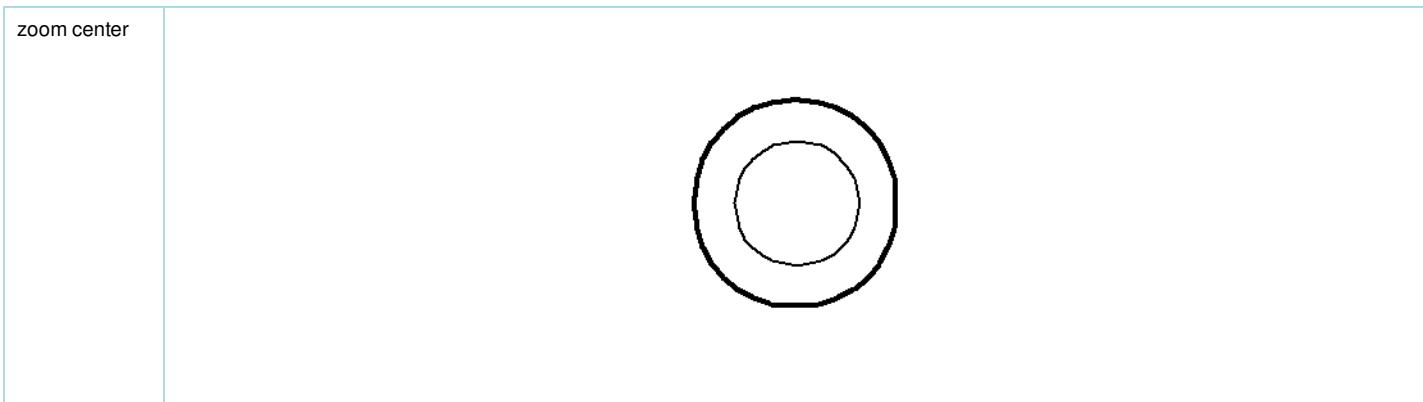
Roaming

Roaming or panning are the terms used to describe the action of moving across a design in the workspace. To pan a design:

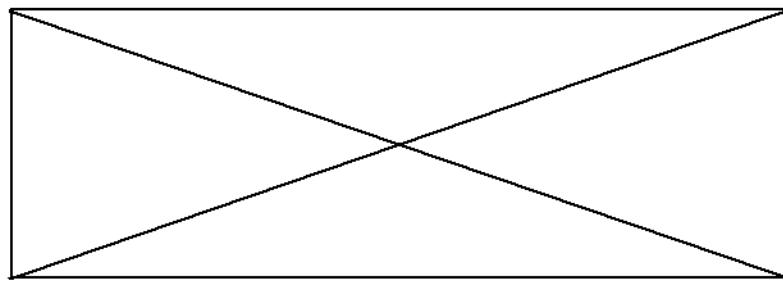
- With the cursor inside the design workspace, click and hold the middle mouse button as you drag the cursor across the design. As long as the mouse button remains pressed, you can move all areas of the design into full view. You cannot drag the cursor outside the boundaries of the design.

Zooming

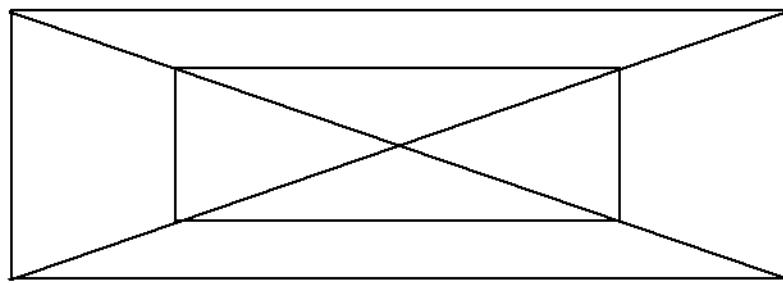
Zoom functionality is dependent on the position of the cursor relative to its location when you first click the middle button (the "starting pick"). Movement of the cursor up or down, left or right of this coordinate determines what zoom function is active (as shown in Figure 2-35). Zoom center is the active zoom mode when the cursor is at its starting pick (dynamically displayed in the design as concentric circles). The mode you are in is displayed in the status bar and by way of dynamic shapes that bound the affected areas. The shape geometry associated with each command is:

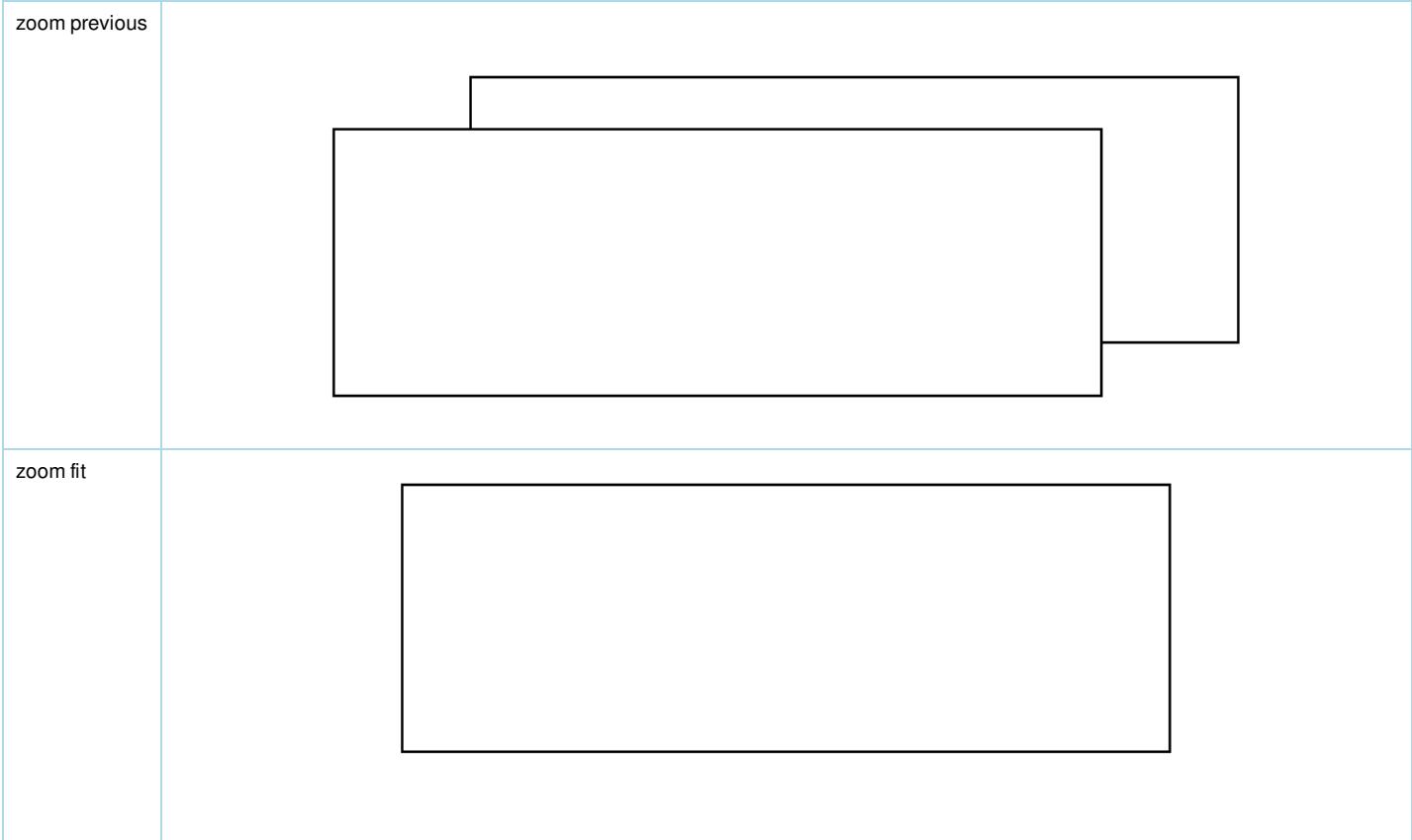


zoom points



zoom out

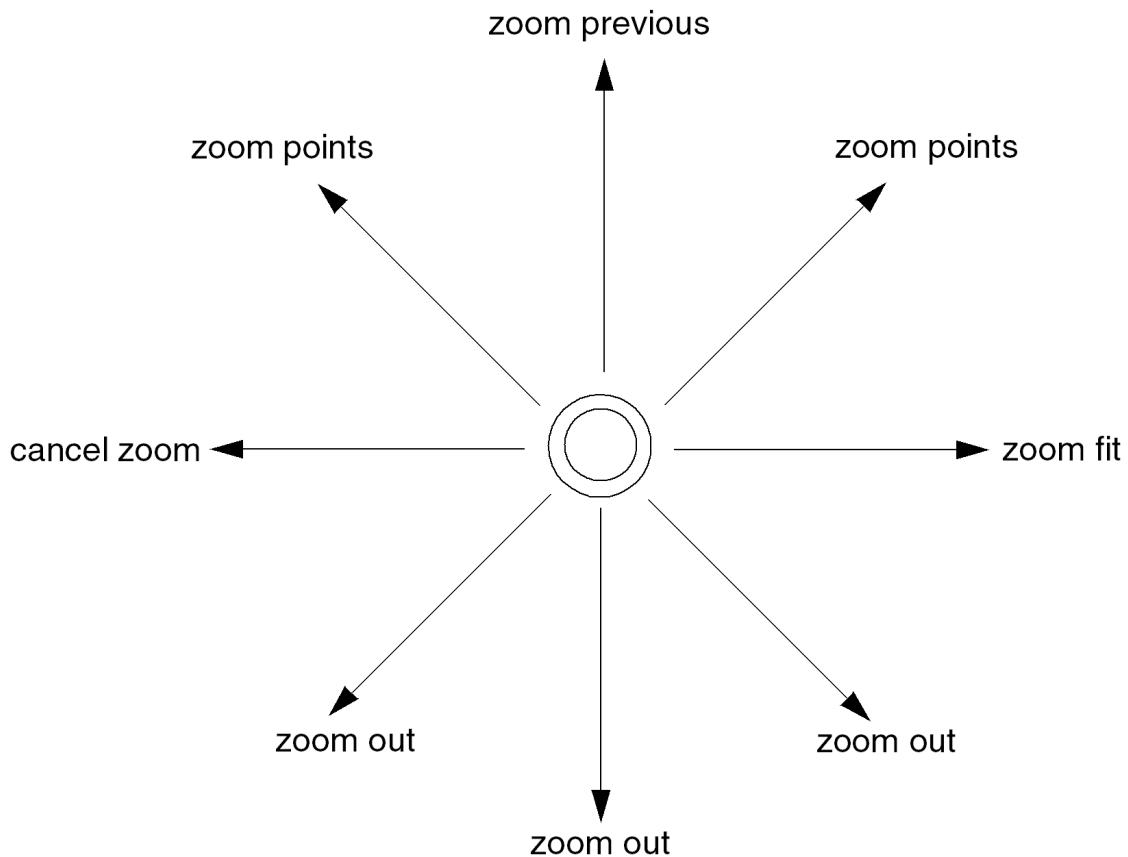




To enhance performance, `zoom out` repaints the design in a minimalized draw mode. This "skeletal" view is maintained until the second click completes the zoom operation. While you are in this mode, the following conditions apply:

- Pins, vias, and ratsnests are not drawn
- Line segments are drawn without endcaps
- Lines are drawn in single pixel width
- Shapes are unfilled
- Only reference designator text is drawn
- Layer visibility settings are ignored (all layers are visible)

Figure 2.25: Zoom Modes Relative to Starting Pick



The zoom function remains active until you click the middle mouse button a second time. (Clicking the left mouse button also takes you out of zoom mode.)

If you prefer not to use the dynamic zoom features, you can disable the functionality by setting the environment variable `no_dynamic_zoom` in the User Preferences Editor. By setting this variable, middle mouse button functionality is limited to zooming in or zooming out.

For example, if you move the mouse pointer left, the design will appear to move to the right in the design window.



View Functions

You can control the view of a design by choosing commands from the *View* menu, or by using designated icons, function keys, keyboard accelerators, or mouse strokes. (See "Using Strokes and Associated Commands" in Using the Layout Editor for details on command strokes.)

The following list includes the ways you can zoom in or out on a design, or move the design in the design window.

- Zooming in on a design
- Zooming in on a specific section of a design
- Zooming out on a design
- Zooming out to a full view of a design
- Zooming out to a full view of a design, excluding legends and borders
- Centering an element in the design area
- Zooming back to the last previous window extents

Customizing the User Interface

You can customize your menus by adding, changing, or removing items. The default menus shipped with Allegro products are available at the following location:

```
<cdsroot>/share/pcb/text/cuimenus
```

Allegro finds the menus with its MENUPATH environment variable in the User Preferences Editor, available by choosing *Setup – User Preferences (enved)* command). You should not modify any other file type in this directory as only the menu files are supported for user modification.

- ① As new products are added in a release; new menu files may be added, and Cadence may change the name of any menu file in a release.

For each graphics editor, separate menus exist for the drawing and symbol editors. Use these menus as starting points for customization. Set MENUPATH to <a local directory> plus \$MENUPATH in the local env file to ensure Allegro retrieves customized local menus first. After customization, deposit your version in the following location:

```
<cdsroot>/share/local/pcb/menus
```

To customize the Cadence-supplied environment, use the operating-system variable CDS_SITE, which overrides the default site location:

```
<cdsroot>/share/local
```

See "Site Customization" in Managing Environment Variables, which describes the options available with CDS_SITE.

Putting the menu in this location lets all your users see this version at startup. To understand this file's format, see:

```
<cdsroot>/share/pcb/examples/skill/DOC/FUNCS/axlMenuDoc.txt
```

- ✓ Switching between the symbol and drawing editors reloads the menu, allowing you to perform test edits without restarting the graphics editor. If you have a tool with Skill access, you can also type the following at the command line:

```
skill axlUIMenuLoad("<menu file>")
```

Currently, no mechanism exists to customize the right mouse button pop-up items in the drawing canvas.

Changing Fonts

The layout editor lets you customize the look of the graphical user interface by changing the size and type of the fonts in the console, status, and *Options* windows, and in the Find window pane. This can be convenient if you find it difficult to read information presented in the default size and type.

To change fonts in the user interface:

1. Exit the layout editor, if you have it running.
 2. Set the font variables in your environment file.
- These variables can also be set in the System dialog box in the Control Panel.

fontSize = -12	where -12 represents the default font size. A larger negative number (for example -20) makes the font larger. Do not use positive numbers in this value.
fontFace = helvetica	where helvetica represents the default font type. Fonts available to you depend on your platform and any user-installed fonts. The value is always a font name.
fontWeight = 500	where 500 represents bolded type. Change the value to 300 to produce unbolded type.

3. Restart the tool.
4. Resize the window, if necessary, to display all information in the larger font size.

You can also change font variables in the User Preferences Editor dialog box by running the `enved` command. Note that you must restart the tool to see the change.

Command Browser

You can access the complete selection of keyboard commands through the **Command Browser**. This dialog box lets you either run the command or view any online documentation associated with that feature. For procedures on using the Command Browser, see the *Allegro PCB and Package Physical Layout Command Reference*.

Optimizing the Display

The display features are optimized by way of the *display_raster_ops* environment variable. This variable is set to *on* by default. However, based on your hardware, you may notice slower performance while performing tasks that use extensive display capabilities — for example, via shoving while etch editing. If performance slows during these tasks, set the variable to *slow* using *Setup – User Preferences* (*enved* command). This setting disables display enhancement for tasks that bring complex displays into use (other tasks are unaffected).

To disable the *display_raster_ops* environment variable, set the variable to *off*.

Running Commands in the Background

This section is specific to the layout editors on UNIX workstations.

Normally, when you run a command from the command line, you cannot use the Design Window until the command is complete. When you type a command at an operating-system prompt in a UNIX shell window, you cannot use the shell window until the command completes. By running commands in the *background* you are able to continue using the design window or shell window.

While the background job is running, you can look at the contents of the output file with the UNIX commands *more* or *type*.

When a job completes, you are notified with a message in the console window.

PCB Editor: Design Flow

Cadence's Allegro PCB Editor integrated suites of software tools for systems design help you perform the major tasks of PCB design, including:

- Logic design entry
Create a printed circuit board design based on data from a Allegro Design Entry HDL, System Connectivity Manager, or Allegro PCB Design CIS schematic, or based on a netlist from another CAE system. Then backannotate from the design to the schematic. Update the Allegro PCB Editor designs by performing engineering change orders (ECOs).
- Physical layout
Place design elements and route them, either manually or automatically with Allegro PCB Router.
- Design analysis
Perform design analysis with SigNoise and EMControl.
- Manufacturing output
Generate silkscreens and penplots, and create artwork.

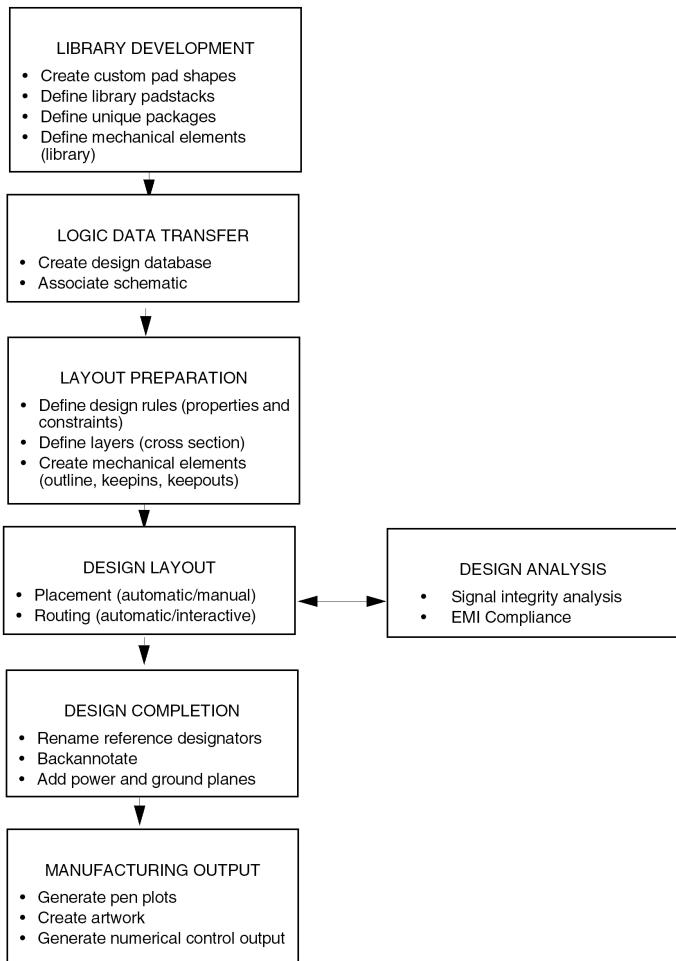
Figure 2-1 shows the functional relationship between Allegro PCB Editor and other Cadence/EDA tools for logic design, physical layout activities, and design analysis.

Figure 2.34: Functional Relationship Among System Design Tools



Figure 2-1 defines the typical PCB design flow process.

Figure 2.35: Design Flow Process



APD: Component-Design Flow

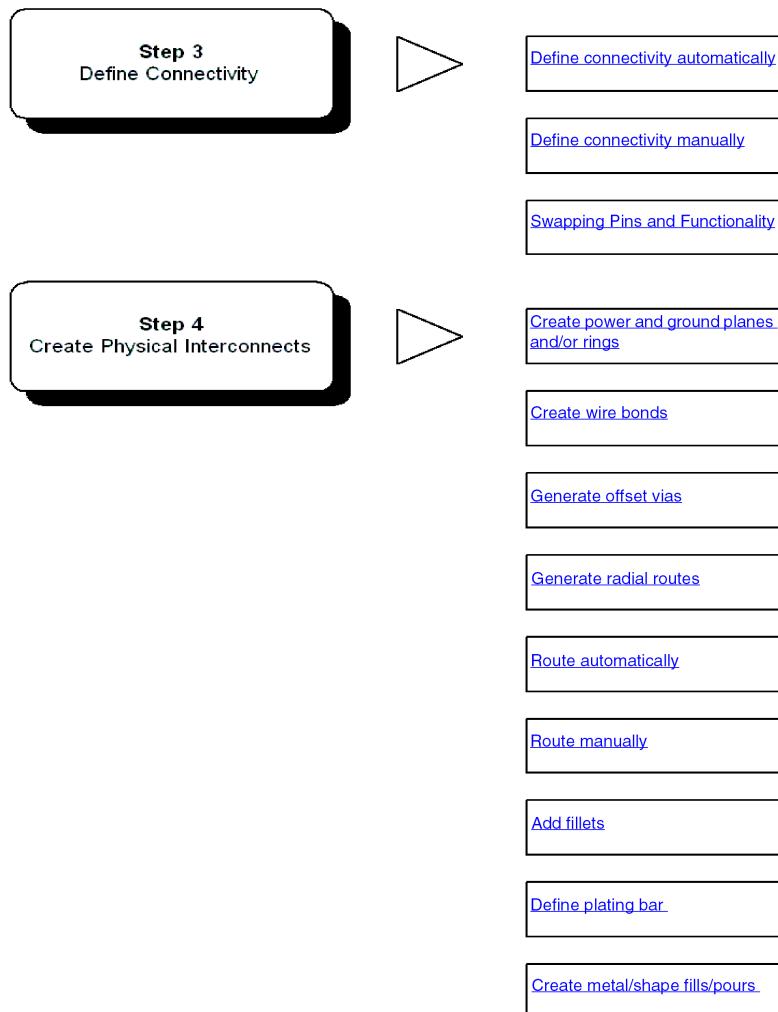
This section includes a general design flow describing the mounting of dies in a component using Allegro X Advanced Package Designer (APD). The component-design flow covers the design process from the import of data into an empty APD design to the export of manufacturing data from APD. The APD physical layout editor imports bare die geometry information from various sources. This information includes the number of die pins, die pin geometry, the die pins' X and Y locations, and the net name associated with each die pin and the die outline extents. The silicon designer can provide this information, ideally in electronic format. APD models the component format, and connects traces from the bare die to the component I/O pins.

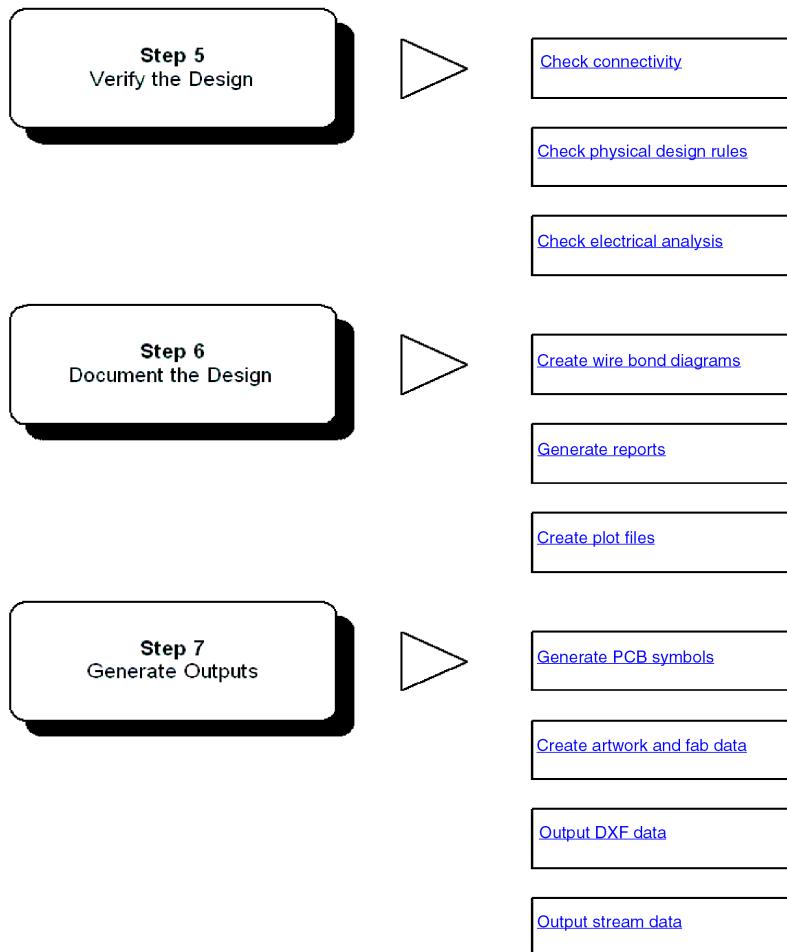
APD generates all necessary design data for component fabrication. If the fabricator uses APD, then the APD database can be used as a design-transfer mechanism. For critical or high-speed nets, you can use the Allegro Package SI (AP SI) analysis solution to analyze traces.

Completing a Component-Design Flow Using APD

Figure 2-3 shows the general steps for creating a component using APD. Hot links provide access to the descriptions of the specific tasks.

Figure 2.34: Steps for a Component-Driven Flow





Program Suite

When you install the tools on your computer, InstallScape allows you to choose between product tiers. Depending on the tier, different components are available.

A number of command-line utilities are also installed. These programs may display graphical user interfaces when run, or they may require that you enter arguments and options from the keyboard. These programs are documented in the appropriate sections of this user guide.

For more information about other products, see their respective user guides.

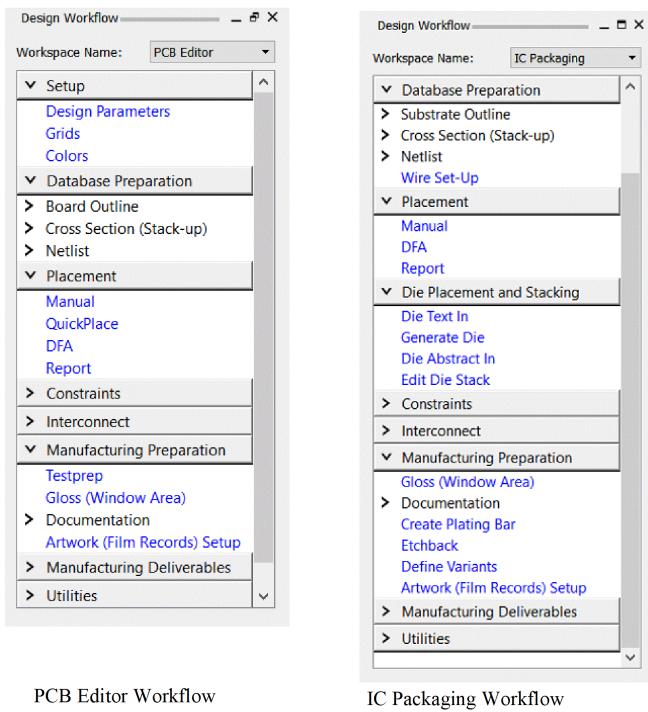
Design Workflow

Layout editors also provides design workflow as part of application user interface. The *Design Workflow* pane lists categories of various tasks performed during the design process.

Tool-specific design flows are available in the *Design Workflow* pane of the respective layout editors.

Getting Started with Physical Design

Getting Started--Program Suite

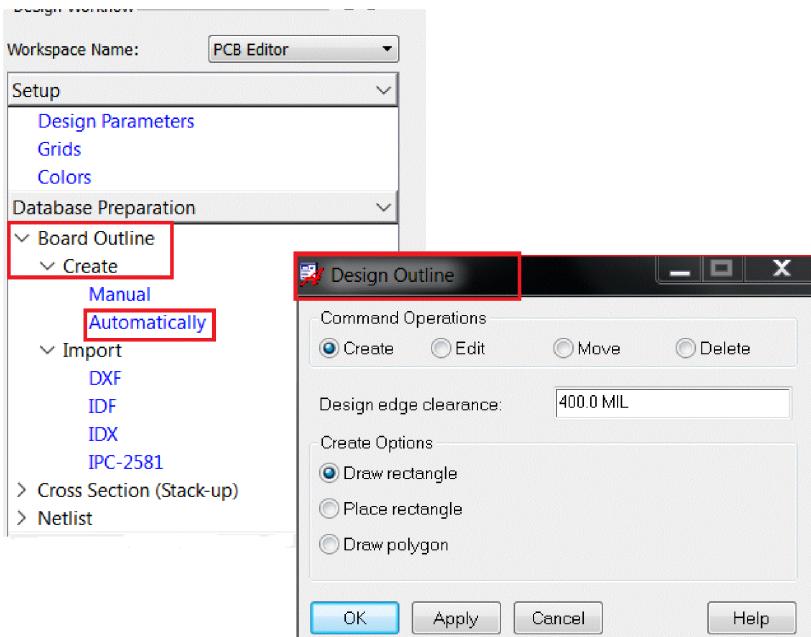


PCB Editor Workflow

IC Packaging Workflow

Figure 2-4 Design Workflows in Layout Editors

All the commonly used commands are listed under each task category. Clicking any task in the workflow pane opens the corresponding dialog box.



Layout editors finds the workflow files with its `WORKFLOWPATH` environment variable in the *Paths – Editor* section of the *User Preferences Editor*, available by choosing *Setup – User Preferences* (`enved` command).

Workflow files are XML files, which you can customize according to your requirement. The default workflow files are available at the following location:

```
<installation_directory>/share/pcb/text/workflows
```

To create a custom workflow, rename the name field and add, change, or remove workflow names and associated tasks in the `<workflow>.xml` file.

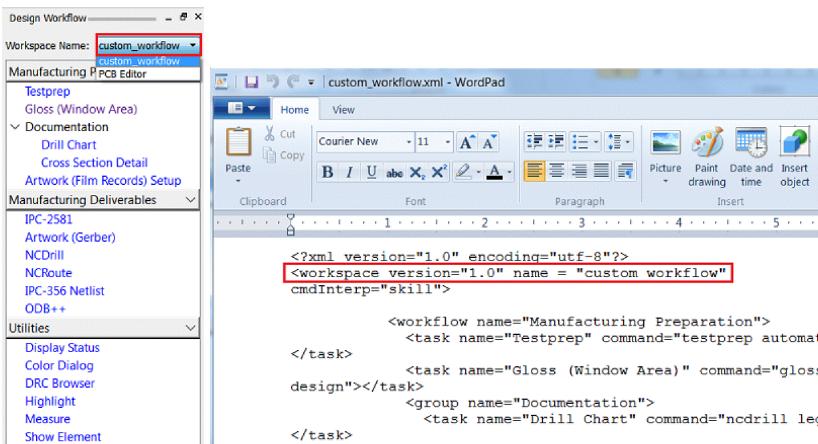


Figure 2-5 Customized Design Workflow

PCB Editor: Design Editing Modes

Allegro PCB Editor contains all the functions required for the layout, interconnect, design rule checking, testability and post processing of a printed-circuit-board design. You can start and run it as a stand-alone tool or as the layout portion of a complete design solution managed with Allegro Project Manager. For further information on Project Manager, see *Getting Started with Allegro PCB Design HDL* and the *Allegro Project Manager User Guide*.

The layout editor's workspace takes many forms — or design editing *modes*—depending on the type of design activity. This affords you the convenience of using a single, variable-mode editor to complete the design. The commands (menu picks and icons) available from the Allegro PCB Editor workspace change to reflect one of the following major design tasks:

- Layout creation and modification
Create the database in this editing mode. Use this mode to perform such tasks as component placement, board or design routing, and other functions.
- Symbol creation and modification
Create symbols for a design in symbol-editing mode. The tool appends the appropriate filename extension when you save a symbol.

You enter a design editing mode by specifying a file type when you choose *File – New* (*new* command) or *File – Open* (*open* command) from the editor. If you are running your layout editor on Windows, you can invoke the file from Windows Explorer (assuming you have set up a file association).

⚠ You must use Pad Designer (a padstack editor) to create or modify a library or design padstack. See [PCB Editor: Design Flow](#) for information on invoking the Padstack Designer.

Application Modes, the Pre-Select and the Post-Select Use Models

The layout editor lets you work in application modes, which provide an intuitive environment in which commands used frequently in a particular task domain, such as etch editing, are readily accessible from right mouse button pop-up menus, based on a selection set of design elements you have chosen.

This customized environment maximizes productivity when you use multiple commands on the same design elements or those in close proximity in the design. Application mode configures your tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the current selection set.

The layout editor comprises two menu use models, post-select and pre-select. The pre-select use model lets you choose a design element (noun), and then a command (verb) from the right mouse button pop-up menu. In the post-select use model, you first select the command and then the design element.

For example, in the pre-select use model the steps to move a symbol are:

- Choose the symbol to move
- Choose Edit - Move

Similarly, the steps to move a symbol in the post-select use model are:

- Choose Edit - Move
- Select the symbol to be moved

By default, the layout editor supports the pre-select use model. This pre-select use model lets you easily access commands based on the design elements you have chosen in the design canvas. The tool highlights the elements and treats them as a selection set, thereby eliminating extraneous mouse clicks and allowing you to remain focused on the design canvas.

In the pre-select use model, the command ends after the current operation completes. However, the post-select use model lets you perform the command on more than one design element until you choose *Done*.

Application Mode Types

When you initially launch layout editor, it defaults to the general-edit application mode, which lets you perform editing tasks such as place and route, move, copy and mirror.

The pre-selection model, or noun-verb model is enabled by default and allows access to any command provided that no element is currently selected. To work in menu-driven editing mode, that is, to choose a command and then the design element, click in a "black space" area of the design first. Commands not supporting the pre-selection use model ignore the selection set.

Mode Activation

Application mode can be activated in several ways. You can also activate the application mode through the right mouse button on the canvas or in the status bar.

- Choose a menu option:
 - *Setup – Application Mode – General Edit*: General-edit application mode lets you perform editing tasks such as place and route, move, copy and mirror.
 - *Setup – Application Mode – Placement Edit* (PCB Editor only) customizes your environment to fine-tune component placement and alignment. It also allows replication of circuits based on common connectivity and devices. A dockable placement window tab that appears in the *Options* window pane affords immediate access to useful placement options available in the `place manual` command. These include placement by group (that is, refdes, module instances, and so on) and filtering based upon property, room, part number, and net.
 - *Setup – Application Mode – Etch Edit*: Etch-edit application mode customizes your environment to perform etch-editing tasks such as adding and sliding connections, delay tuning, and smoothing cline or cline segment angles.
 - *Setup – Application Mode – Flow Planning*: Interconnect Flow Planner (IFP) application mode customizes your environment to perform route planning for complex (highly constrained, high pin-count) high-speed designs. For example, it enables you to bundle rats and perform bundle flow analysis.
 - *Setup – Application Mode – Signal Integrity* provides quick and easy access to frequently-used SI commands. The SI application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
 - *Setup – Application Mode – Symbol Edit* provides quick and easy access to edit changeable symbols, such as BGAs by adding, moving, deleting, changing, or swapping-pins, in a design. The Symbol Edit application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
 - *Setup – Application Mode – RFEdit* provides quick and easy access to RF command specific to the selected object or group of objects. This application mode configures the tool for a specific task by populating the right mouse button pop-up menu with RF commands that operate on the currently selected RF object or group of RF objects. In addition, the menu also displays some generic RF commands (not specific to the object or objects selected) under a Quick Utilities sub-menu of the RMB.
 - *Setup – Application Mode – Wire Bond Edit* configures the tool for wire bond specific tasks by populating the right mouse button pop-up menu with wire bond commands that operate on the currently selected items.
 - *Setup – Application Mode – Shape Edit* provides quick and easy access to edit the shape boundaries, such as sliding of shape edges with or without corners, multisegment sliding, and adding notches. The Shape-edit application mode configures the tool for a specific task by populating the right mouse button pop-up menu only with commands that operate on the currently selected element(s).
- Enter `etchedit`, `generaledit`, `ifp`, `placementedit`, `symboledit`, `shapeedit`, `signalintegrity`, `rfeedit_Appm`, `wbedit` in the Command Console window pane.
- Choose the appropriate toolbar icon (if added to your toolbar).



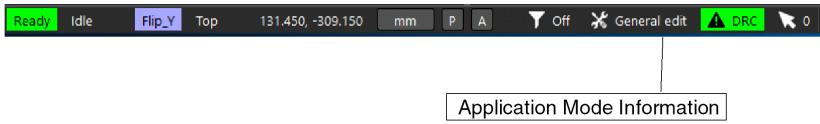
Use *Setup – Application Mode – None* (`noappmode` command) to exit from the current application mode and return to a menu-driven editing mode, or verb-noun use model, in which you choose a command, then the design element.

You can also use the `appmode` environment variable to control the application mode that launches on startup, which defaults to the application mode used on previous invocation of the tool.

Mode Verification

You can quickly check to see which application mode is active by hovering your cursor over the application box in the status bar.

Figure 2.35: Application Mode Status in Status Bar



Design Element Selection Model in Application Mode

As designs become denser, discerning a particular design element in a dense design may be difficult. To help you choose the correct element, hovering your cursor over an element highlights it and, a context-sensitive datatip that identifies the element appears next to the cursor. You can make datatip to appear above the Console Command window pane by setting the variable `datatips_fixedpos`.

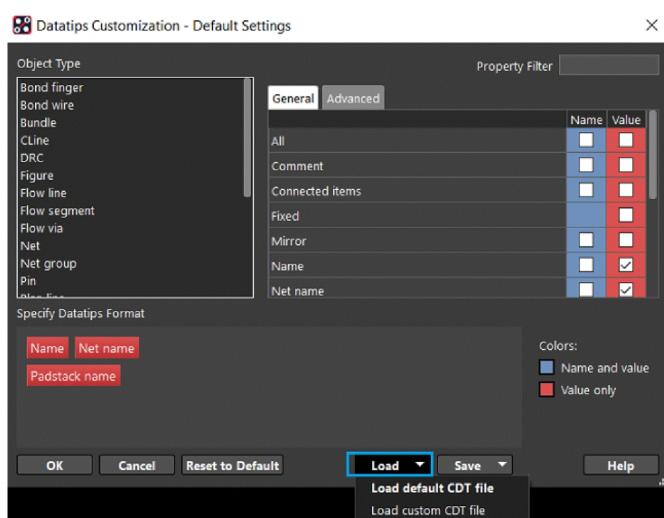


To disable the display of datatips, use the environment variable `disable_datatips`.

⚠ You can set the environment variable for datatips from the *Datatips* category in the *Display* section of the *User-Preference* dialog box.

Customizing Datatips

You can control the information that displays in a datatip for various objects like clines, nets, symbol instances, pins, vias, DRCs and so on by using *Setup – Datatip Customization* ([custom datatips command](#)).



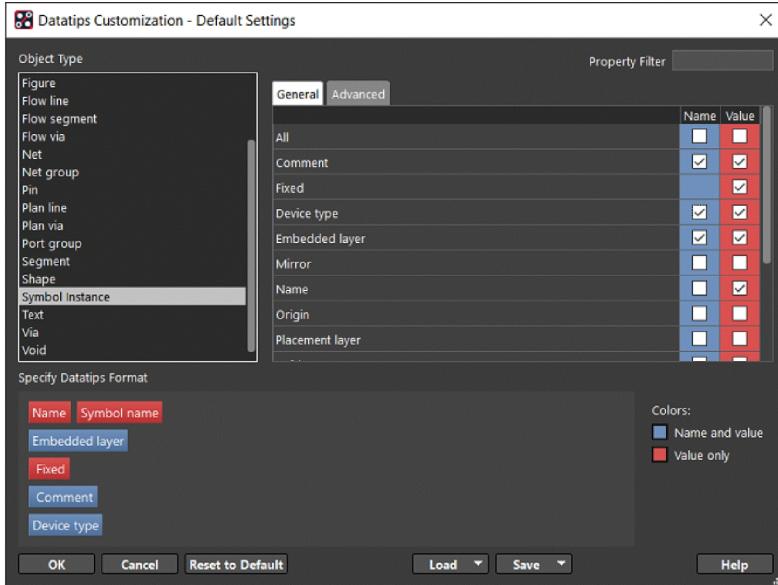
To save the datatip customization use *Save – Save default CDT file* button in the Datatips Customization - Default Settings dialog box. A datatip configuration file `custdatatips.cdt` is saved in the local `pcbenv` directory. You can import the local default CDT file using *Load – Load default CDT file* button.

You can reset the datatip customization to default by using *Reset to Default* button. It will load the .cdt file from the <installation_directory>/share pcb/text.

However, you can also create a customized .cdt file for a particular design, by saving customized datatips settings using *Save – Save custom CDT file* button. A datatip configuration file *custdatatips.cdt* is saved in the desired directory. You can then import the custom.cdt into another design using *Load – Load custom CDT file* button.

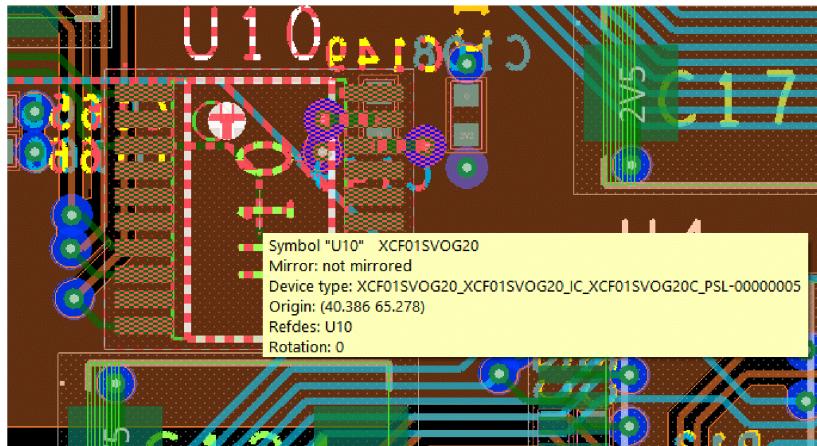
The *General tab* lists the information, available for display, about the element chosen in Object Type. The selected properties are added to the *Specify Datatip Format* field. The red labels indicate that only the *Value* will be displayed in the datatip and the blue labels will display both *Name and value* for the selected Object type.

Figure 2.36: Datatips Customization General tab and Specify Datatip Format



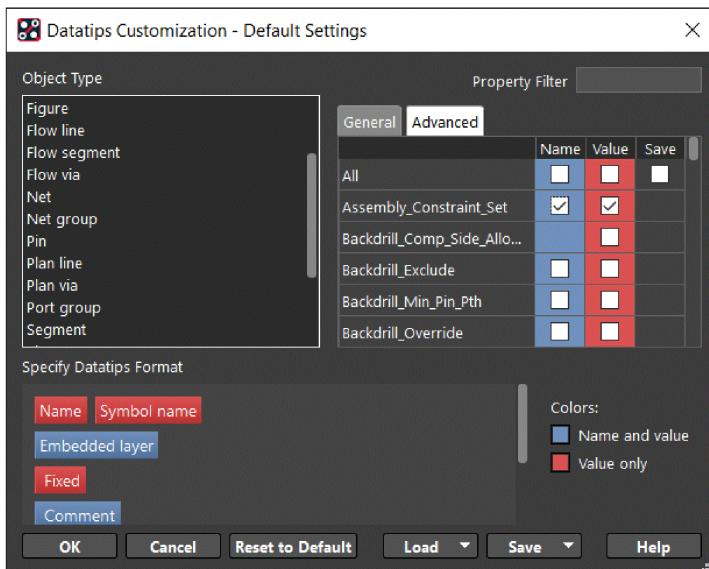
Following image displays the symbol instance datatip after customization:

Figure 2.37: Customized Datatip



The *Advanced* tab displays all properties applicable to the selected element and matching the string in property filter to available for inclusion in the datatip.

Figure 2.38: Datatips Customization Advanced Tab



The **Save** box appears next to user-defined attributes; select the save box to include the attributes in the `.cdt` file on saving it.

By default, a datatip disappears 250 milliseconds after the cursor leaves an object. You can, however, set the delay for datatip disappearance by setting the variable `datatips_delay`. For a custom datatip, you can set variable `custom_datatip_remove_delay`. In order to access the datatip features, move towards the datatip otherwise datatip will be removed.

You can turn the datatips on or off using icon in the Display toolbar.

Figure 2.39: Datatip toolbar icons



The dynamic display of datatips can be controlled by setting the variable `focus_followmouse`. The possible options are

Blank	Shows datatips only when the design canvas is in focus, that is, only when the design window is active.
Allegro_derived	Shows datatips when the sub-window is in focus, such as show element, reports and so on.
Anywhere	Shows datatips regardless of the focus. For example, datatip appears on pointing to an element on the design canvas even if any other application window is active.

Navigating Design Elements

While base elements such as cline segments, pins, and vias cannot be parents of other elements, they are the building blocks of hierarchical elements such as nets, clines, and components of which they are made. A pin is a child of a net, as well as that of a symbol and a function. Similarly, a cline could be a child of a symbol and a net. For a symbol with a shape containing a void, for example, the hierarchy may span five levels. The segment comprising the void has a hierarchy of Other Seg – Void – Shape – Symbol – Component.

If you enable more than one base or hierarchical element in the *Find* window pane, the base element determines the hierarchical elements you may choose. You navigate through the hierarchy by using the following or any other pre-defined hot keys:

- **Ctrl-Tab** for all base elements

⚠ The **Ctrl-Tab** key is unavailable when you select by window.

- **Tab** for all hierarchical elements

⚠ The **Tab** key is unavailable when you select by window, which chooses only top-level hierarchical elements.

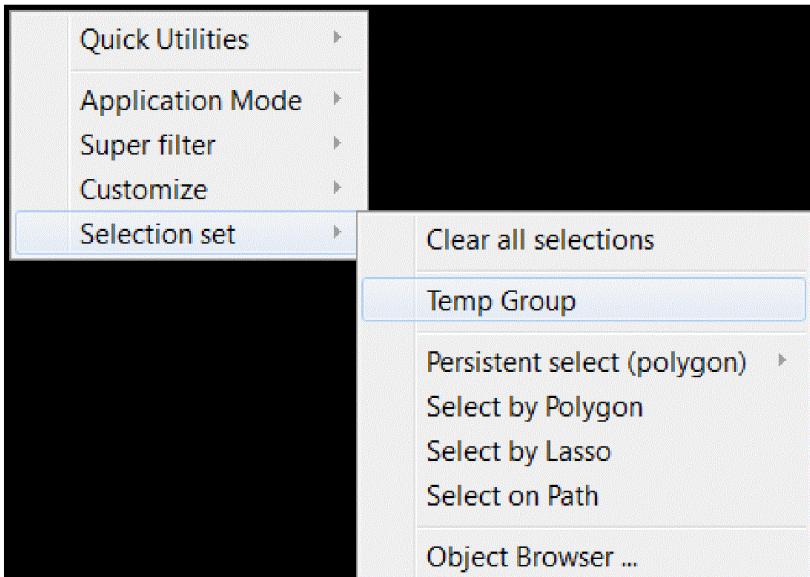
- The *Find* window pane to disable unwanted elements

The base element you initially chose remains highlighted in a different highlighting scheme.

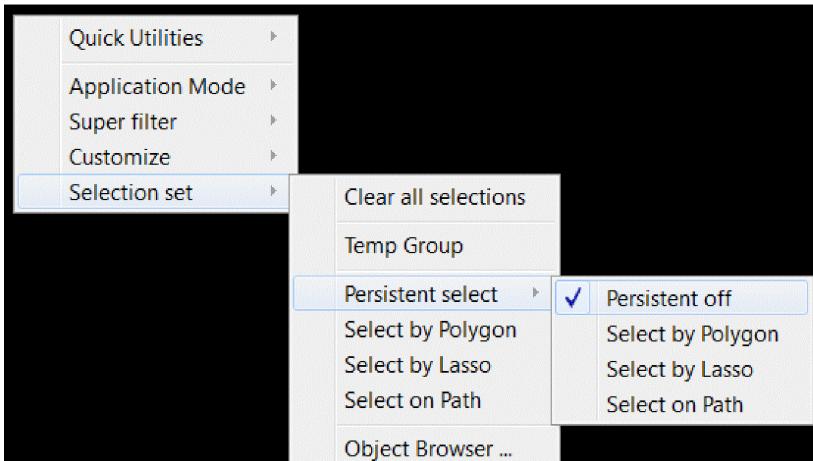
Using the Selection Set

The tool highlights design elements you've chosen in the design canvas as a selection set. Commands that operate on this selection set then appear on the right mouse button pop-up menu. If no elements are selected, when you right-choose and choose *Selection Set*, only the first five options appear; otherwise, the following depending on the number of item types selected and the cursor location (black space or hovering over elements):

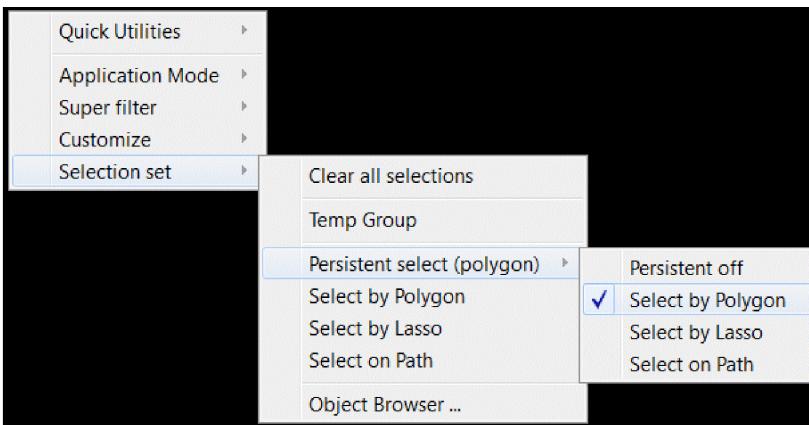
- *Clear All Selections* empties the selection set.
- *Temp Group* lets you choose the elements you want to group together. Each element you choose is highlighted. When you choose all the elements, right-click again to display the pop-up menu and choose *Complete*. All the selected elements are added to a preselect buffer.



- *Persistent select* lets you specify the selection mode (*Select by Polygon*, *Select by Lasso*, and *Select on Path*) for selecting multiple objects



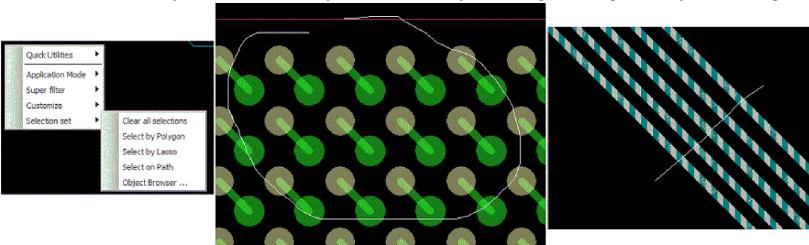
Enabling a mode for persistent select changes the menu and highlights the selected mode.



The active persist select mode remains unchanged during the session and applies to all the commands that access selection modes using right-click pop-up menus.

To turn off the persistent select mode, enable *Persistent off*. By default, the *Persistent select* is disabled.

- *Select by Polygon* lets you choose multiple elements by drawing a polygon around the elements during an editing session.
- *Select by Lasso* lets you choose multiple elements by drawing a open loop path around the elements during an editing session.
- *Select on Path* lets you choose multiple elements by drawing a straight line path during an editing session.



- *Object Browser* lets you search for elements by name or by property.
- *Select* appears only if elements are available to choose at the current mouse position.
- *Narrow Select* lets you refine your selection when multiple elements have been chosen during an editing session.
- *Toggle Select* lets you further refine the elements in the selection set after you select by window.
 - Clicking an element with a minus sign next to it removes it from the selection set.
 - Clicking an element with a plus sign adds it to the selection set.

You modify the elements in the selection set using the following.

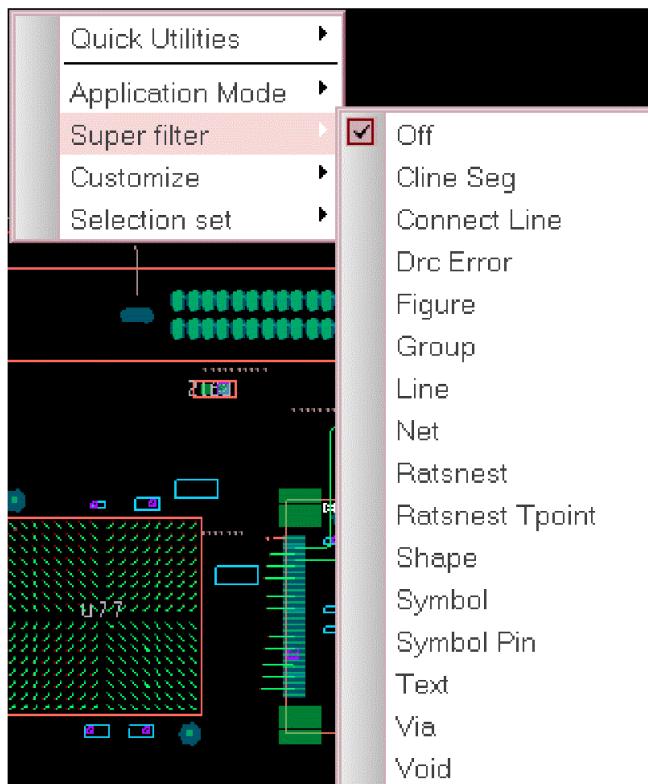
Click (single select)	Clears previous selection set and adds highlighted element at the mouse location to the selection set. If nothing is selectable at this location, clears previous selection set.
Shift + click (extend select)	Adds highlighted element at the mouse location to the selection set.
Ctrl + click (toggle select)	Adds the highlighted element at the mouse location if not already in the selection set. Removes the highlighted element from the selection set if the selection set already contains it.
Selection by window	Clears previous selection set. Adds elements enabled in the <i>Find</i> window pane and that overlap the window to the selection set.
Shift + Select by window	Adds elements enabled in the <i>Find</i> window pane and that overlap the window to the selection set.

Ctrl + Select by window	Removes elements: <ul style="list-style-type: none">• enabled in the <i>Find</i> window pane• overlapping the window• already in the selection set
-------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Choosing Design Elements with the Superfilter

The Superfilter lets you choose a particular element type to refine your selection set and temporarily disable all other elements from the right-mouse button pop-up menu rather than the *Find* window pane. By default, the Superfilter is set to *Off*. This means that all objects in the design are selectable (selection is unfiltered).

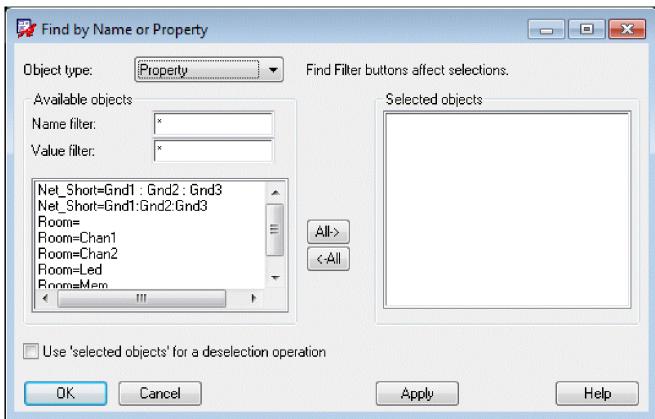
Figure 2.40: Superfilter in PCB Editor



Choosing Design Elements with the Object Browser

The Object Browser lets you select or de-select specific objects in the design by type, name or value. This selection method is particularly useful if the objects you want to operate on are difficult to see or are located on different layers of the design. To access the Object Browser right-click and choose *Selection Set – Object Browser*.

Figure 2.41: Object Browser



Default Hover-over Selection

In general-edit application mode, the highest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when your cursor hovers over an element.

In etch editing and IFP application mode, the lowest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights and becomes selectable when you hover the cursor over it. This is because the lowest level elements are most frequently used. Use the *Tab* key to navigate to other hierarchical level elements.

In placement-edit application mode, the lowest level hierarchical element enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when you hover the cursor over it, unless the element is a child of a symbol, or if a symbol is a member of a group. In these cases, the group assumes priority over the symbol, and the symbol assumes higher priority over the child element. Use the *Tab* key to navigate to other hierarchical level elements.

In shape-edit application mode, the lowest level hierarchical element (or segment) enabled by the *Find* window pane or *Superfilter* highlights by default and becomes selectable when you hover the cursor over it. In these cases, segment is a child of a shape. Use the *Tab* key to navigate between the segment and shape.

Context-sensitive pop-up menus

Application-mode commands are accessible from a right mouse button pop-up menu based on the current selection set. The commands that populate the pop-up menu depend on:

- Current application mode
- Design elements already in the selection set
- Design elements selectable at the current mouse position

Hovering your cursor over...	...populates the pop-up menu with
an element already in the selection set	commands applicable to the selection set.
an area where nothing is selectable, such as black space in the design	commands and functions independent of the selection set contents, which appear under <i>Quick Utilities</i> , such as <i>Undo</i> , <i>Design Parameters</i> , and <i>Grids</i> , as well as <i>Application Mode</i> , <i>Customize</i> , and <i>Superfilter</i> , depending on the current application mode. ⚠ In <i>Shape edit</i> application mode, four additional commands <i>Add Polygon</i> , <i>Add Rectangle</i> , <i>Add Circle</i> , and <i>Delete Islands</i> are available for creating shapes.

You can further filter all elements chosen during the current editing session by right-clicking and choosing *Select Set* from the pop-up menu, then *Narrow Select*. This is useful, for instance, when both base and hierarchical elements comprise the selection set, and you want to only include one of the hierarchical elements, such as symbols, in a particular area.

Common Options on the Pop-up Menus

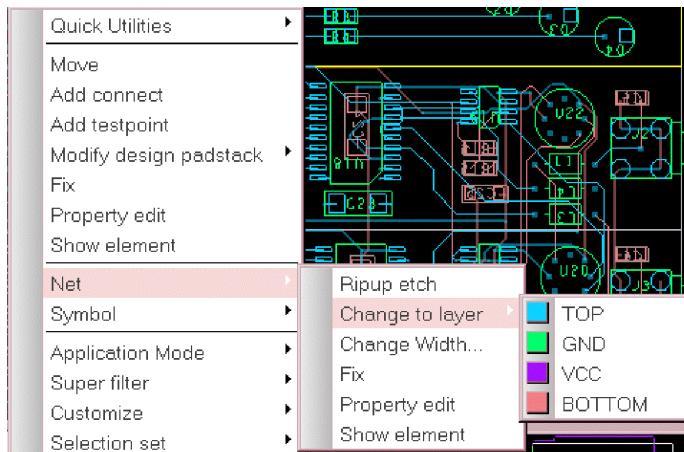
The right mouse button pop-up menus let you perform additional functions, and the available options vary:

- **Quick Utilities** allows access to frequently used functions, such as *Undo*, *Design Parameters*, *Grids*, *Global Shape Parameters*, and *Change Active Subclass*. These options are independent of the selection set contents.
- **Application Mode** lets you choose General Edit, Interconnect Flow Planner (IFP), Etch Edit, Placement Edit customized environments, or none.
- **Superfilter** confines your work to a particular element type, such as all nets, and disables the *Find* window pane.
- **Customize**
 - *Enable Single Click Execution* lets commands execute with a single rather than double-click, such as `add connect` in etchedit application mode
 - *Disable Automatic Drag Operations* initiates select by window rather than `slide` functionality
 - *Enable Shape Selection through Shape Fill*
 - *Reset to Defaults*
- **Selection Set**
 - *Clear All Selections* empties the selection set.
 - *Temp Group* lets you choose the elements you want to group together.
 - *Persistent select* lets you specify the selection mode (*Select by Polygon*, *Select by Lasso*, and *Select on Path*) for selecting multiple objects. By default, the *Persistent select* is off. The active persistent select mode remains unchanged and applies to all the commands that access selection modes using right-click pop-up menus. Select by Lasso
 - *Object Browser* lets you search for elements by name or by property.
 - *Select* appears only if elements are available to choose at the current mouse position.
 - *Narrow Select* lets you refine your selection when multiple elements have been chosen during an editing session.
 - *Toggle Select* lets you further refine the elements in the selection set after you select by window. Clicking an element with a minus sign next to it removes it from the selection set; clicking an element with a plus sign adds it to the selection set.

To work on a single element, hover your cursor over that element and then choose *Select – Select – <element>* from the pop-up menu, which also clears all previous selections.

If the selection set contains a mix of elements, the right mouse button pop-up menu displays pop-up submenus containing commands applicable to those elements.

Figure 2.42: PCB Editor: Selection Set Elements Determine Right Mouse Pop-up Menu Contents



If a command executes on a subset of the whole selection set or on hierarchical parents, corresponding elements append to the selection set and the others are ignored.

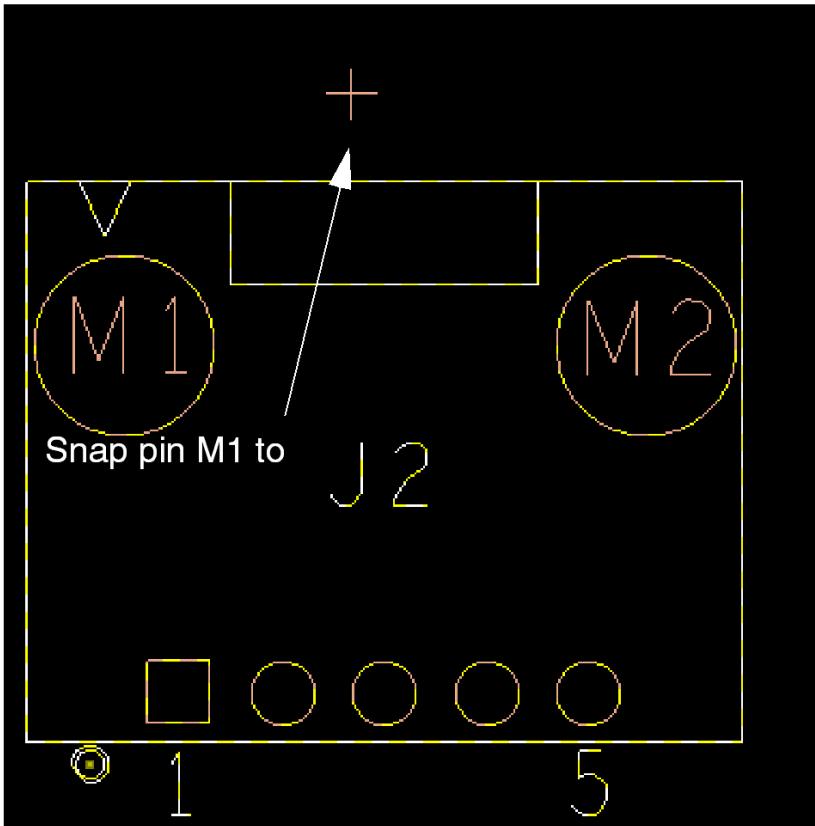
- **Snap pick to**, available in all application modes, lets you choose a snap mode from a list of options found on the right mouse button pop-up submenu when in an interactive editing command, for instance *Move* and *Copy*. On demand snapping lets you control both the starting pick point as well as the destination. The snap to point reaches from the current mouse position to the chosen snapping mode and is based on the chosen snapping mode, visibility, and object type. These parameters supersede the settings in the *Find* filter. It detects the nearest object in close proximity to the current mouse position. If no objects are available, snapping fails. A message appears in the command prompt window indicating that snapping was unsuccessful. The snapping mode is in effect for one pick event only and does not save the current snap mode for future sessions.

- Supported snapping modes include:

Snap Mode	Description
Persistent snap	Lets you select any of the <i>Snap pick to</i> options which will retain snapping for all further selections. Each pick of an operation will snap to the persistent option. The active snap mode applies to all the commands and application modes that performs snapping. By default, this option is off.
Segment vertex	The pick point snaps to the nearest vertex of any of the supported object types.
Segment Midpoint	The pick point snaps to the nearest mid-point of the supported segment type.
Segment	The pick point snaps to the nearest point on the segment.
Intersection	The pick point snaps to the intersection of the segments.
Shape Center	The pick point snaps to the center of a shape.
Arc/Circle Center	The pick point snaps to the center of an arc segment or a circle.
Symbol Origin	The pick point snaps to the origin of the symbol.
Symbol Center	The pick point snaps to the center of a symbol.
Pin	The pick point snaps to the nearest pin.
Via	The pick point snaps to the nearest via.
Figure	The pick point snaps to the nearest figure.
Pad Edge Vertex	The pick point snaps to the nearest vertex of the pad edge.
Pad Edge Midpoint	The pick point snaps to the nearest mid-point of the pad edge.
Pad Edge	The pick point snaps to the nearest point on the pad edge.
Off-grid Location	The pick point directly returns without snapping.
Grid point	The pick point snaps to the nearest grid point.
Snap Offset(0.00 0.00)...	<p>The pick point snaps at a given offset to one of the selected snap destinations. You can define the offset from a snapping destination in two ways by setting either:</p> <ul style="list-style-type: none"> ■ X and Y coordinates ■ or distance and angle <p>If <i>Persistent snap</i> is enabled, the <i>Snap Offset</i> values are remembered throughout the editing session and for all the commands that uses snapping.</p>

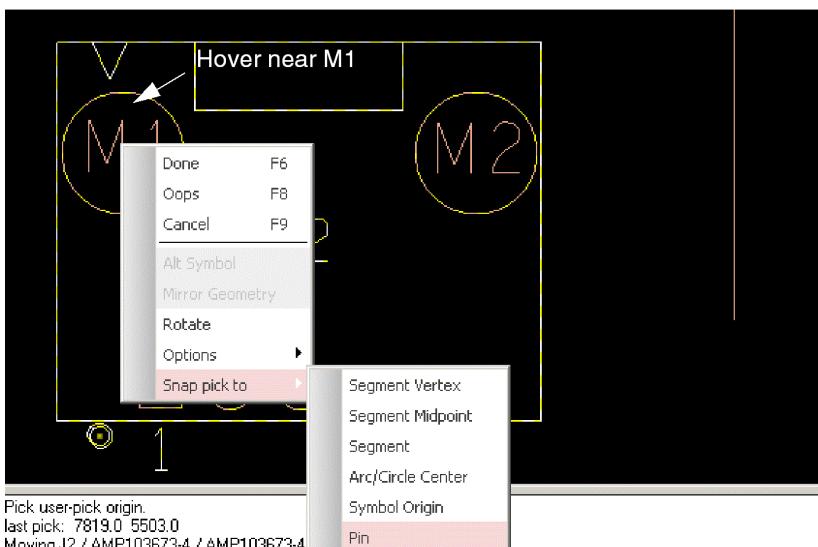
On demand snapping involves the following basic steps for all interactive commands. This example uses the *Move* command to illustrate snapping a mechanical pin associated with a connector to a cross hair target. The *Intersection* option represents the snap object for the cross hair (Figure 2-14).

Figure 2.43: Snapping To a Cross Hair Target (PCB Editor)



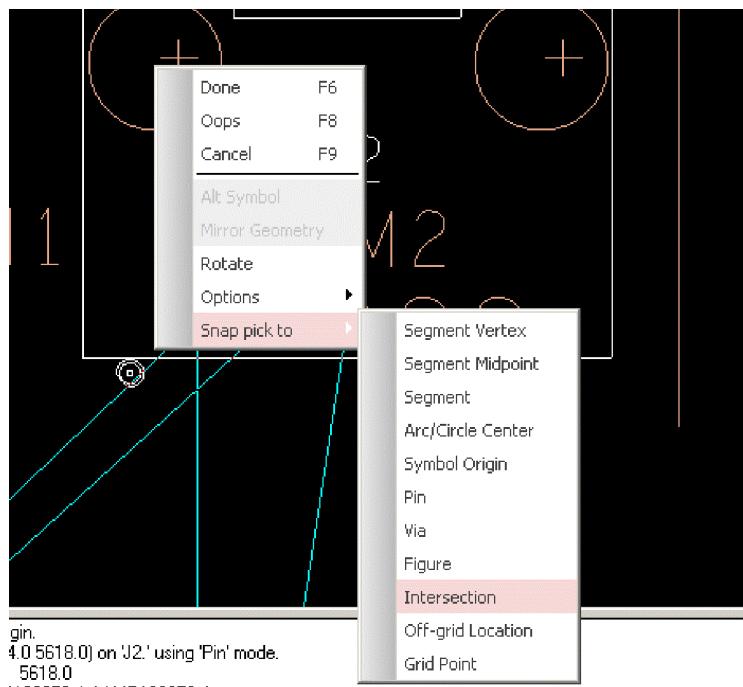
Selecting the mechanical pin (M1) as the initial snap object requires a setting change associated with the *Move* command. From *Edit – Move*, change the rotation point, normally set to *Body Center* to *User Pick*. Next click on the connector when the message: `Pick user-pick origin` appears in the command window prompt. Hover over `M1` and then use the right mouse button popup menu option *Snap pick to – Pin* ([Figure 2-15](#)).

Figure 2.44: Snapping a Mechanical Pin to a Cross-Hair Target (PCB Editor)



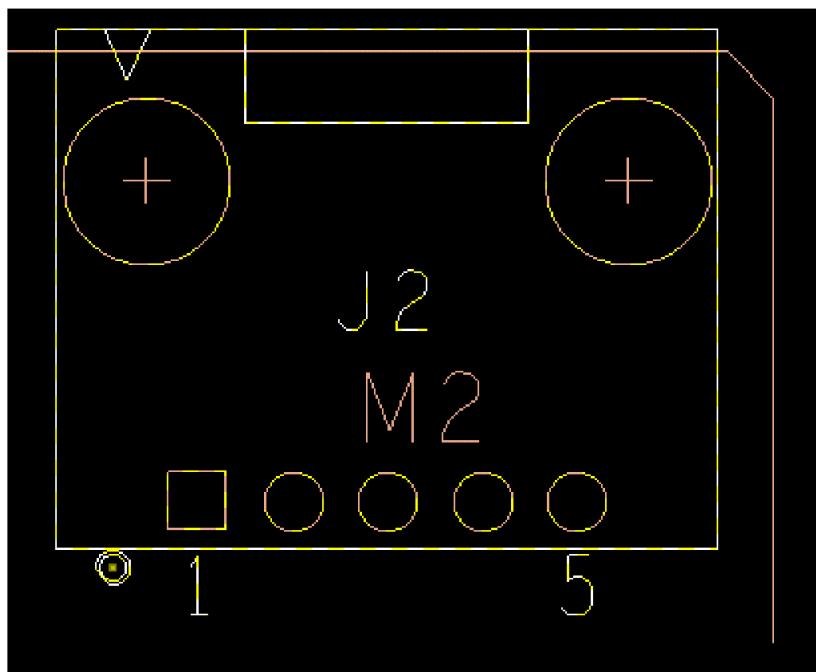
Move the connector with your cursor locked on M1 towards the location of the cross hair target and then use the right mouse button pop-up menu option *Snap pick to – Intersection* to snap the mechanical pin to the target ([Figure 2-16](#)).

Figure 2.45: Snap Pick to Intersection (PCB Editor)



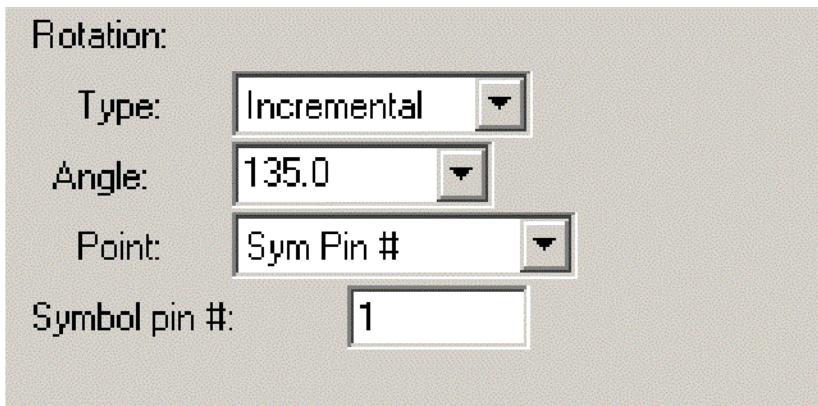
The end result is the center of the mechanical pin on the connector snaps to the intersection of the lines of the crosshairs. A message appears in the command prompt window indicating that snapping is complete

Figure 2.46: Completed Snapping Operation



You can use the same procedure as in this example to snap any pins on a connector to the target. For example, to snap Pin 1 of the connector to the target, change the rotation point associated with the Move command from User Pick to Sym Pin # and then enter a value of 1 in the Symbol pin # field (Figure 2-18).

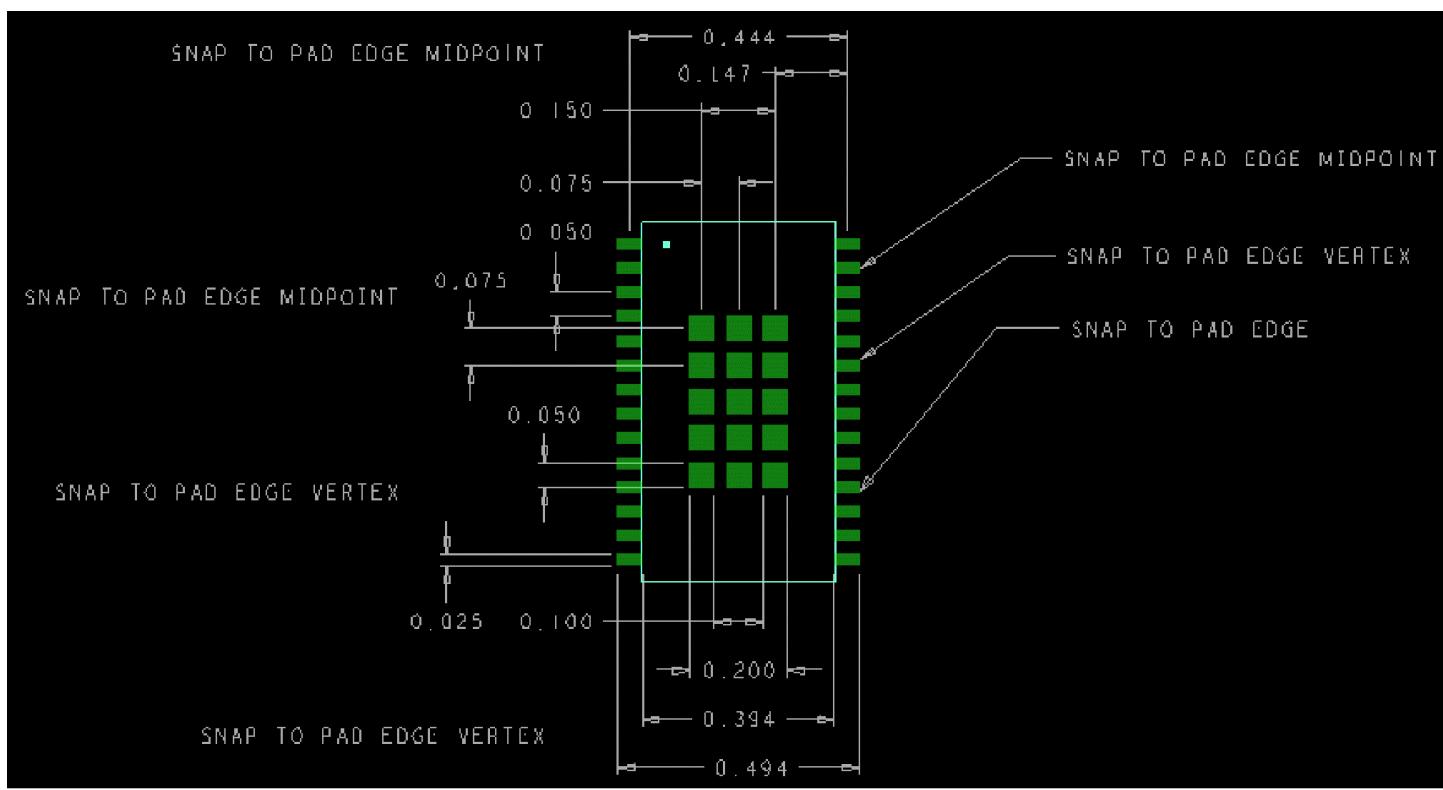
Figure 2.47: Changing the Rotation Point



Example

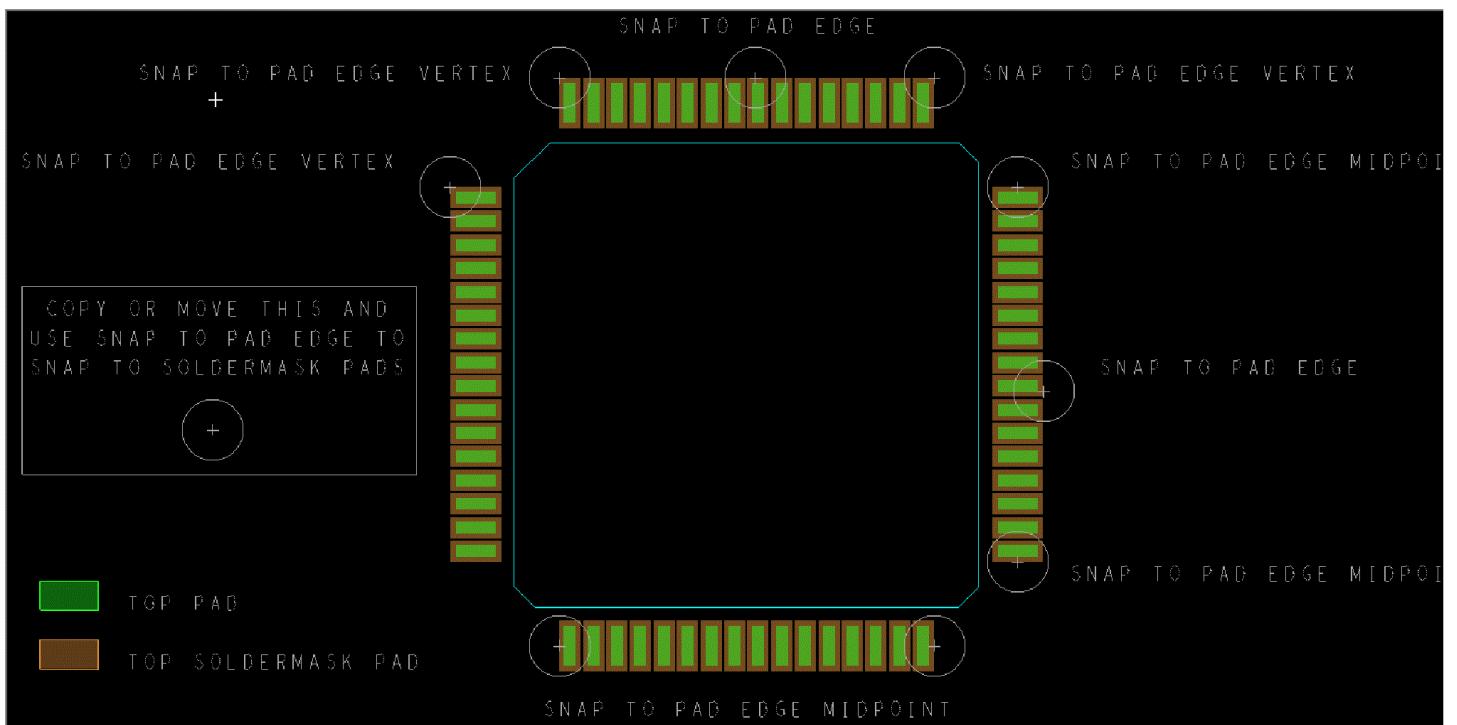
Snap Pick to Pad Edge options in Dimensioning Environment

Dimensioning using *Snap Pick to Pad Edge*, *Snap Pick to Pad Edge Midpoint*, and *Snap Pick to Pad Edge Vertex* snaps to objects that are on the conductive subclasses.



Snap Pick to Pad Edge options with Move and Copy commands

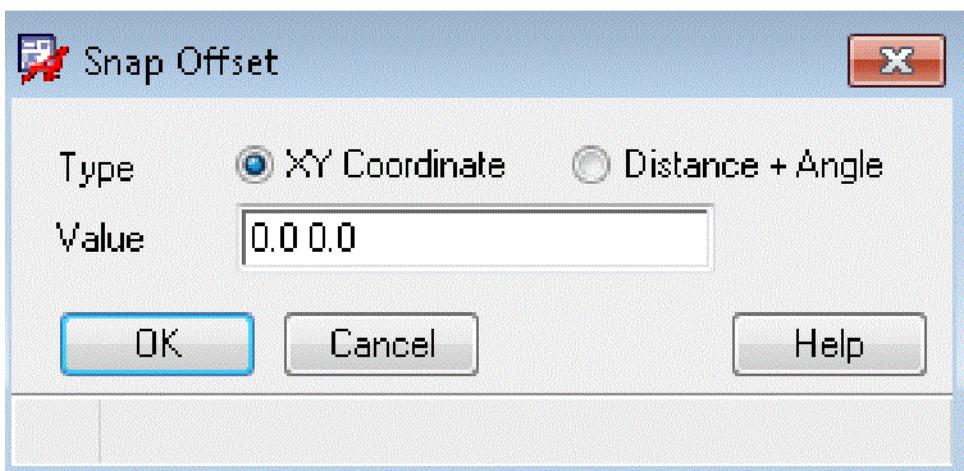
You can use *Snap Pick to Pad Edge*, *Snap Pick to Pad Edge Midpoint*, and *Snap Pick to Pad Edge Vertex* to snap objects to pads on any subclass. In the following figure, change active class and subclass before executing the `copy` and `move` commands, to select pad edges on Pin class.



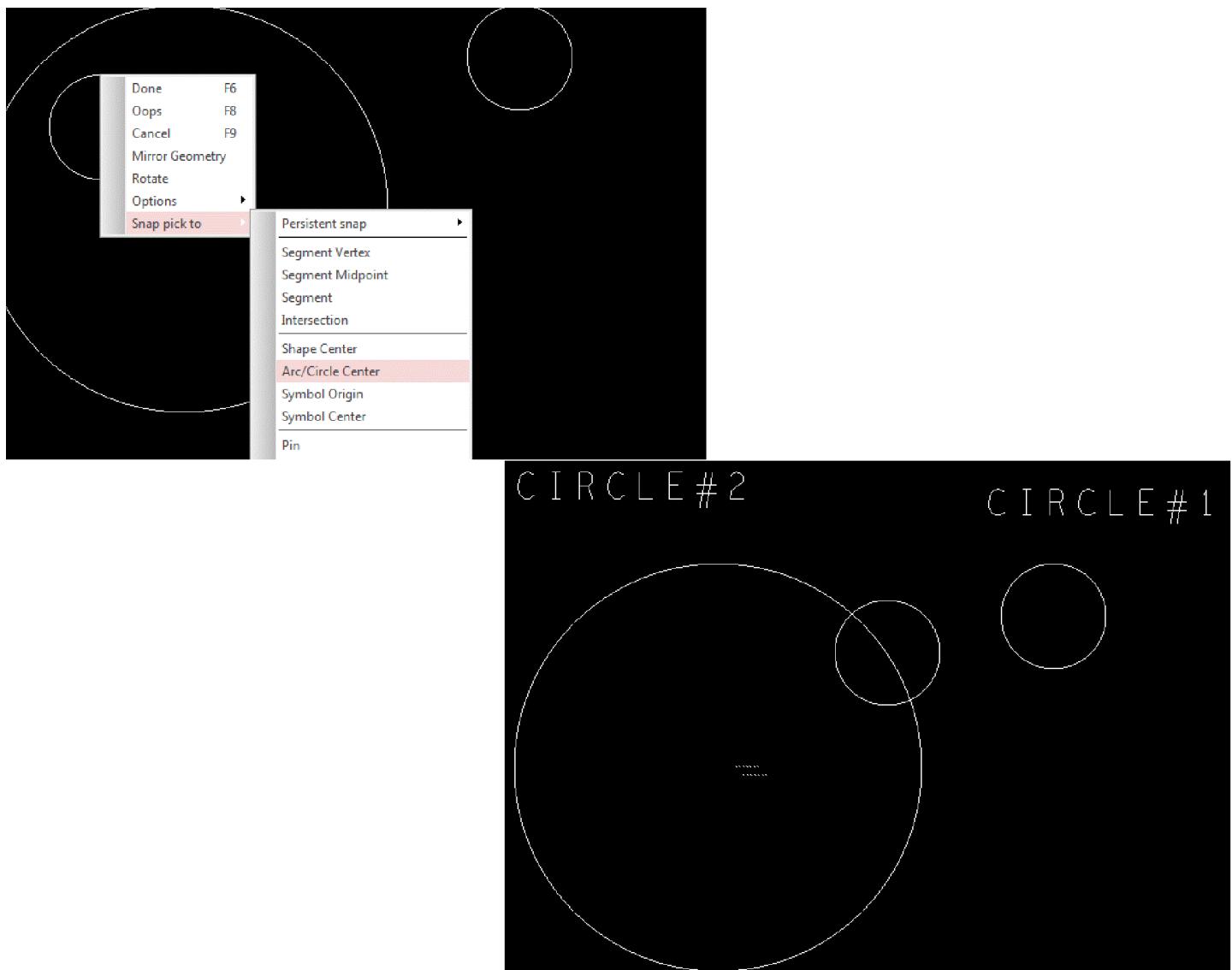
Snap Object at an Offset

You can use *Snap Pick* to along with *Snap Offset* to snap objects to a specific destination and at a defined offset. You can define the offset either by setting X and Y coordinates or by setting a distance and an angle.

For example, copy a circle (circle#1) and snap it to the center of another circle (circle# 2) at an offset. Use `copy` command to copy circle#1, right-click and choose snap mode as *Snap Offset*. Set the offset of (1000.0 1000.0).



Place the cursor at the circle#2, right-click and choose the *Snap pick* mode as *Arc/Circle center*.



Application Mode Default Command Execution

Depending on the current application mode, you can automatically execute a default command with a click, double-click, or a drag-and-drop operation on an element.

Default commands that execute with a click, double-click, or drag-and-drop operation depend on the following:

- Element chosen with the double-click.
- Current application mode; for example, when in etch-edit mode, single clicking a pin executes *Route – Connect* (`add connect` command).

Etch-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements. Single-click execution is enabled by default, which you can disable by right-clicking and choosing *Customize* from the pop-up menu.

Element	Drag	Shift Drag	Ctrl Drag	Shift Ctrl Drag	Double-click
Group	move	move	copy	none	none
Symbol	move	spin	copy	none	move

Pin	none	none	none	none	add connect
Via	slide	move	copy	none	add connect
Cline*	move	move	copy	none	none
Line	move	move	copy	none	none
Shape	move	move	copy	none	none
Frect	move	move	copy	none	none
Rect	move	move	copy	none	none
Line Seg*	slide	none	delay tune	none	slide
Arc Seg	slide	none	none	none	slide
Figure	move	move	copy	none	none
Text	move	move	copy	none	none
Ratsnest	none	none	none	none	add connect
Rat T	slide	move	none	none	none

*Choosing the midpoint of a cline or line seg invokes the `slide` command; to invoke the `add connect` command from the midpoint of a cline or line seg, right-click and choose `add connect` from the pop-up menu.

General-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements in general-edit application mode. Single click execution is enabled by default, which you can disable by right-clicking and choosing *Customize* from the pop-up menu.

Element	Drag	Shift Drag	Ctrl Drag	Shift Ctrl Drag	Double-click
Cline	move	move	copy	none	none
Cline_seg	slide	none	none	none	none
Component_inst	none	none	none	none	none
Figure	move	move	copy	none	none
Drc_error	none	none	none	none	none
Function_inst	none	none	none	none	none
Group	move	move	copy	none	none
Line	move	move	copy	none	none
Net	none	none	none	none	none
Other_seg	none	none	none	none	none
Ratsnest	none	none	none	none	none
Rat_t	slide	move	none	none	none
Shape	move	move	copy	none	none
Symbol_instance	move	spin	copy	none	none
Text	move	move	copy	none	none
Var_pin	none	none	none	none	none
Via	slide	move	copy	none	none
Void	none	none	none	none	none

PCB Editor: IFP Application Mode Automatic Command Execution

The following commands execute by default with a double-click or drag-and-drop operation on these design elements in IFP application mode.

⚠ This feature applies only to Allegro PCB Editor.

To...	Position your cursor here...	Press and hold this key...	Use this mouse action...
Insert and position a new flow vertex.	Over a flow line segment	n/a	Depress the left button and drag-and-drop
Slide an existing flow line segment.	""	Shift	""
Insert a new flow via.	""	n/a	Double -click
Move an existing flow line vertex.	Over a flow line vertex	Shift	Depress the left button and drag-and-drop
Slide an existing flow line vertex.	""	n/a	""
Move an existing flow via.	Over a flow via	Shift	""
Slide an existing flow via.	""	n/a	""
Remove a flow via.	""	n/a	Double -click

Placement-edit Application Mode Automatic Command Execution

The following commands execute by default on these design elements in placement edit application mode.

Element Type	Drag	Shift Drag	Ctrl Drag	Single Click
Group	Move	Spin	Copy	Move
Symbol	Move	Spin	Copy	Move
Text	Move	Spin	Copy	Move
RatT	Move			Move

Shape-edit Application Mode Automatic Command Execution

The following commands execute by default with a double-click or drag-and-drop operation on these design elements in shape-edit application mode. Single click execution is enabled by default, which you can disable by right-clicking and choosing Customize from the pop-up menu.

Element Type	Drag	Shift Drag	Single Click
Segment	Move and Slide	Move	Move and Slide
Vertex	Move and Slide	Move	Move
Shape	Move and Slide	Move	Move

Support for the `undo` and `redo` Commands

Using the `undo` command preserves the selection set that existed when you initially launched the command whose results you subsequently have reversed.

About the User Interface

The Allegro layout editor features a task-oriented user interface, with the following components:

- *Start Page*
- *Design Window*
- *Menu Bar*
- *Toolbar*
- *Control Panel* with these foldable window panes

- Options
- Find
- Visibility
- Command foldable window pane
- WorldView foldable window pane
- Status bar
- About Window

 Wherever user input is required, the interfaces support English (United States) format for numbers.

The design is by default appear in dark theme except for the title bar. To change the theme, set the environment variable `allegro_theme` in the User Preferences Editor and restart the application.

Start Page

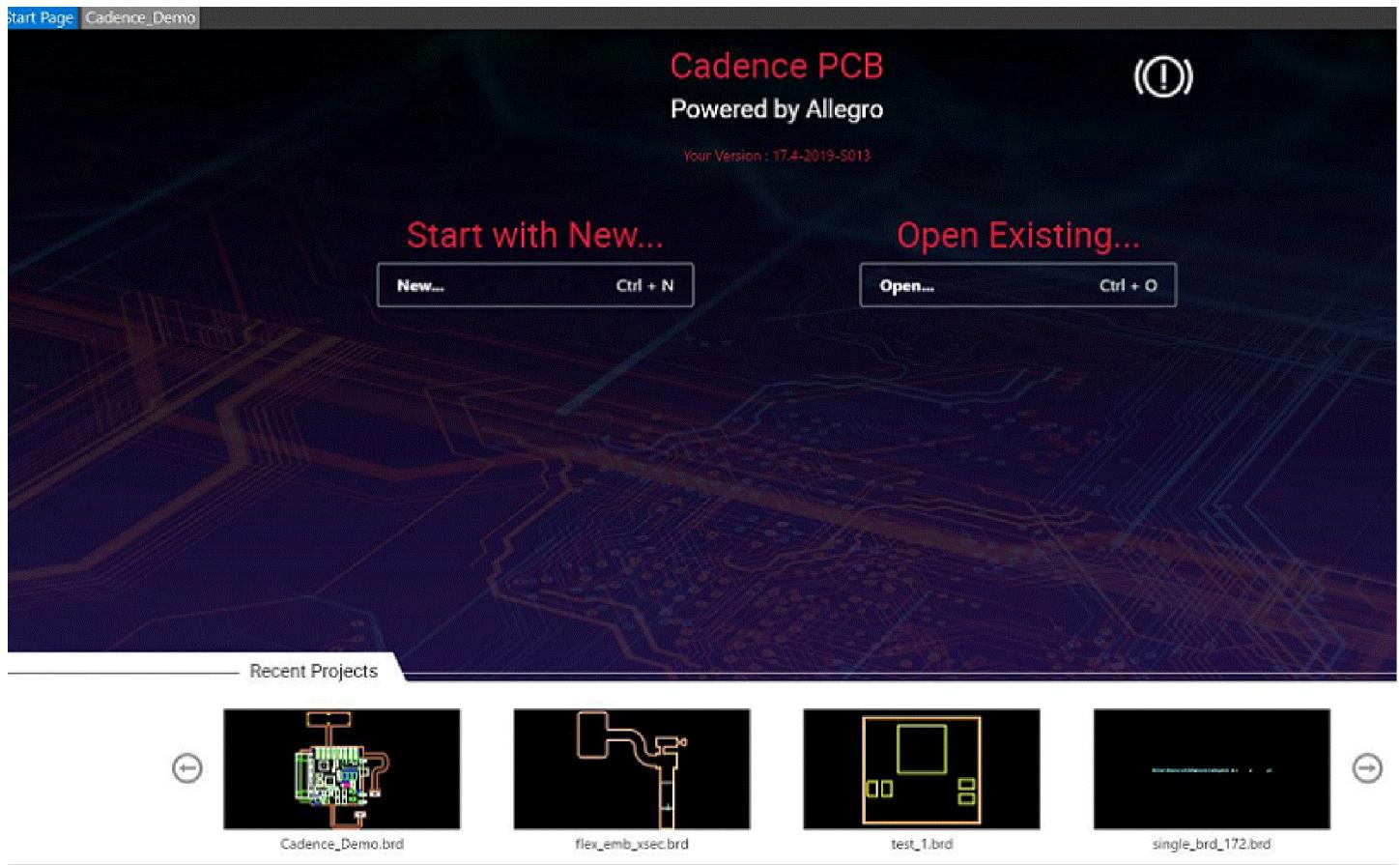
On starting a layout editor, you see a renewed, content-rich, and reorganized start page. Using the start page, you can easily access a variety of information, and projects. From this page, you can read about layout editors, go through brief descriptions of the available features, and access quick start guides and video walkthroughs. You can also access help content, product announcements, and industry news. The page also provides contact information for connecting to Cadence customer support.



The *Start Page* tab provides options to create a new design or to open existing designs. The *Recent Designs* section provides an easy access to recently opened designs. Click the design name to open it in a new tab. The *Recent Designs* tab displays a quickview of the designs that are saved in the current

release. The designs created in earlier releases can be accessed through buttons only and no quickview is available.

When you are working offline, without an internet connection, only static content of the start page is available.

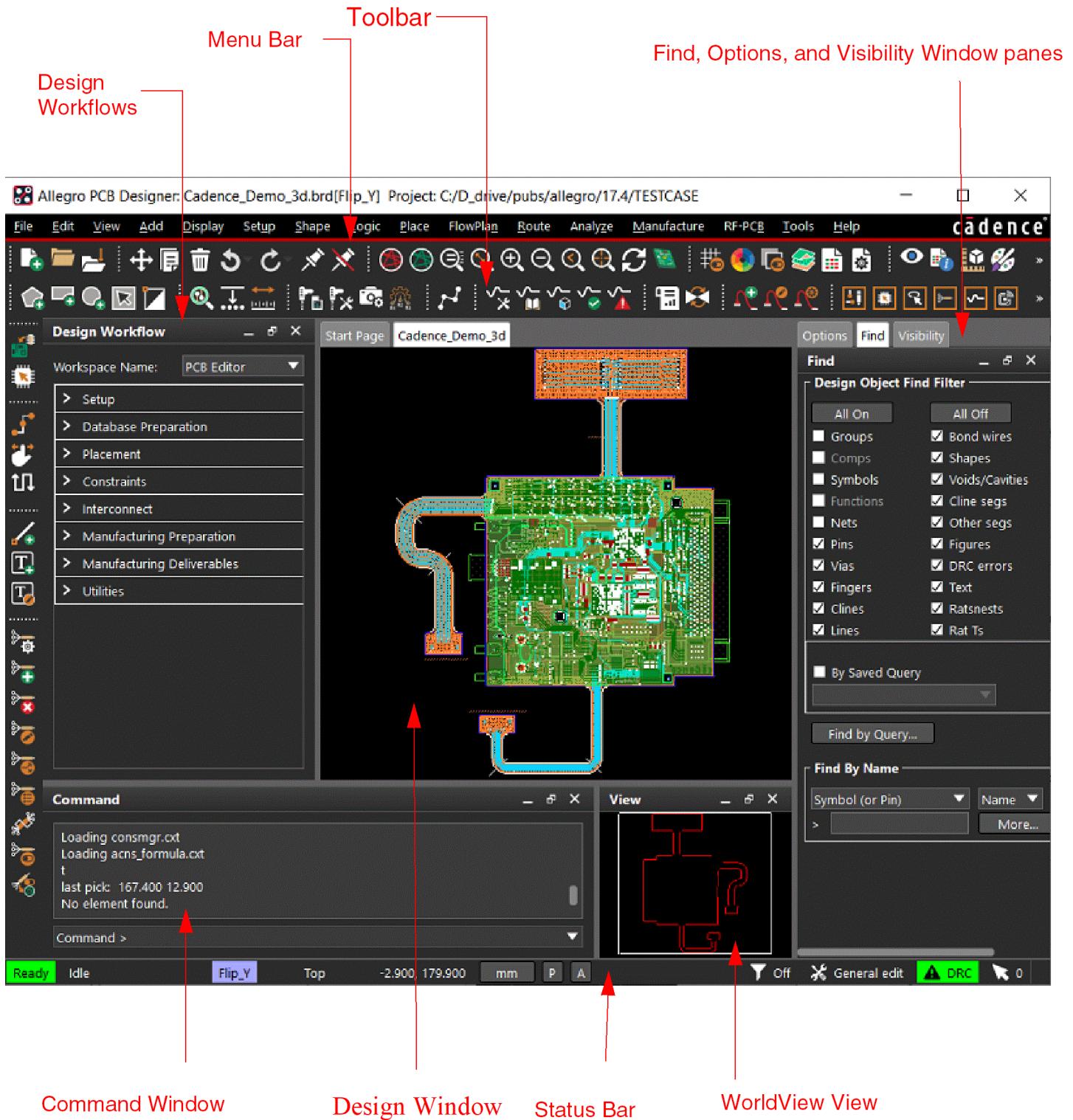


The Design Window

The Design Window is where you create a design.

When you are reviewing logs or reports using the Viewlog and Show Element commands, you can click on coordinate values within these files and zoom center on the corresponding locations in the design window. For additional information, see [viewlog](#) and [show element](#).

Figure 2.35: PCB Editor: User Interface



Panning the Design Window

You can remain at a zoomed-in view, and move the design window across a design in any direction.

You can pan a design using a mouse or arrow keys on the keyboard.

There are several ways to pan in a design:

1. Place the cursor in the editor window. Press and hold the middle mouse button down and slide the mouse to the left, right, up, and down.
2. Use the arrow keys on your keyboard to pan the design. To control the amount of panning using the arrow keys:
 - a. Select Setup – User Preferences
 - b. Select Display/Roam in the Categories section
 - c. Set a value for the *roaminc* environment variable and select *OK*

The Menu Bar

The pull-down menus in the menu bar provide all of the commands that you need to create or modify a design. The menu command sets (*Layout*, *Symbol*) that are available to you depend on the task that you are performing and the tool that you are running.

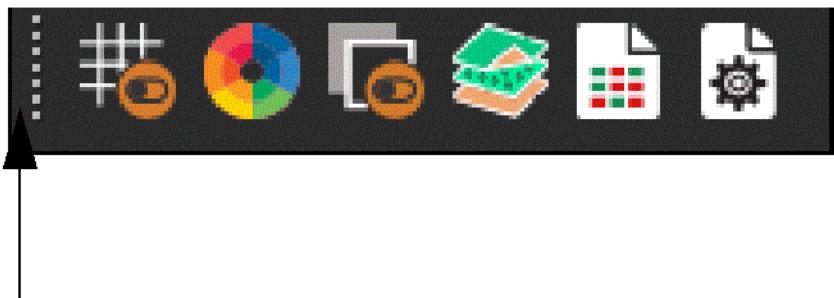
You can also use the accelerator key combinations to execute the command. The key combinations appear in the pull-down menu, to the right of the command.

The Toolbar

The toolbar contains functionally related icons, such as those for routing or placement, to access common commands. To learn a toolbar icon's function, position the cursor over the icon without depressing the mouse button and view its description in the tool tip that appears. Icons can be customized to suit specific needs.

Dock or undock any toolbar by left-clicking on the small circles, or grippers, next to it and moving it. The size of toolbar buttons are fixed and remain the same when the toolbar is expanded.

Figure 2.36: Grippers



Grippers

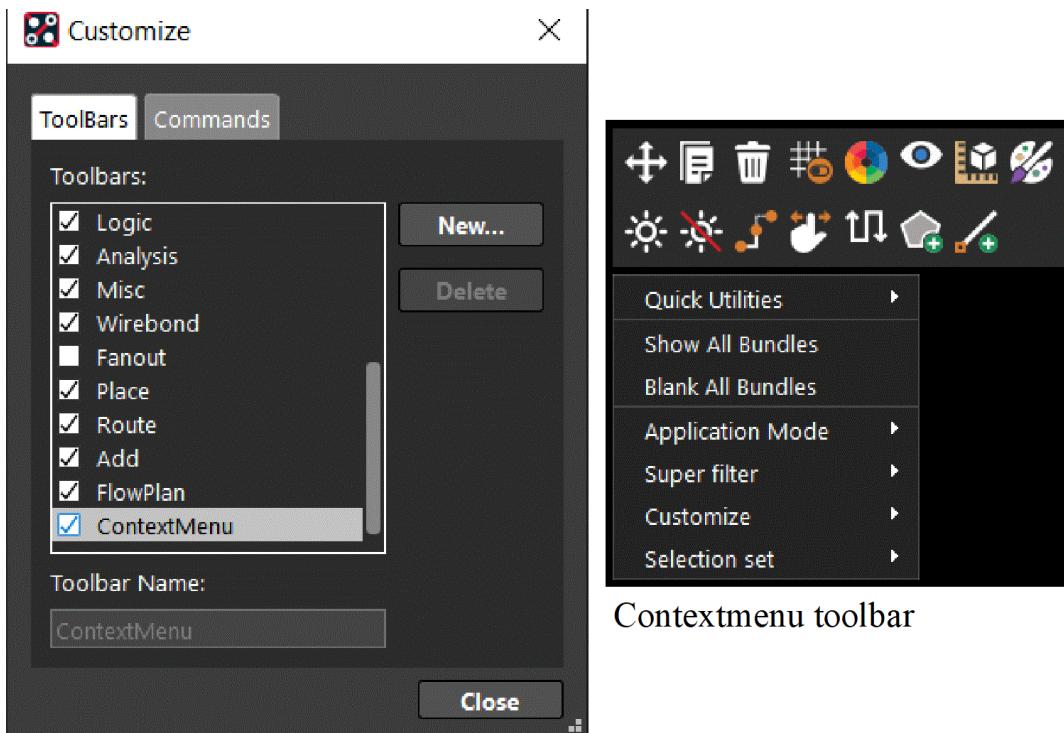
Customizing the Toolbar

You can customize the toolbar with icons or commands of your choice. New toolbars can be created and custom icons can be assigned to commands. Choose *View – Customize Toolbar* to open the Customize dialog box. A *ContextMenu* toolbar is available to assign frequently-used icons. This toolbar is available on right-click context menu. A maximum of 16 icons can be added.

⚠ The *ContextMenu* is not available in Linux.

In the *Toolbars* tab, checked the toolbars you want as part of your workspace.

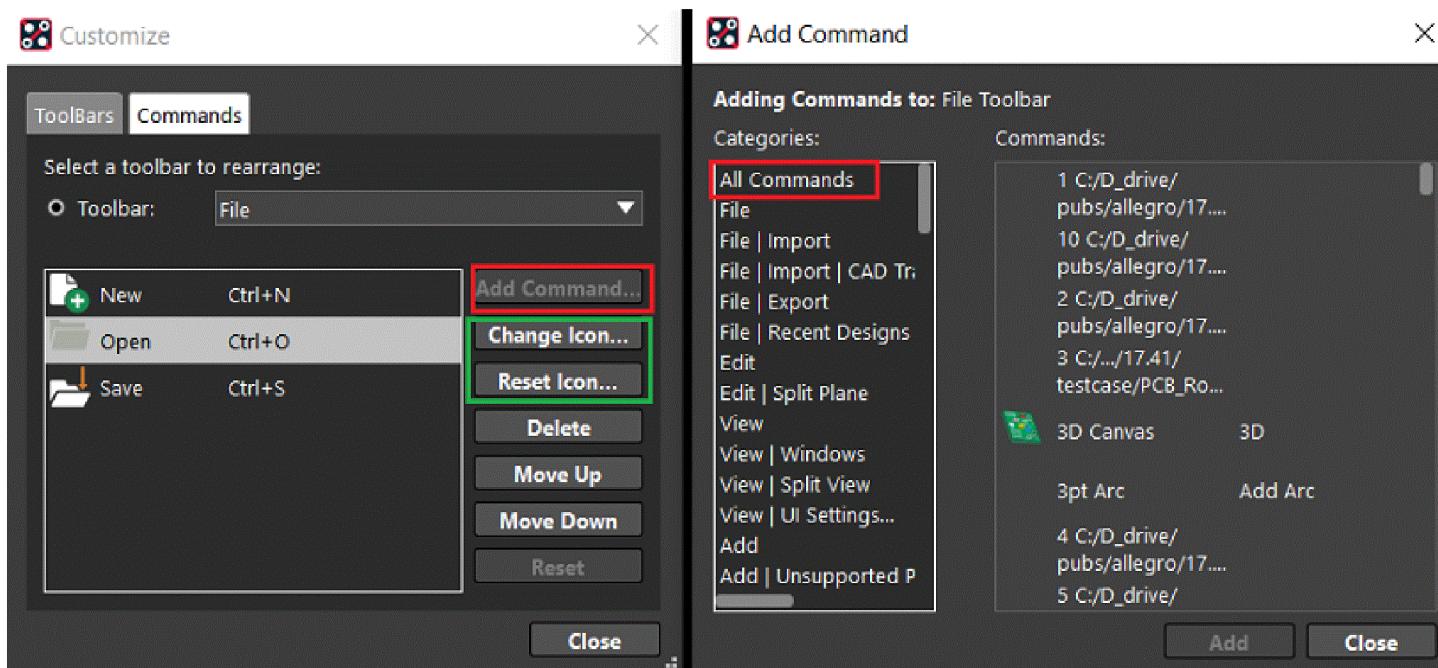
Figure 2.37: Customize Dialog Box for Toolbars



ContextMenu toolbar

In the *Commands* tab, customize individual toolbars. Select the toolbar and then click one of the buttons to add a new command (*Add Command*), delete an existing command (*Delete*), move a command up (*Move Up*), move a command down (*Move Down*) or reset the all changes (*Reset*). The *Add Command* button opens the *Add Command* dialog box, where you can select a category and a command under that category to add to a toolbar. A complete list of all the commands is available, which is sorted by command name, not by menu or submenu or order in menu.

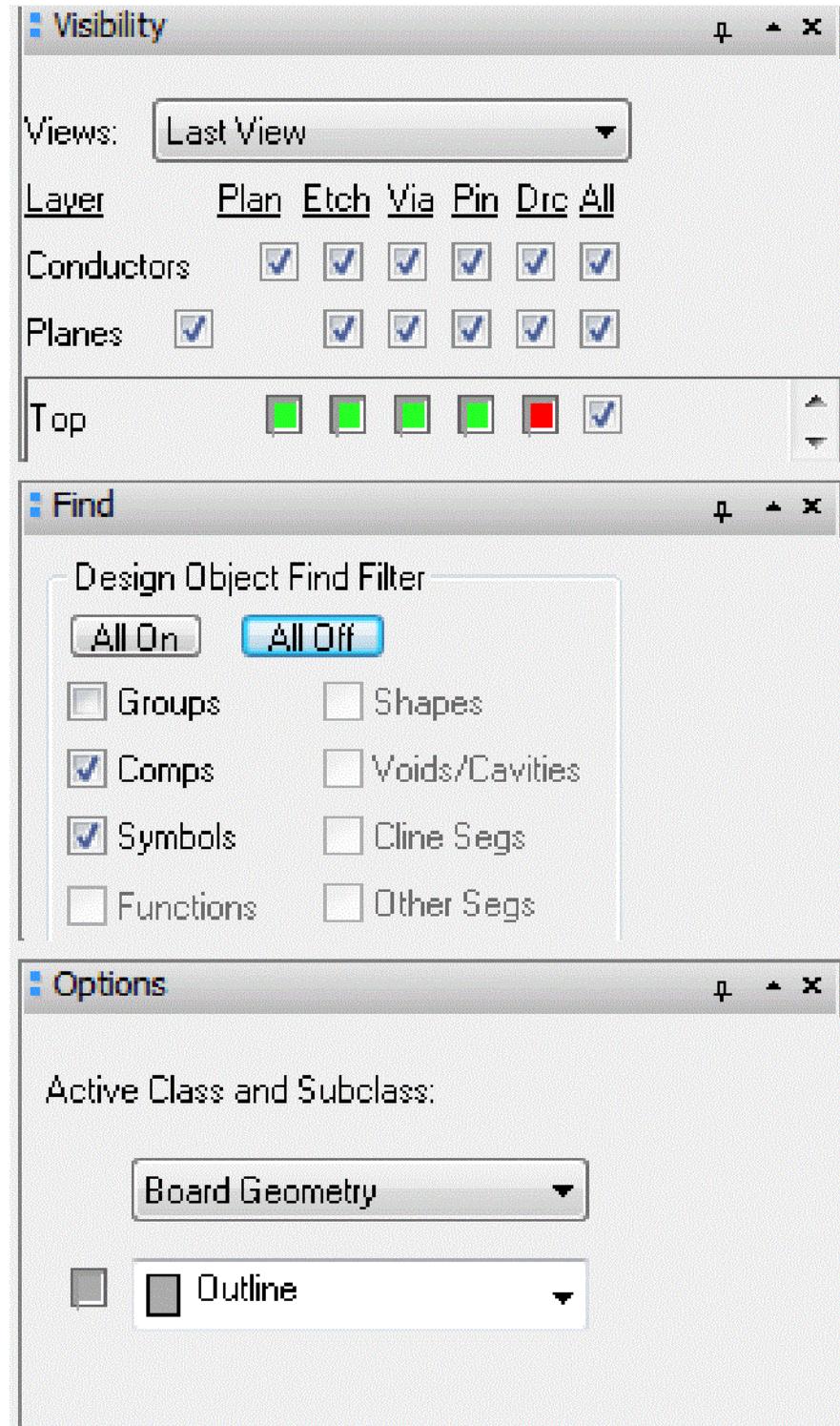
The *Change Icon* button opens a custom icon directory from where you can select a custom icon and assign to any existing or a new command. The path of the custom icon directory is set by the *iconpath* environment variable in User Preference Editor. The *Reset Icon* button restores the default icon applied.



The Control Panel

The Control Panel uses foldable *Options*, *Find*, and *Visibility* window panes that may be quickly resized or relocated to maximize the working design area. Using the pin icon, you can "pin" a window so it remains visible while unpinned windows remain as tabs bordering the design window.

Figure 2.38: Stackable Control Panel Window Pane



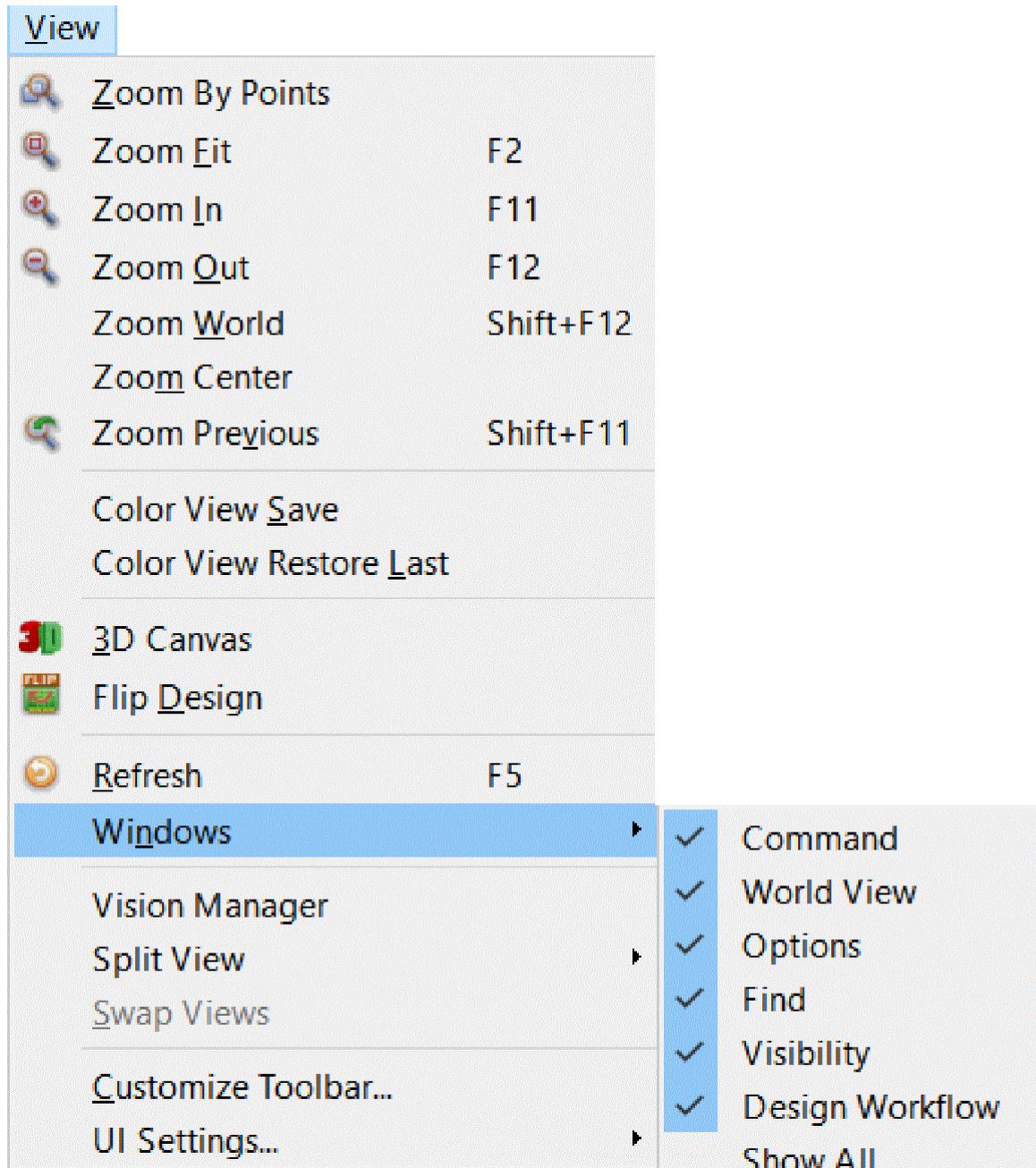
Working with Foldable Windows

The foldable windows are particularly useful on a single monitor setup because they provide more work space, while giving the designer the option of seeing the window-pane information by simply hovering over the tabs bordering the design window. Passing the cursor over any of them quickly unfolds the window pane for viewing or editing, then retracts it.

Click anywhere along the pane name and drag the pane to dock or undock. You can move the panes or windows anywhere within or outside the design window. The available docking area turns light blue. In a dual-monitor system, undocking windows are useful as they can be moved to the second monitor, maximizing the work space.

You control the visibility of these windows by clicking an arrow to expand a docked window pane, clicking the X to hide it, or by using the *View* menu choices to hide or display it.

Figure 2.39: View - Windows commands



The Options Window Pane

The *Options* window pane displays current parameters and values for the active command. Parameters that appear in the *Options* window pane differ according to the active command.

For a command that functions in a pre-select use model, parameters relevant to the command may also be set by right-clicking to display the pop-up menu from which you may choose:

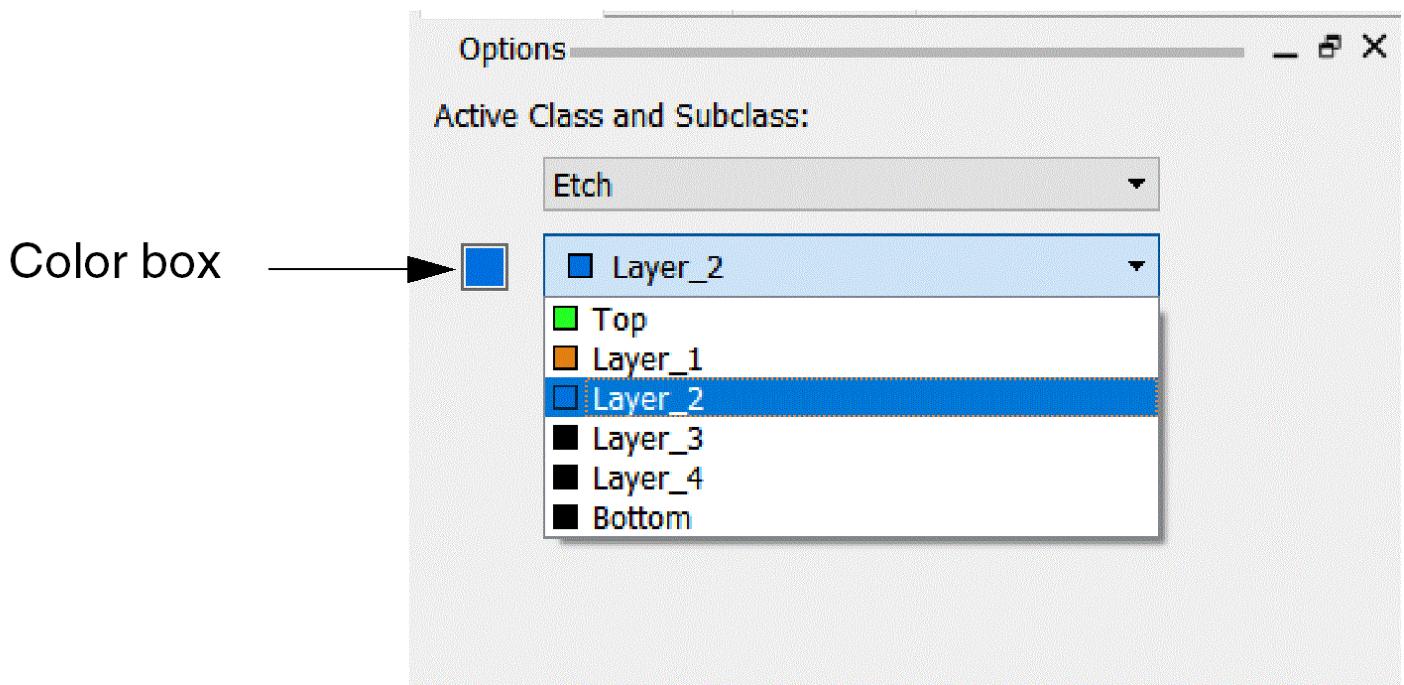
- *Design Parameters* to access the Design Parameter Editor (*Setup – Design Parameters* or `prmed` command)
- *Options*

If a command functioning in a pre-select use model has no parameters that must be set to use the command, *Options* does not appear on the pop-up menu. Changing a parameter using either of these pop-up menu choices automatically updates the *Options* window pane parameters as well.

Dock or undock the window by left-clicking to choose it and moving it anywhere within or outside the design window.

You control the window pane's visibility by clicking an arrow to expand a docked window pane, clicking the X to hide it, or by using *View – Windows – Options* to hide or display it.

Figure 2.40: Options Tab Window Pane (Pinned)



Active Class and Subclass Fields

When you choose a command, the *Options* window pane changes to reflect the appropriate class and the default subclass (the first subclass on the list for that class). For the ETCH/CONDUCTOR class, the subclasses are listed in the order that the layers appear on the design. For non-ETCH/CONDUCTOR classes, the subclasses are sorted alphabetically.

The color swatch to the left of the subclass field indicates the visibility status of the subclass in a design. When in *Visibility* pane, the subclass is enabled the swatch displays the color assigned to the subclass. When the subclass is disabled, the swatch displays the design's background color. You can control the display of a subclass using the *Visibility* pane or the Color Dialog. Choose *Display – Color/Visibility* (`color192` command), described in the *Allegro PCB and Package Physical Layout Command Reference*.

The parameters and values you set in the *Options* window pane take effect immediately and override definitions for the same parameters and values that may exist elsewhere in the tool. For example, the tool looks to the Design Parameter Editor for the rotation and text values. If a different value exists in the *Options* window pane, however, the tool ignores the information in the Design Parameter Editor dialog box.

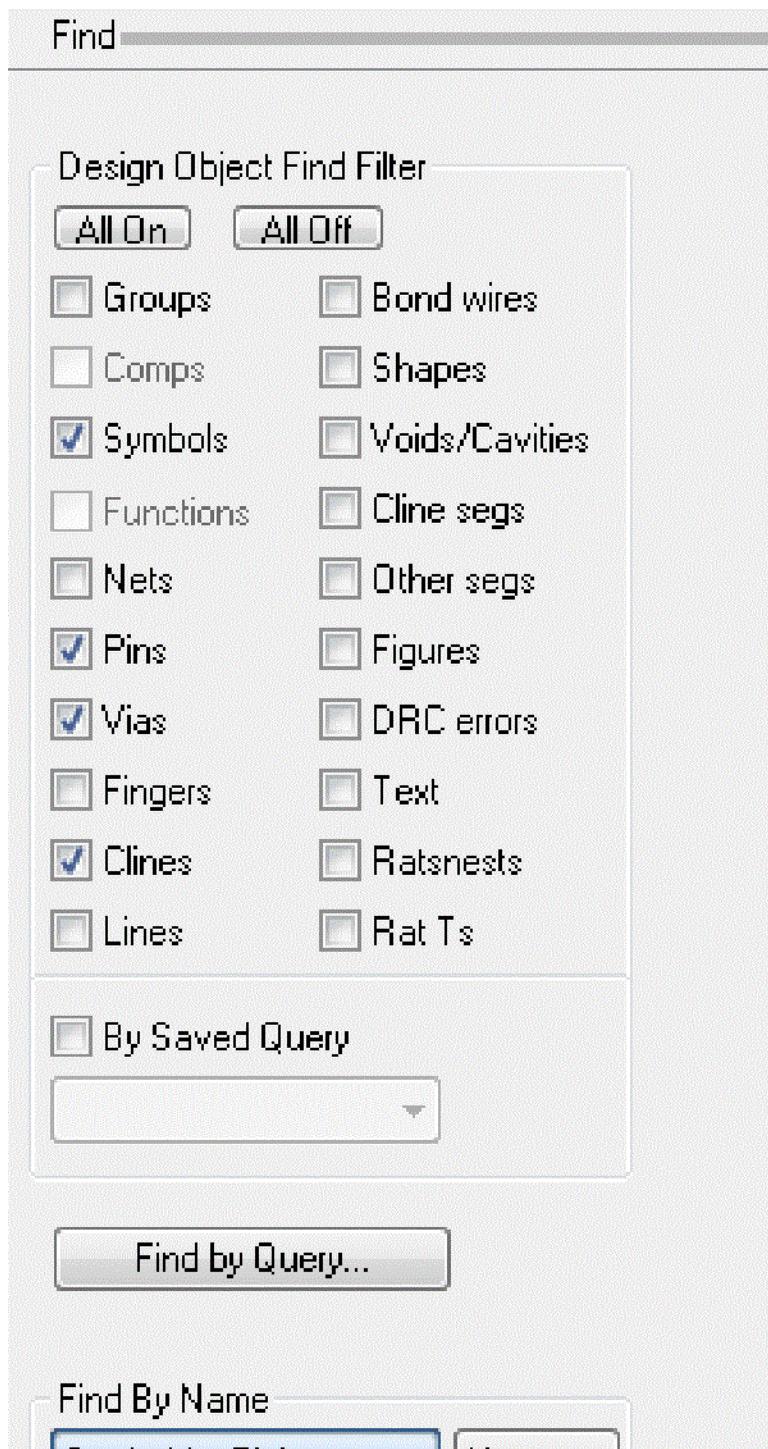
⚠️ When you update values in the Design Parameter Editor, the values in the *Options* window pane change as well.

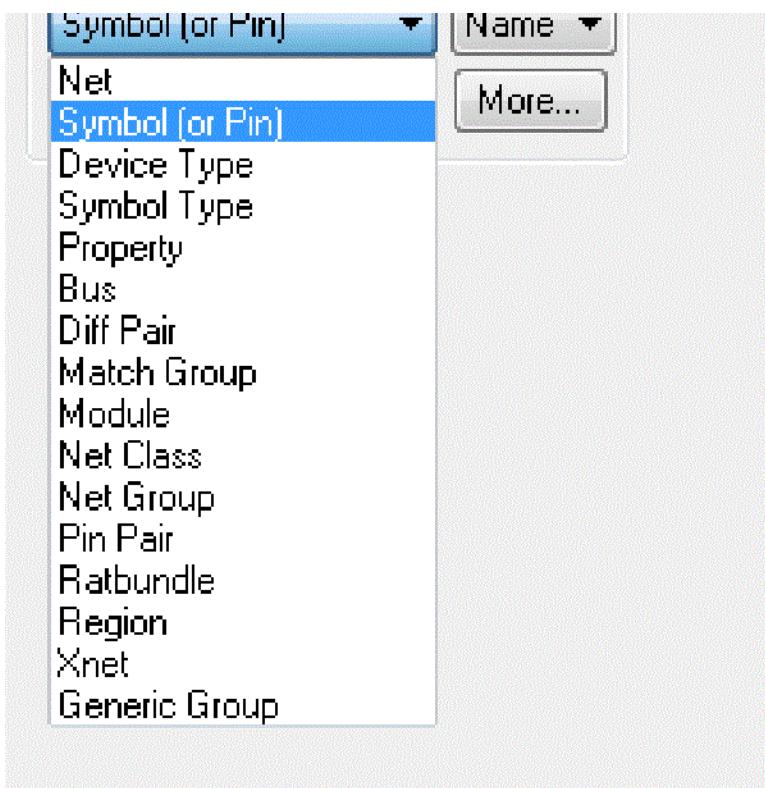
The Find Window Pane

The *Find* window pane lets you specify design elements the active command affects. When you run an interactive command, such as *Edit – Move* ([move](#) command), the *Find* window pane displays the elements the command requires.

To refine your selection set and confine your work to a particular element type, such as all nets, you can also right-click and choose the *Superfilter* temporarily to disable the *Find* window pane.

Figure 2.41: The Find Window Pane (Pinned)





The *Find by Name* section lets you choose elements by name, rather than graphically, or from a text file that contains a list of the names for the design objects.

If you choose *Name* from the drop-down menu and click the *More* button, the *Find by Name* or *Property* dialog box appears displaying a list of all available names for the design object you chose.

If you change from *Name* to *List* and click the *browse* button, a browser window appears that lets you navigate to the directory that contains the specific list file you want.

When using either of these two methods, the layout editor ignores the check boxes in the *Design Object Find Filter* section, unless you use the *Property* pull-down option. For property search, *Find by Name* shows the properties available on displayed objects that are enabled on the *Find* filter.

The Visibility Window Pane

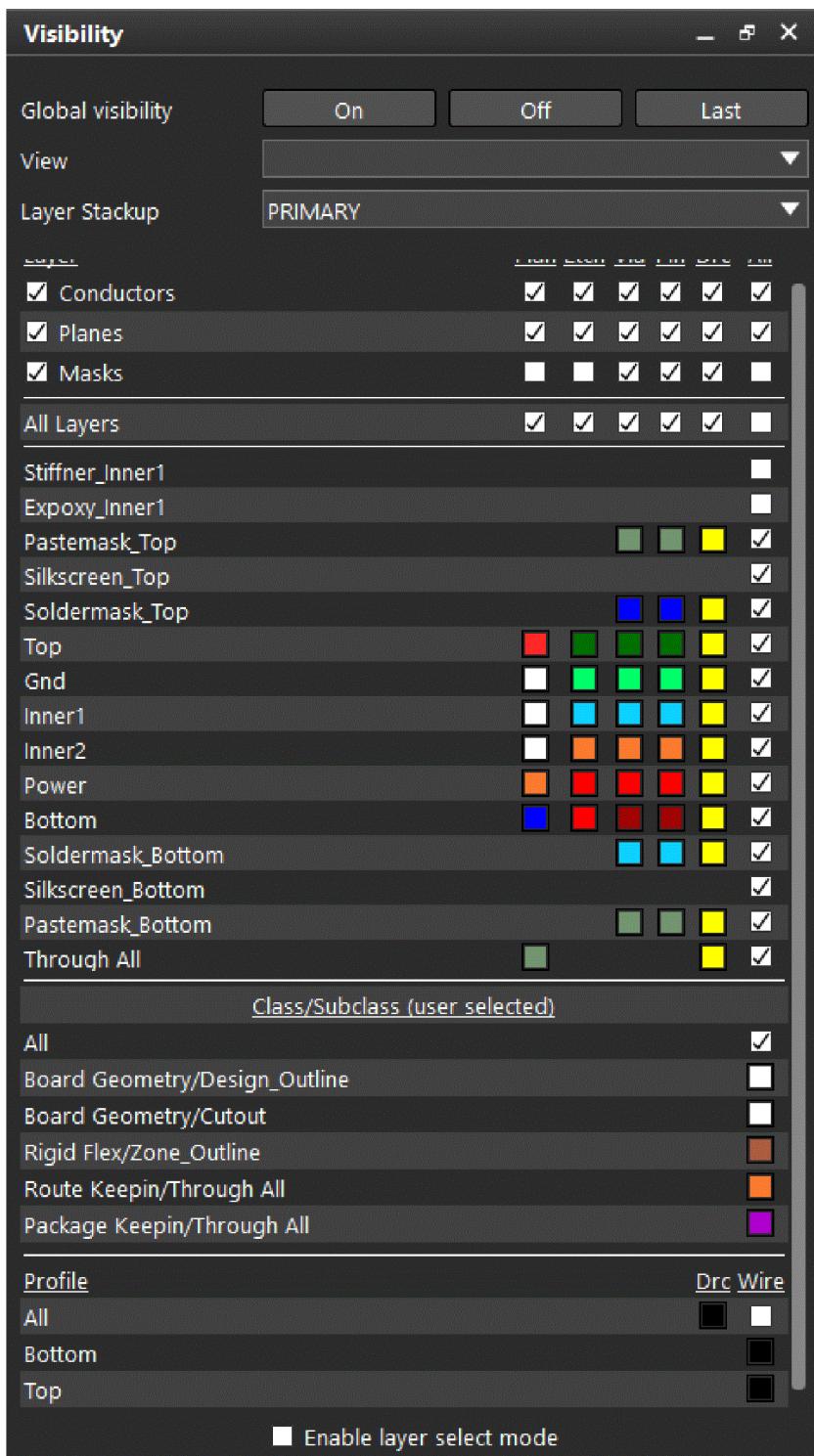
The *Visibility* window pane lets you selectively display or hide conductive elements in a design in a single and multi-stackup designs. The pane lists all design elements, namely, layers, classes/subclasses, profiles, and user-added subclasses present in the design. For example, a PCB layout might list *Conductors*, *Planes*, and *Masks* under *Layer*, whereas a package substrate layout might list *Conductors*, *Dielectrics*, and *Die Stacks*. Same is the case with the other listed items. For easy readability, the rows are colored alternatively.

Once you have assigned colors to each class of design element you can use the *Visibility* window pane to selectively display ETCH/CONDUCTOR, pins, and vias on each layer in the design. The *Visibility* window pane displays the color assigned to a design element when that element is visible, and displays the background color of the design window when the design element is invisible.

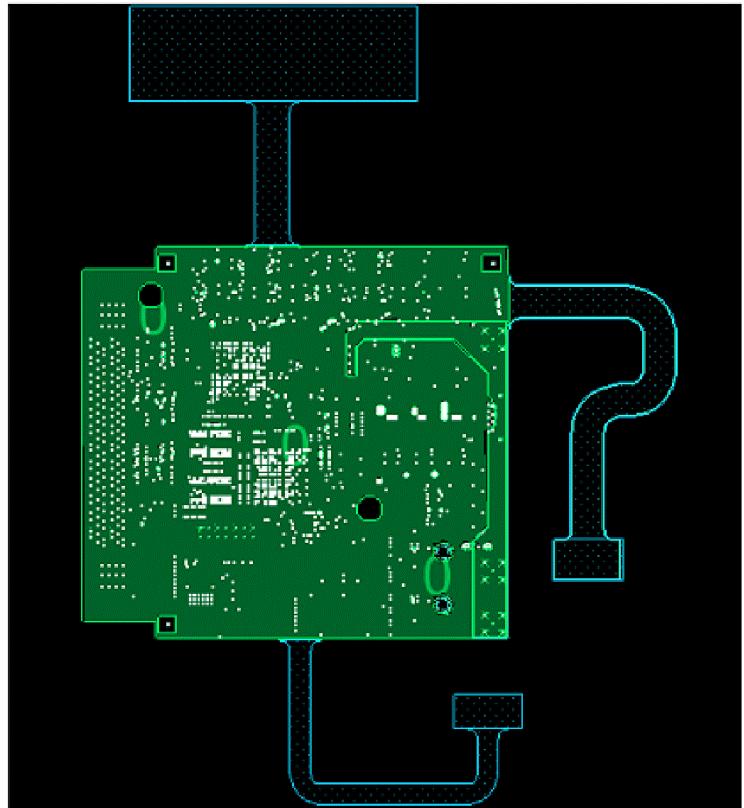
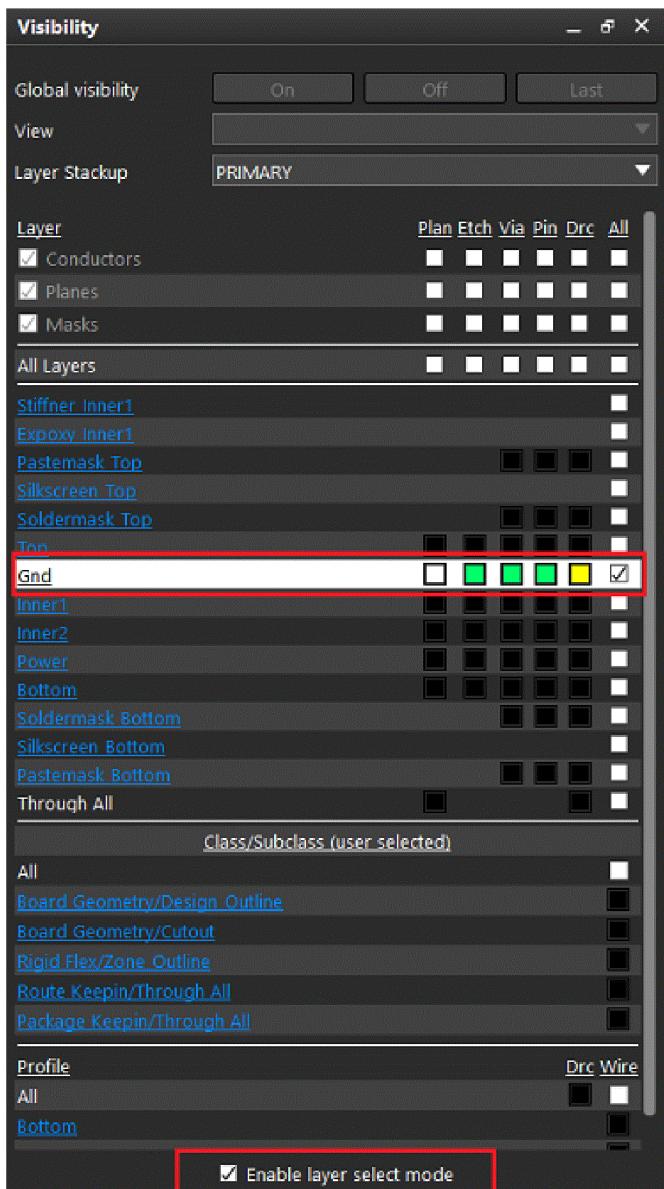
When the button displays the assigned color, visibility is enabled and the design element is visible. When the button displays the background color, visibility is disabled and the design element is hidden. You can quickly control the visibility of all layers by clicking the *All* button associated with the desired design element.

To delete plane layers in the *Visibility* window pane, click the *Planes* checkbox, a convenience if a design has a large number of layers that you might have to scroll through. You can set global visibility from the *Visibility* window pane. You can also turn on or off mask layers.

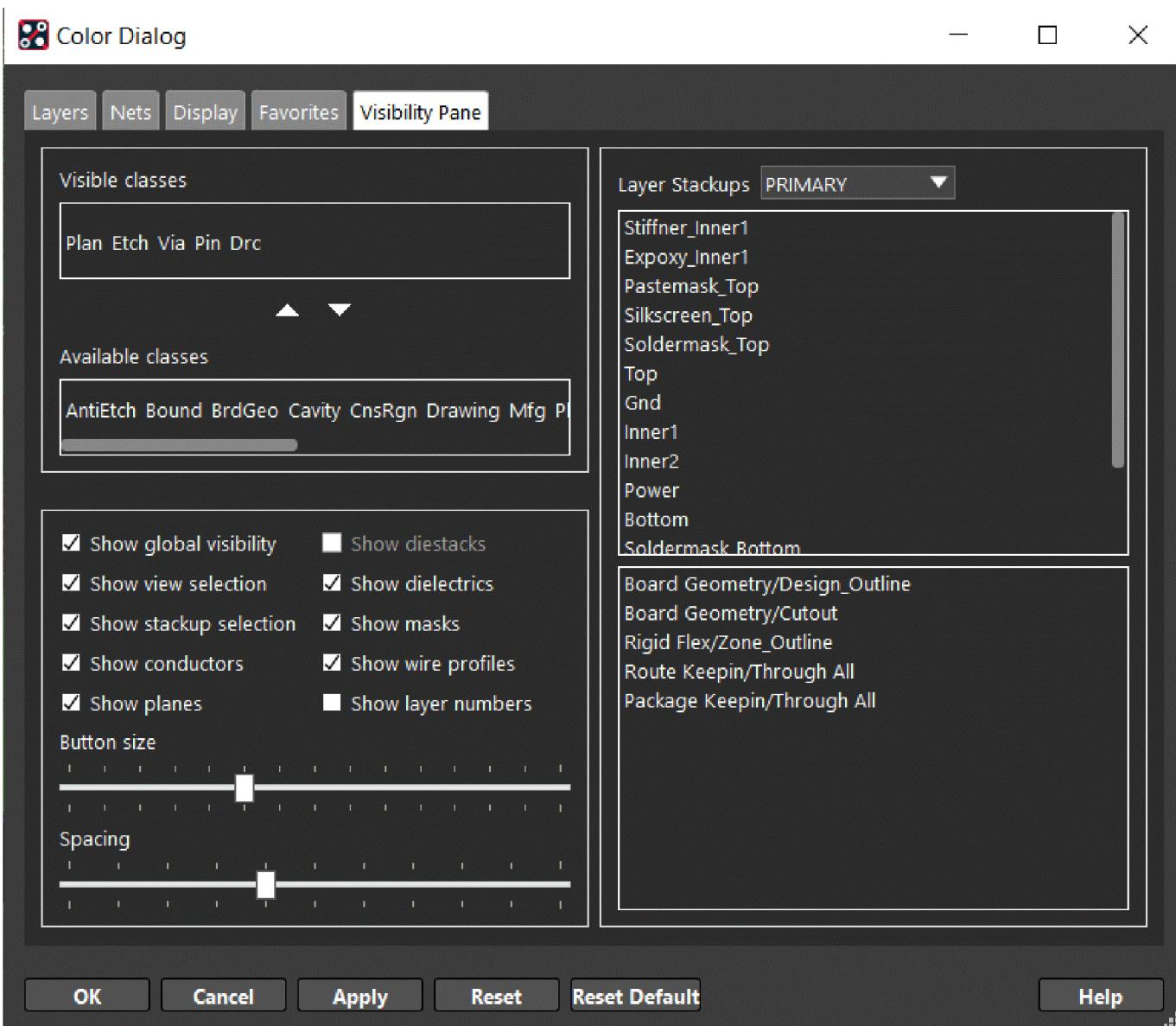
Figure 2.42: The Visibility Window Pane



The *Enable layer select mode* puts the canvas into a single layer view and lets you quickly view or scroll through each available layer. When this mode is enabled, the individual design elements (layers, classes/subclasses, and profiles) changes into active links in blue color that you can click to select. These links allow you to switch between layers. When in the layer select mode, any combination of multiple layers can be selected and viewed using **ctrl** key.



You can customize the Visibility window pane from the Color Visibility dialog box ([The Color Dialog Box](#)). You can enable and disable global visibility, stackup selection, layers, planes, and so on.



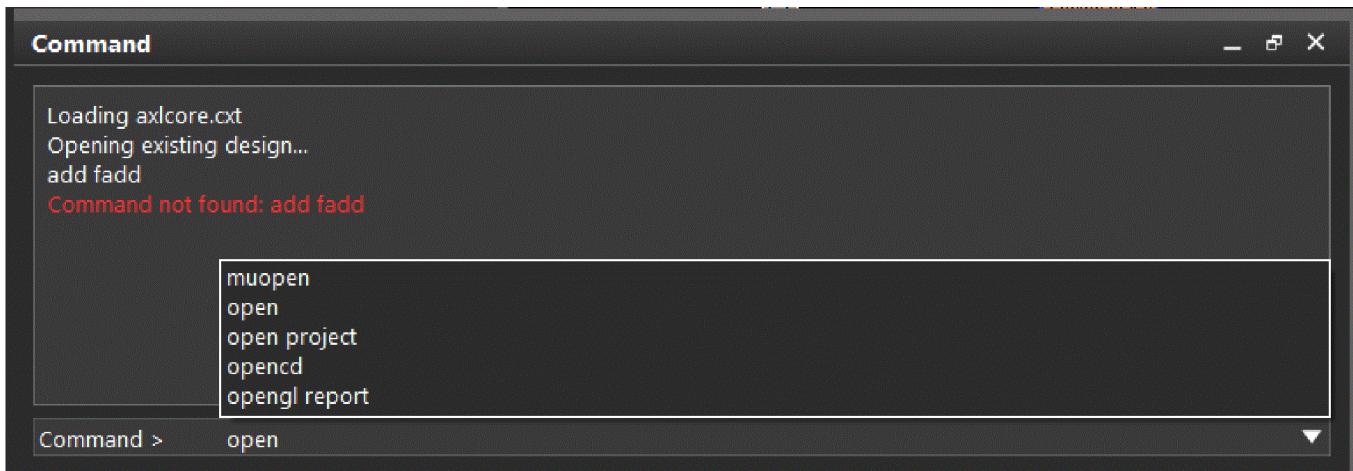
Layers types that do not exist in a design are automatically hidden in the *Visibility* plane even though they are enabled in the Color Dialog.

The Command Window

The Allegro GUI includes a command window that allows you to enter commands while also displaying messages and command output. The command window has two separate sections for input and output. You can increase or decrease the font size in these windows using a mouse with a wheel. Hold the **Ctrl** key on the keyboard and scroll the wheel up to increase the text size or scroll down to decrease the text size. The input section of the command window placed in bottom provides auto-completion of commands. The command line is editable similar to a text editor. The output section shows full history of recently-used commands and the messages. These messages follow a color scheme to easily detect their type:

- Black: Messages stating information are displayed in black color.
- Orange: Messages showing warnings are displayed in orange color.
- Red: Messages showing errors are displayed in red color.

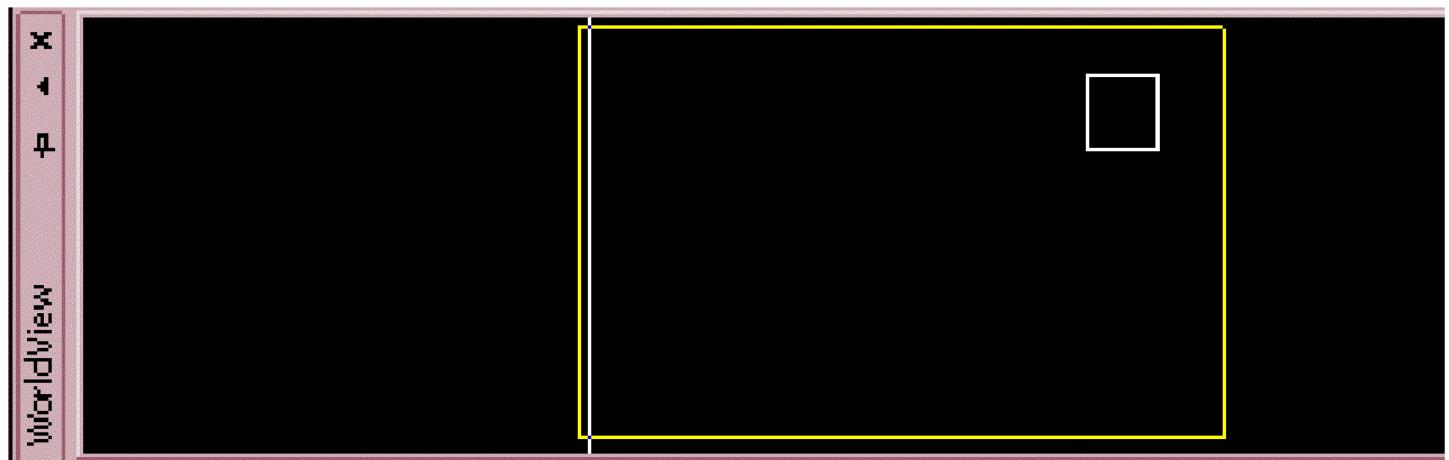
Figure 2.43: The Command Window



The WorldView Window

The *WorldView* window provides a bird's-eye view of your design. Using the *WorldView* window, you can zoom in to display a smaller area of the design outline or zoom out to display a larger area of the design. You can use the *WorldView* window alone with the *View* menu commands and accelerator keys.

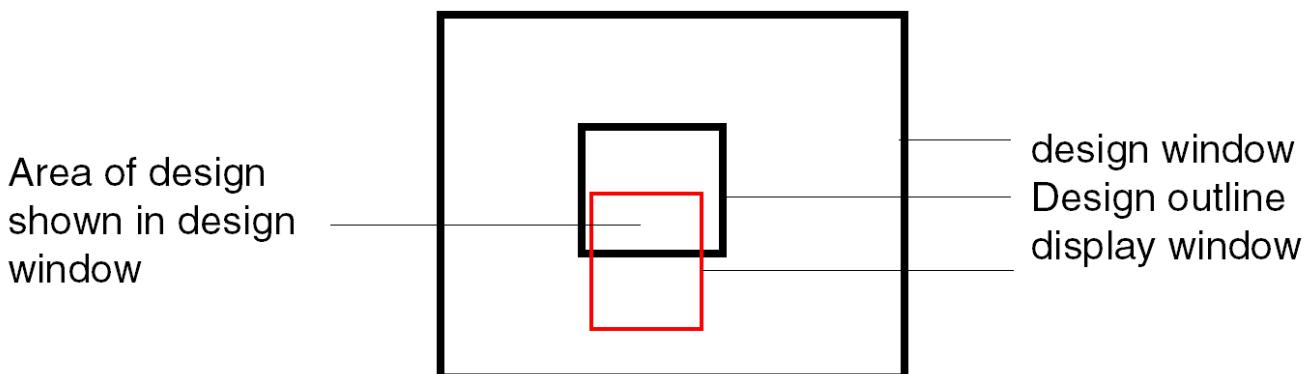
Figure 2.44: WorldView Window Pane



Using the WorldView Window Pane

There are three ways you can control the view of the design using the *WorldView* window:

- To display specific areas of the design
- To scroll through the design
- To zoom in or out of the design



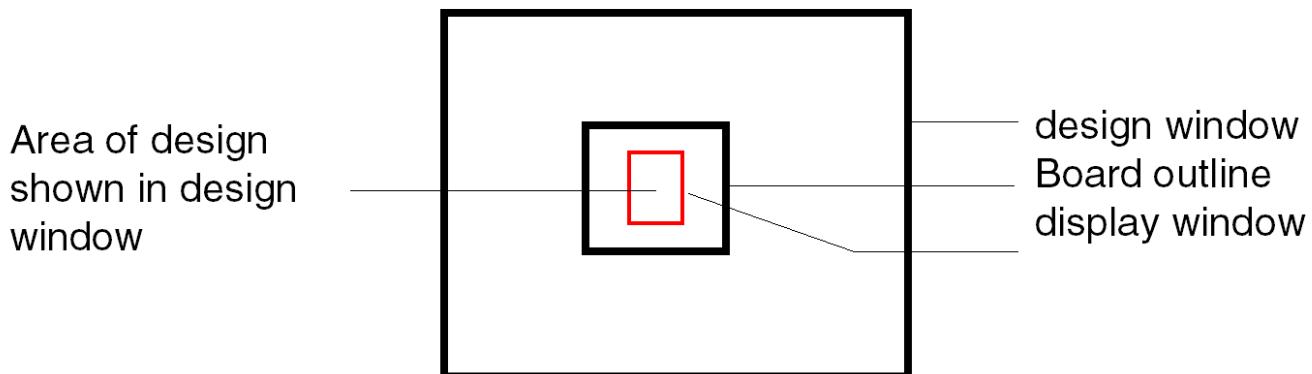
Displaying Specific Areas of a Design

To use the *WorldView* window to display specific areas of a design:

- In the *WorldView* window, left-click and drag-and-drop the display window over the area of the design that you want to display in the design window.

⚠ If you are using a three-button UNIX mouse, the middle button gives you a greater degree of control when performing this operation.

If you size the display window over a small area of the outline (using the left button), the design window zooms in on that area.

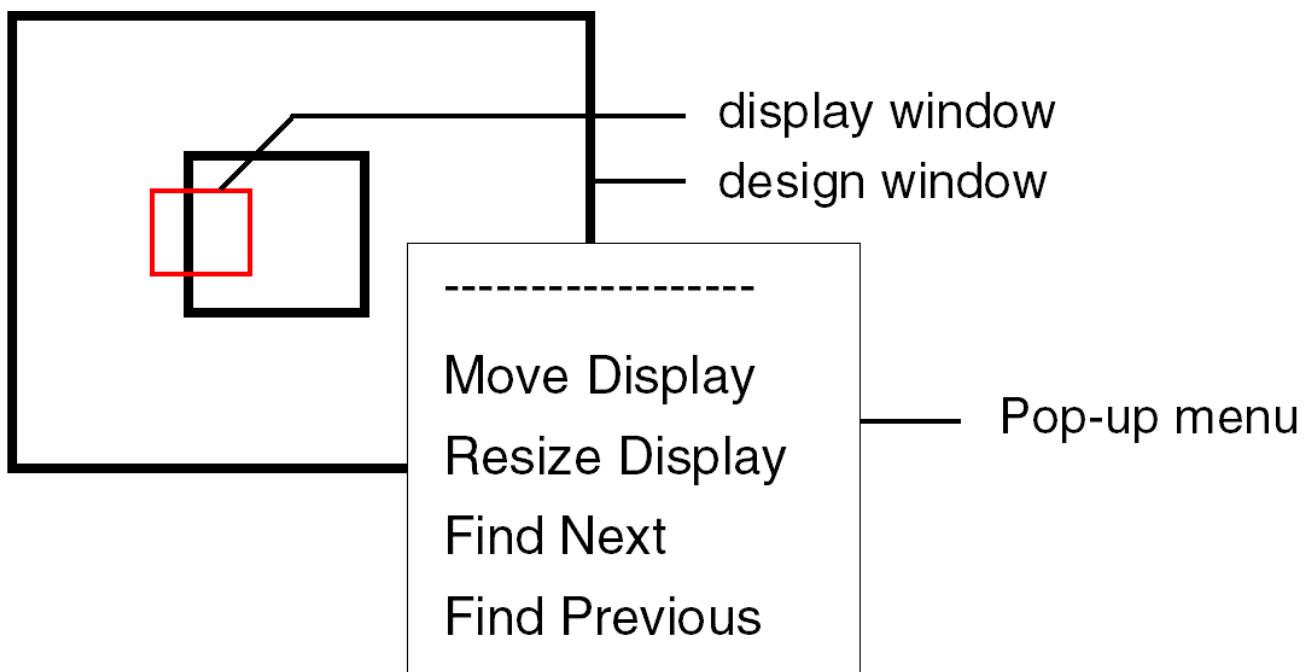


If you size the display window over a larger area of the outline (using the left button), the design window zooms out to display that area.

The WorldView Window Pop-Up Menu

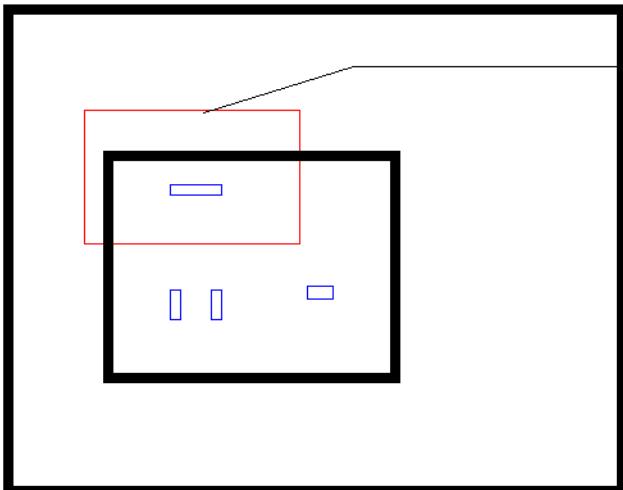
To display the *WorldView* window pop-up menu

- Right-click in the *WorldView* window.
The pop-up menu appears.



Following are descriptions of the options in the *WorldView* pop-up menu.

<i>Move Display</i>	lets you move the display window to select an area of a drawing for display in the design window.
<i>Resize Display</i>	zooms the design window on an area you define by selecting points in the <i>WorldView</i> window. You can also type <code>window center</code> at the console window to perform the same function, but you then specify the new window area by selecting its center in the design window.
<i>Find Next</i>	advances through the list of any highlighted items, centering the display on each of them, in the order in which they were highlighted.
<i>Find Previous</i>	reverses through the list of highlighted items.



display window
centered over
highlighted item

You can continue choosing *Find Next* or *Find Previous* by left-clicking in the *WorldView* window. The click repeats the last command. *Find Next* is the default command in effect with a left click after new elements have been highlighted.

The command window identifies each element as you cycle through the highlighted items in the *WorldView* window. The > symbol indicates that you are advancing to the next element in the list whereas the < symbol indicates that you are advancing to the previous element. For example, after centering on a line with *Find Next*, the message is > *Line*.

The Status Bar

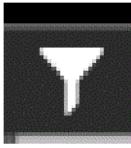
The Status bar shows the active command, subclass, application mode and number of selected objects. These coordinates change as you move the mouse over the canvas. You can also customize the status bar by right-clicking and selecting the options to show or hide the different panes.

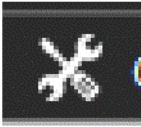
Status Bar Panes

Figure 2.45: Status Bar



The Status bar has the following elements:

<i>App status</i>		 Displays the current state of the application.  process.  Halts execution of the currently active process.	Ready to activate any command or processing action or command that cannot be cancelled.
<i>Active command</i>		Displays name of the current active command.	
<i>Compatibility Mode</i>		Displays the compatibility mode status, if set.	
<i>Flipboard Mode</i>		Displays the design flipped along Y axis.	
<i>Active Class/Subclass</i>		Indicates active class and subclass. Click this field to display a pop-up menu that lets you choose class and subclass. Switching a class/subclass from the status bar sets the <i>Active class and subclass</i> fields with the same values in the Options tab and makes the layers visible in the design canvas.	
<i>Mouse XY Coordinate</i>		Displays coordinates of the current location of the cursor	
<i>Units</i>		Displays design units. To change unit, click to open design tab of the Design Parameter Editor.	
<i>Pick command</i>		Lets you display a dialog box. When you click this button, and you are in an interactive command, for example, <i>add connect</i> , the Pick dialog box appears and remains displayed until you dismiss it. If the <i>Cmd</i> status is <i>Idle</i> , and you click the <i>P</i> button, the Zoom Center dialog box appears and remains displayed until you dismiss it. You can enter specific or incremental values in these dialog boxes. For additional information, see the <i>Pick</i> dialog box.	
<i>Toggle XY: Absolute/Relative</i>		Toggles the x, y read-out from absolute mode to relative mode. When you are in absolute mode, the x y coordinates location is from the origin of the design. When you are in relative mode, the origin is always from the last pick and the button is labeled <i>R</i> . The layout editor always starts designs in absolute mode.	
<i>Aux/Script Text</i>		Displays the name of the script playing.	
<i>Super Filter</i>		Lets you choose a particular element type to refine your selection set and temporarily disable all other elements from the right-mouse button pop-up menu rather than the <i>Find</i> window pane.	

App Mode		Displays the currently set application mode.
DRC Status		Empty display string indicates design rule checking is disabled.
		If the display string displays DRC it indicates that design rule checking is enabled. Right-click to access the following options: <ul style="list-style-type: none">• Enable On-Line DRC• Enable On-Line DFM checks• Update• Display Status A red color box indicates DRC is out of date or Batch DRC is required. A yellow color box indicates DRC is up to date, but DRC errors exist. A green color box indicates DRC is up to date and no DRC errors exist.
Selected Objects		Indicates the number of selected objects on the canvas. Click this field to display a pop-up menu with commands that operate on the currently selected element(s). <div style="border: 1px solid #ccc; padding: 5px; background-color: #fff;"><p>⚠ You can resize the status bar fields by setting the environment variable <code>resizable_status_bar</code> from the <i>General</i> category in the <i>Ui</i> section of the <i>User-Preference</i> dialog box.</p></div>
Pulse Status		Indicates the status of layout editor connection with Pulse platform. Clicking the icon opens Allegro Pulse Status dialog. Three statuses are possible: <ul style="list-style-type: none">• Connection: <i>Disconnected</i>, <i>Connecting</i>, and <i>Connected</i>.• Health: <i>Maintenance mode</i>, <i>Normal</i>, and <i>Error</i>.• Pending Operations: <i>0</i>, or <i>N-many</i> <div style="border: 1px solid #ccc; padding: 5px; background-color: #fff;"><p>⚠ This icon is available only if Pulse is enabled.</p></div>

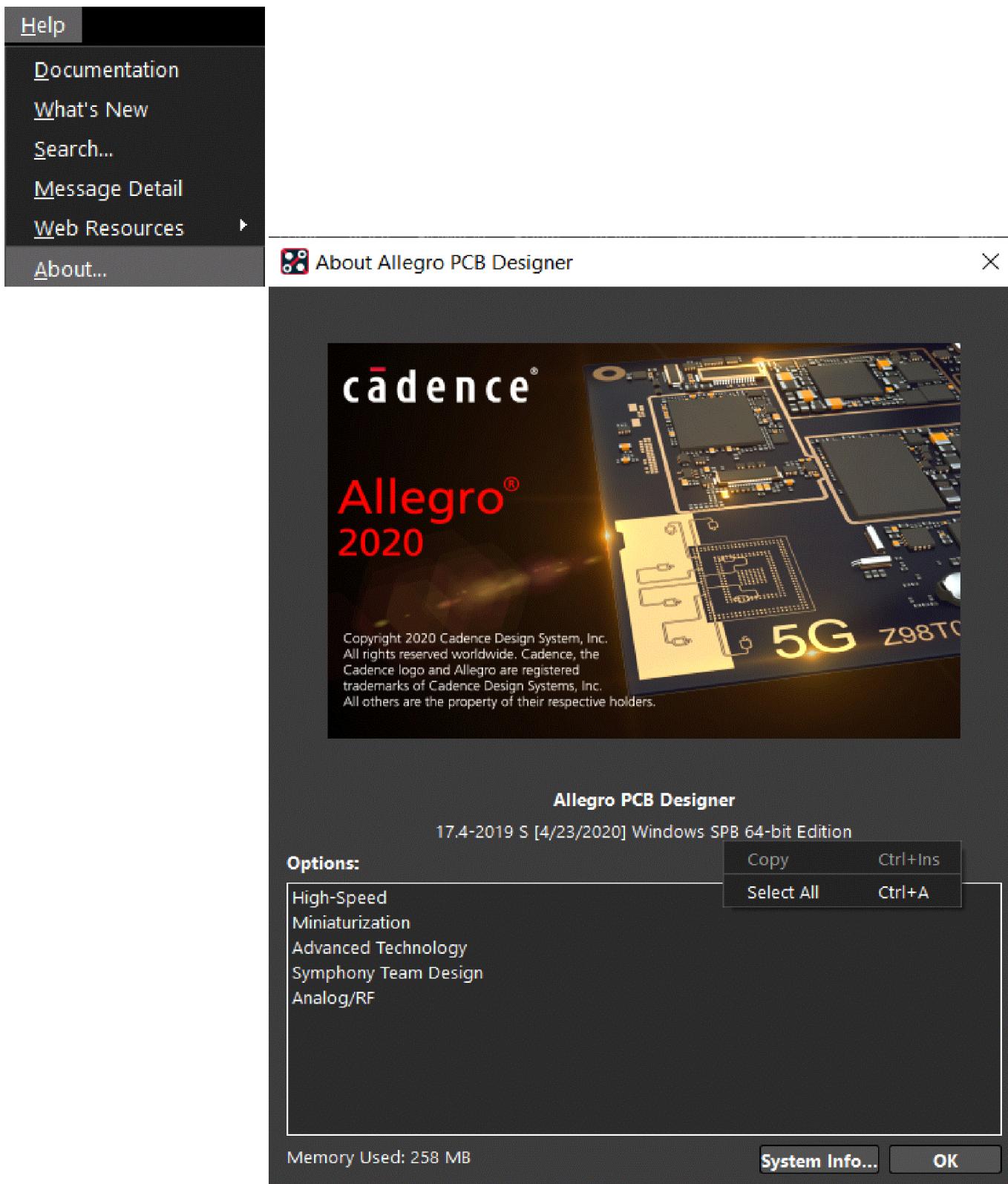
Customizing Status Bar

You can customize the status bar to show or hide the elements or panes. Right-click the status bar and change selection to show or hide the panes.

- ✓ App Status
- ✓ Active Command
- ✓ Compatibility Mode
- ✓ Extra 1
- ✓ Flipboard Mode
- ✓ Active Class/Subclass
- ✓ Mouse XY Coord
- ✓ Units
- ✓ Pick Command
- ✓ Toggle XY: Absolute/Relative
- ✓ Aux/Script Text
- ✓ Super Filter
- ✓ App Mode
- ✓ DRC Status
- ✓ Selected Objects

The About Window

To find out which version of layout editor you have, click *Help – About*. The following illustration displays the *About* dialog box that opens.



Following information can be viewed:

- Product name

- Complete version
- Product options
- Memory used

You can select the text within the *About* window for sharing information about layout editor. Pop-up commands *Select All* and *Copy* are also available to copy the necessary detail.

To view system-level information, click *System Info* button.

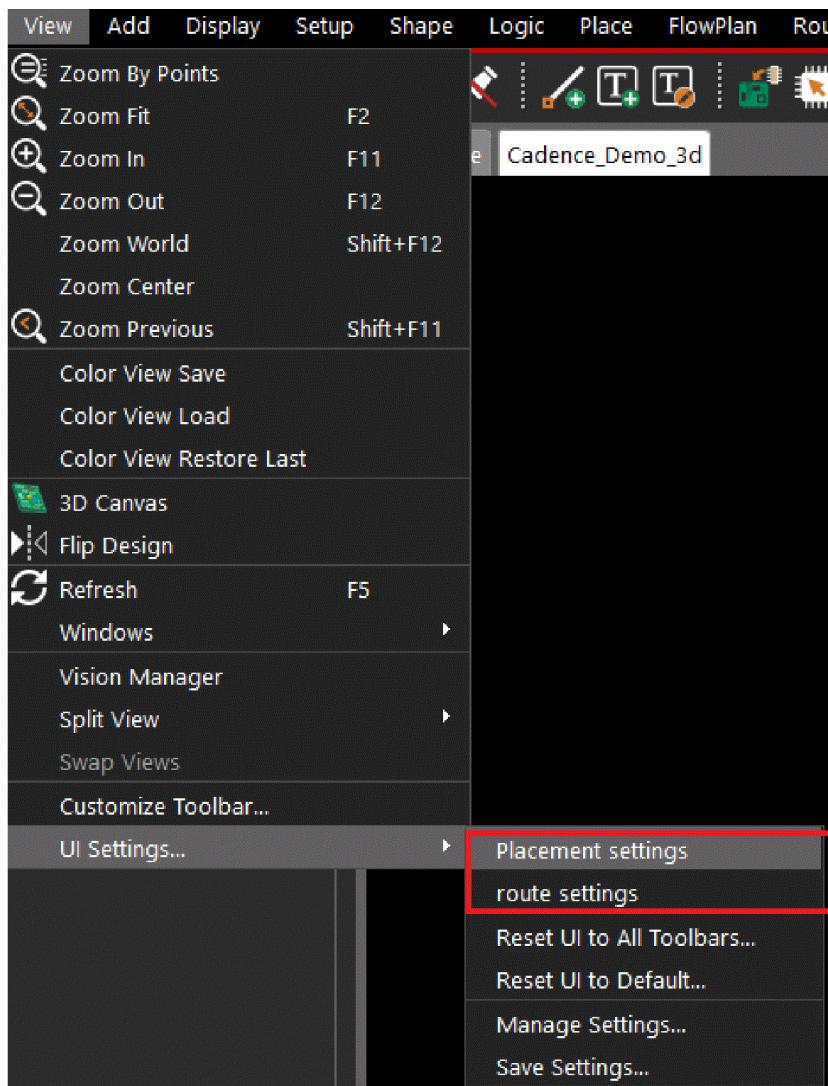
Customizing Design Canvas

Layout editors, by default, shows a minimalistic set of toolbars, icons and pre-defined size and positions of docking windows. The arrangement of toolbars and icons, placement and size of docking panes can be further customized and saved per design requirements. Multiple views can be created, saved and recalled when required. You can also export and import customized settings across systems. These settings can be saved at site level.

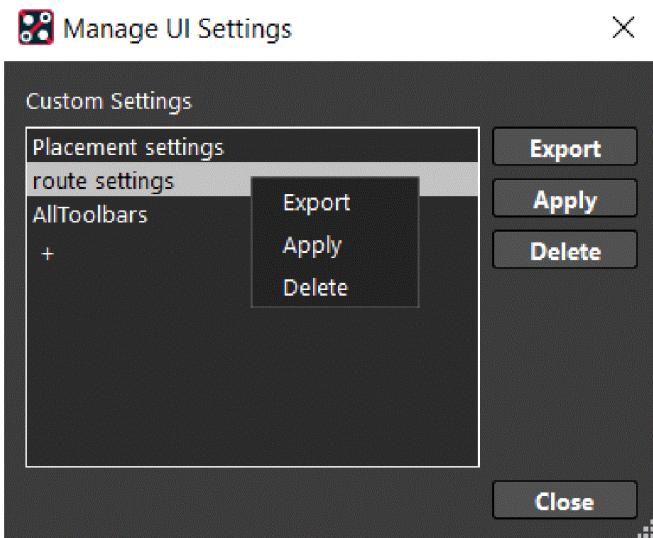
To save a user-defined setting, choose *View – UI Settings – Save Settings*, enter a name and click *Save*.



The setting adds to *UI Settings* menu. The user-defined settings becomes available in the menu and can be selected and applied to the active database.



To export, import, or delete any setting choose *View – UI Settings – Manage Settings*.



A custom setting on export is saved as a configuration (.ini) file in the <HOME>/PCBENV directory. To import an already saved custom setting, click + icon and browse to the location of configuration file (.ini). The selected custom setting becomes available to apply.

To restore the legacy (pre 17.4 release) all toolbars icons view, you can choose *Reset UI to All toolbars* menu. This command brings back default toolbar settings available in the installation directory. It also checks for any site-level UI settings saved at the path specified for the MENUPATH environment variable.

You can also access default configuration settings from the Manage UI Settings. In this window, click + icon and browse to the location of `AllToolbars.ini` file. This file is located at the <installation_directory>/share pcb/text. A new entry `AllToolbars` gets added to the list of *Custom settings*. Select the setting and click the *Apply* button.

The customized configuration can be saved and used as default. Set the operating-system variable CDS_SITE, which overrides the default site location:

```
<cdsroot>/share/local
```

See [Site Customization](#), which describes the options available with CDS_SITE.

You can also use *Reset UI to Default* menu to go back to the conventional window and toolbar positions and display settings.

Padstack Designer

The Padstack Designer lets you create or edit library padstacks, including:

- Defining the parameters of padstacks
- Creating blind and buried via padstacks
- Adding padstack layers
- Copying padstack layers
- Deleting layers in a padstack

A library padstack defines pad data for all layers. You must define padstacks before you create any package symbols, because each pin in a package symbol must have an associated padstack.

When you double-click the Pad Designer icon (in Windows) or type `pad_designer` at the UNIX system prompt, the Padstack Designer appears.

For information on the Padstack Editor, see "[Using the Padstack Designer](#)" in "[Library Padstacks](#)".

Maintaining Databases

The DBDoctor program checks the database for errors and other problems and reports them as they occur. DBDoctor supports .brd, .mcm, .mdd, .psm, .dra, .pad, .sav, and .scf databases. DBDoctor can:

- Analyze and fix database problems.

- Eliminate duplicate vias.
- Perform batch design rule checking (DRC).
- Uprev databases more than one revision old.

Running DBDoctor

To verify the integrity of a drawing database at any time during the design cycle, run DBDoctor at regular intervals but always after completing a design and prior to creating an artwork file. For specific procedures, see *Tools – Database Update* (`dbdoctor` command) in the Allegro PCB and Package Physical Layout Command Reference.

You can run DBDoctor to verify work in progress, or from a terminal window outside the layout editor, perhaps to check multiple input designs in batch mode by using wildcards and various switches. You do not have to run the layout editor to use DBDoctor.

During processing, DBDoctor generates `dbdoctor.log`, which records check summaries and detailed information on records that contain errors, as well as names of symbols and nets and x.y coordinate information. If DBDoctor finds an error, then it adds the `dbdoctor.log` to the design as an attachment. The layout editor only saves the log file from the last run of DBDoctor that found an error.

DBDoctor uses the input file name by default and copies it as `<boardname>.brd.orig` or `<>.mcm.orig` in the same directory, thereby permitting you use wild cards. If you use wildcards with the input file, then each design you enter is copied under `<boardname>.brd.orig` or `<>.mcm.orig` unless you choose the *No Backup* field on the dialog box that appears when you launch DBDoctor externally or use the `-no_backup` switch, in which case, the tool replaces the original design.

Partial Versus Full Database Consistency Checks on Saving

When you save a design, the tool executes a partial database consistency check by default, in essence, a quick check.

The `dbsave_full_check` environment variable indicates to the database save utility when to do a full check rather than a quick check. A number of 1 or 0 specifies that each time a design is saved, execute a full check. If you set the variable to 100, then every 100 checks a full check occurs.

For example, to set the `dbsave_full_check` environment variable to do a full check every five saves, at the console window prompt, type:

```
set dbsave_full_check = 5
```

If the tool detects errors, it saves the file as `<name>.SAV`.

 A full database check may considerably lengthen the time required to save large databases.

 For information on opening a design saved in a previous version of the layout editor in the current version, see [Uprevvng](#).

Database Revision Support

The oldest database revision support for uprev on a platform depends on when the layout editor initially supported that platform. The following table lists the older database that you can uprev.

Platform	Allegro Physical Database
Linux	14.0
Windows	11.0

Databases older than 11.0 require you to maintain 16.6 on a Sparc Workstation to update the design to revision 16.6 before accessing in on Windows or Linux.

Uprevvng Differential Pairs from Release 14.x to Release 15.x

When uprevvng a Release 14.x database to Release 15.x, the layout editor shifts the differential pair primary gap from the spacing rule set to the physical rule set assigned to the differential pair.

Since you can associate a physical rule set with nets tied to different spacing rule sets, the tool takes the value of the new 15.x gap from the first instance of the differential pair information found.

Differential Pair Log

When you uprev a design containing differential pairs, any problems with migrating the differential pairs appear in the `uprev_diffpair.log`, which you can scan using *File – Viewlog (viewlog)* command, described in the *Allegro PCB and Package Physical Layout Command Reference*. The tool only creates the log if problems occur.

The `uprev_diffpair.log` file lists the discrepancies for all other nets that share the physical rule but had different spacings for the differential pair.

These warnings are a guide that you can use in recreating differential pair constraints through ECsets or new physical rule sets in Release 15.x. You can set the gap values to the original values once data is moved into Release 15.x.

Information appears in the `uprev_diffpair.log` in this format:

```
Warning: Already seeded gaps for the physical rule PAIRS found.
```

```
DP1 requires new physical rules.
```

```
Original Primary gap on TOP was 8.0.
```

```
Original Primary gap on BOTTOM was 10.0.
```

Additional information that cannot translate to Release 15.x rules occurs when Release 14.x databases contain differential pair data specific to spacing rule sets.

Since constraint areas no longer apply to differential pairs, you should carefully review the differential pairs in Release 15.x. Updating the DRC, in this case, shows problem areas within constraint areas. You can then apply the smallest gap spacing found in constraint areas for differential pairs to the new physical constraint value for *DiffPair neck gap* in the appropriate constraint set for the differential pairs.

Also, data uprevved to Release 15.x has spacing rule sets that you may not need. You can delete them if they only apply to differential pairs.

 Cadence recommends recreating differential pair constraints at the differential pair object level rather than on individual nets.

For additional information, see the *Creating Design Rules* user guide in your documentation set.

Removing the DIFFERENTIAL_PAIR Property

During the uprev process, the layout editor removes the DIFFERENTIAL_PAIR property (obsolete in 15.x releases) from the nets in the pair and places the nets in a differential pair group object. The object group name is the same as the property value. Differential pairs appear in the Assign Differential Pair dialog box, available by choosing *Logic – Assign Differential Pair (diff pairs)* command.

If more than two nets have the same DIFFERENTIAL_PAIR property value, the tool randomly uses two of the nets to create the differential pair group. It skips the remaining nets, and a warning appears in the `uprev_diffpair.log`.

Converting Spacing Constraints

The differential pair spacing constraints, which are now electrical constraints, convert as shown in the following table:

14.x Diffair Spacing Constraint	Converted to this 15.x Electrical Property	Notes
<i>Length Tolerance</i>	DIFFP_PHASE_TOL	The DIFFP_PHASE_TOL property replaces the old DIFFP_LENGTH_TOL property. Delay percentage is no longer supported.
<i>Primary Max Sep</i>	DIFFP_PRIMARY_GAP	
<i>Secondary Max Sep</i>	—	This constraint is obsolete.
<i>Secondary Length</i>	DIFFP_UNCOUPLED_LENGTH	The DIFFP_UNCOUPLED_LENGTH property replaces the old DIFFP_2ND_LENGTH property.

In the case where a property on the net overrides an old spacing constraint, the most conservative value (the lowest value) converts to the new electrical property.

For those instances when nets have different values assigned to their differential pair constraints, including any assignments for constraint areas in the Spacing Rule Set Assignment Table, the most conservative value converts to the new electrical property for both nets. This is true, even when the value is zero.

⚠ The layout editor flags any converted properties that result in a value of zero for the 15.x property in the `uprev_diffpair.log` file.

Differential pair properties placed on nets automatically bubble up to the differential pair group. The 14.x spacing constraint set name is kept on the nets, along with any non-differential pair constraints. The tool does not create a new electrical constraint set containing the new electrical constraints for the nets. Consequently, during uprev the same properties connect to each net in the pair, through the differential pair group.

Converting a DRC Mode

The DRC modes for the 14.x *Length Tolerance* and *Secondary Length (Max Len over Prim Sep)* spacing constraints on the nets convert into one 15.x DRC mode for the pair called *All Differential pair checks*.

If the 14.x modes differ, the layout editor assigns the mode based on this order of precedence: *Always/On, Batch, Never/Off*.

The tool converts the modes as follows:

14.x DRC Modes	Converted to this 15.x DRC Mode
Any mode is set to <i>Always</i>	<i>On</i>
A variety of <i>Batch</i> and <i>Never</i> settings	<i>Batch</i>

Converting Environment Variables

The `drc_diff_pair_overide` and `drc_diff_pair_primary_separation_tolerance` environment variables are retained in Release 15.x only for uprevving purposes. You can no longer set these variables. During migration, they convert to new `DIFFP_COUPLED_PLUS` and `DIFFP_COUPLED_MINUS` electrical properties that define the coupling tolerances around the primary gap for the differential pair. For details about these properties, see [DIFFP_COUPLED_PLUS](#) and [DIFFP_COUPLED_MINUS](#) in the *Allegro Platform Properties Reference*.

The layout editor converts the `drc_diff_pair_overide` environment variable as follows:

14.x <code>drc_diff_pair_overide</code> Value	Converted to these 15.x Properties*
0 or blank	Nothing done
100	<code>DIFFP_COUPLED_PLUS = 1</code> <code>DIFFP_COUPLED_MINUS = 1</code>
200	<code>DIFFP_COUPLED_PLUS = 2</code> <code>DIFFP_COUPLED_MINUS = 2</code>

*These values are in database units using the specified accuracy (both settings are in the *Design* tab of the *Design Parameter Editor*). Use *Setup – Design Parameters* (`prmed` command) to access the Design Parameter Editor. For example, for a `drc_diff_pair_overide` value of 100, if the *User Units* are `mils` and the *Accuracy* is 2, these become the 15.x property values:

`DIFFP_COUPLED_PLUS = 0.01 MIL`

`DIFFP_COUPLED_MINUS = 0.01 MIL`

The `drc_diff_pair_primary_separation_tolerance` environment variable can specify optional minimum and maximum values. The tool converts these values in the following ways:

14.x <code>drc_diff_pair_primary_separation_tolerance</code> Values	Converted to these 15.x properties
blank	Nothing done
minimum value specified (example: 10 mil)	<code>DIFFP_COUPLED_MINUS = 10 MIL</code>
maximum value specified (example: 20 mil)	<code>DIFFP_COUPLED_PLUS = 20 MIL</code>

⚠ If both the `drc_diff_pair_overide` and `drc_diff_pair_primary_separation_tolerance` environment variables are set, The tool only converts the `drc_diff_pair_primary_separation_tolerance`.

Setting Up a UNIX Environment

The tool set operates in a windows environment on UNIX workstations. UNIX contains two major shell families: `csh` and `sh`. The `csh` family includes `tcsh` while the `sh` includes `ksh` and `bash`.

Depending on default login shell you need to perform either of the following steps described in the `csh` or `sh` sections to access the Cadence Allegro tools.

To identify login shell, type following command in a terminal:

```
echo $SHELL
```

Using csh environment to access Cadence Allegro tool set

If you are working in a C shell, you must source the `.cshrc` file to initialize your environment before starting your tool. You can do this in two ways:

- Source the `cshrc` file each time you start the layout editor.
- Copy the contents of the `cshrc` file into your own `.cshrc` file.

To source the `cshrc` file:

- At an operating-system prompt, type

```
source <install_dir>/tools/bin/allegro_cshrc
```

The `install_dir` is the directory in which the layout editor was installed.

⚠️ In release 17.0 location and name of the Cadence provided `cshrc` file has changed. It has been updated to add a default Cadence PATH variable. The presence of Sigrity tools may require an additional location.

Copying the contents of the `cshrc` file into your own `.cshrc` file

1. To install it in your `.cshrc` file, at an operating-system prompt, type:

```
source <install_dir>/tools/bin/allegro_cshrc  
Replace <install_dir> with the root location of the Allegro release.
```

⚠️ This file is normally hidden. To view the file, in your home directory type following in a terminal

```
ls -a
```

Using sh/ksh Environment to access Cadence Allegro

If you are working in a korn shell, you must incorporate the layout editor's `profile` file into your environment before starting the tool. You can do this in two ways:

- Source the `profile` file each time you start the tool.
- Copy the contents of the `profile` file in your own `.profile` file.

To incorporate the `profile` file into your korn shell environment:

- At an operating-system prompt, type:

```
. <install_dir>/tools/pcb/bin/allegro_profile
```

The `install_dir` is the directory in which the layout editor was installed.

⚠️ In release 17.0 location and name of the Cadence provided `profile` file has changed. It has been updated to add a default Cadence PATH variable. The presence of Sigrity tools may require an additional location.

Copying the contents of the `profile` file into your own `.profile` file

1. To install it in your `.profile` file, at an operating-system prompt, type:

```
source <install_dir>/tools/bin/allegro_profile
```

Replace `<install_dir>` with the root location of the Allegro release.

 This file is normally hidden. To view the file, in your home directory type following in a terminal

```
ls -a
```

Starting the Layout Editor from an Operating-System Prompt

When you start a tool from the operating-system prompt, you have the following options:

- Type an Allegro command and do not include the drawing name, or include the name of a new design.
- Type an Allegro command and include the name of an existing padstack, symbol, or layout (without an extension) to be opened.

To start the layout editor from an operating-system prompt:

- Type one of the following Allegro commands:

To open padstacks:

```
padstack_editor [-s  
script] [-p startdir] <padstack (.pad) filename>
```

The arguments for the Allegro command `allegro` are

<code>- product product_Name</code>	Determines the product tier that is run.
<code>-s script_name</code>	Runs a specified script file.
<code>-o journal -j journal</code>	[Default] Starts a journal file that records your Allegro PCB Editor work session. The name of the file is <code><program>.jr1</code> .
<code>-p start_directory</code>	Lets you specify a startup directory. If you start the layout editor with a drawing name that includes a path to the drawing (for example, <code>/home/joe pcb/designs/layout_name</code> (<code>.brd</code> or <code>.mcm</code>), other files created during processing (<code>.log</code> and <code>.jr1</code> files) are created in the directory you specified and not the directory in which the drawing is located.
<code>filename</code>	Specifies a design file. You do not have to include the file type (extension).
<code>-product license_filename</code>	Starts the product based upon the name of the product license file.
<code>-proj cpm_file</code>	Reads the HDL-indicated <code>.cpm</code> file at startup.
<code>-mpsXXX</code>	Standard Cadence mps argument support (This is not typically required.)
<code>database_name</code>	Starts the product with the indicated database name.
<code>-version</code>	Prints the version of the product, then exits.
<code>-nographic</code>	Runs the layout editor in a non-graphic mode but still requires an X server. UNIX operating systems only.

For example:

```
allegro -product <  
product_name  
> -s <  
scriptfile>  
<  
filename>
```

```
apd -product <  
product_name  
> -s <  
scriptfile>  
<  
filename>
```

If you do not include a design name, the tool displays the editor you selected and opens a default file called `unnamed.pad`, `unnamed.dra`, `unnamed.brd`, or `unnamed.mcm`. You can then use the `open` or `new` command to open an existing or new drawing from the user interface.

If you have previously opened sessions of the layout editor, the last saved design in the previous session opens, based on information written to the `master.tag` file.

The `master.tag` file is a text file automatically generated when you launch a session of your layout editor. The file contains the name of the last database that you saved before ending a session. The tool reads this file when you next launch a session and opens the database of that name.

If, for any reason, you do not want the tool to open to the last saved database, you can move or delete the `master.tag` file. The tool then opens a new, unnamed design file. To locate `master.tag`, open the initialization (`.ini`) file, located in your `pcbenv` directory. Search on `directory=` to locate the file.

Starting Allegro PCB Editor

You can start Allegro PCB Editor in one of the following ways:

On Windows

- From the Windows Start menu, select *Cadence PCB 2022 – Allegro PCB Editor 2022*.
- Or
- At the Windows command prompt, navigate to the install directory, `<install dir>/tools/bin`, type `allegro` and press `Enter`.

On LINUX

1. At the shell prompt, type `allegro &` and press `Enter`.
The *Cadence 22.1 Allegro Product Choices* dialog box opens listing all the available product options.
2. Select the required product and product option, and click *OK*.

To stop this dialog box from appearing each time you launch the tool, select the *Use As Default* option, and click *OK*.

Starting Allegro PCB Editor in Safe Mode

If you experience any issue with user customizations or extensions, you can run the tool in the safe mode to debug the issue.

- To start Allegro PCB Editor in the safe mode, run the following command:

```
allegro -safe
```

 The safe mode should only be used to help diagnose issues. It should not be used for production work. Some applications may access local customizations as a prerequisite.

When launching the layout editor in the safe mode, the following are not loaded on startup:

- Local env file (`<HOME>/pcbenv/env`)
- `cds_site` configuration data
- Any user SKILL code
- Pre-registered scripts
- The `ini` file which stores window size and position information
- Most Recently Used files (MRU)
- Remembered Windows positions (`.geo` files)

Setting up a pcbev Directory for Windows or UNIX

The layout editor creates a `pcbev` directory with the `env`, `allegro.ini`, and `allegro.geo` startup files at a location determined by the value of the environment variable `HOME`. The `pcbev` directory stores your window and toolbar preferences. Do not edit these files. Instead, use the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* (`enved` command) to make changes. For additional information, see [The User Preferences Editor](#).

If your initial default directory is inaccessible, you cannot save any of your preferences.

If you have not explicitly set a `HOME` variable, the tool uses a combination of the `HOMEDRIVE` and `HOMEPATH` variables to generate the home directory (`HOMEDRIVE:\HOMEPATH`) on Windows. If the `HOMEDRIVE` and `HOMEPATH` variables do not exist, the tool uses `c:\`.

The layout editor also lets you set the `ALLEGRO_PCBENV` environment variable to override the default location of the `pcbev` directory. You must set the `ALLEGRO_PCBENV` variable before starting the tool, so that the Allegro tool looks for the startup files in the new location.

The `ALLEGRO_PCBENV` must be set at the operating-system level. On UNIX, add it to your `.profile` (`sh/ksh`) or to your `.cshrc` (`csh/tcsh`). On Windows, add it to your user environment variables using the same technique as adding a `HOME` variable, described below. Adding it to your environment file will not work.

Creating or Changing the HOME Variable

The `HOME` variable is used to locate the `pcbev` environment file as well as other required user-specific files. By default, it is not used to store design data. Starting in Release 15.5, you can also set `ALLEGRO_PCBENV` (see above) to modify the location of the `pcbev` directory, and if the `HOME` variable is not set, the default is the standard Microsoft *My Documents* location. On most Windows systems, this defaults to:

```
c:\Documents and Settings\<user login>
```

 Earlier versions of Allegro tools require a `HOME` variable to be set to a directory without any spaces.

To create or change the `HOME` variable for Windows:

1. Right click on *My Computer* and choose *Properties*, or choose *Start – Settings – Control Panel – System*.
2. Choose the *Advanced* tab.
3. Choose *Environment Variables*.
4. In the *User Variables* section, either click *New* or *Edit*.
5. To specify a `HOME` directory located at `d:\work`, for example, do one of the following:
 - a. If you clicked *New* in the previous step, add the following in the *New User Variable* dialog box:
 - b. Variable Name = `HOME`
 - c. Variable Value = `d:\work`
 - d. If you clicked *Edit* in the previous step, modify the following in the *Edit User Variable* dialog box:
 - e. Variable Name = `HOME`
 - f. Variable Value = `d:\work`
6. Choose *OK* to save the setting and dismiss the dialog box.
7. Choose *OK* to save and dismiss the *Environment Variables* dialog box.
8. Choose *OK* to save and dismiss the *System Properties* dialog box.

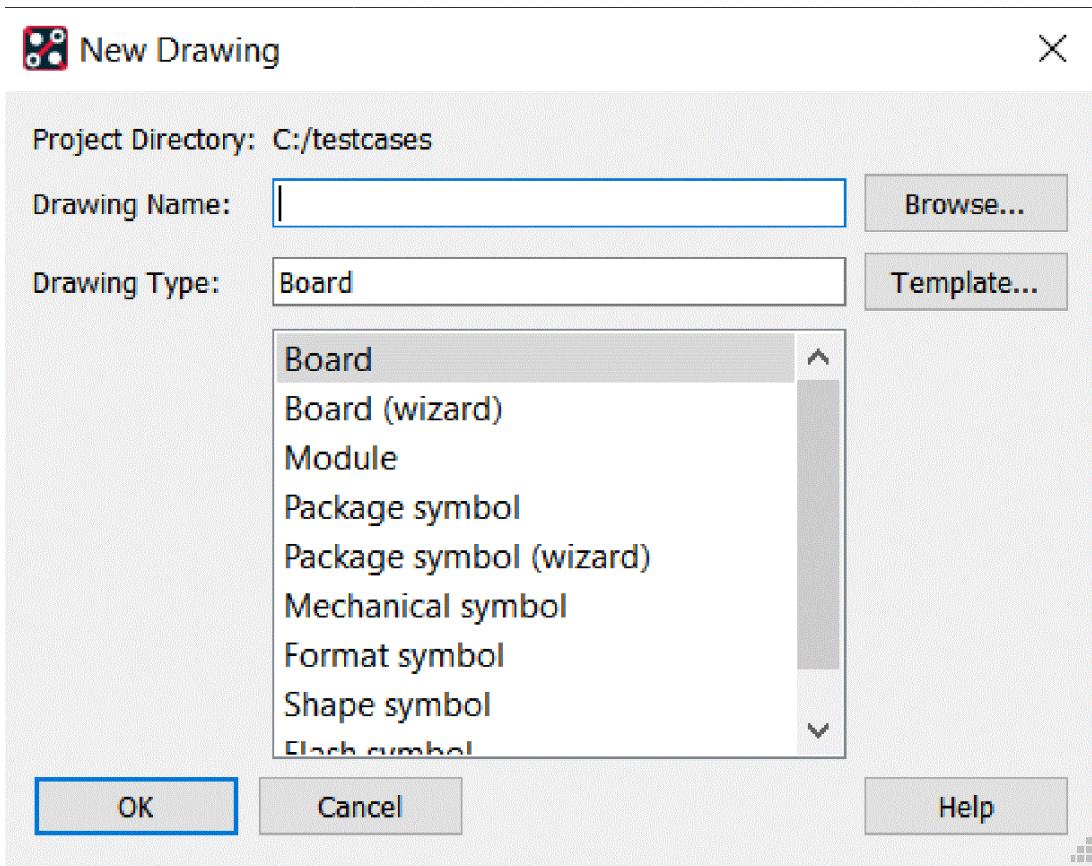
The next time you start your layout editor, the `d:\work\pcbev` directory is created. The tool looks in this location for startup files (`env`, `allegro.ini`, `allegro.geo`, and so on.)

PCB Editor: Creating New Designs

Once your layout editor is running, you can open new and existing drawings using the appropriate items in the *File* menu. (If you have created designs in previous sessions, the editor opens the last saved design, based on information written to the `master.tag` file, described above.)

When you create a new design file, you must specify the type of design you want to create, using the New Drawing dialog box to select whether you want to create a board file or a symbol file.

Figure 2.46: The New Drawing Dialog Box (PCB Editor)



The choices are:

Layout

Creates a board file (.brd) or design file. You create a design database in this editing mode. Use this file to perform such tasks as component placement, board or design routing, and other functions.

Board (wizard)

The board wizard is designed either to help beginning users create a design in Allegro PCB Editor (board wizard is not available on Allegro Package Designer) or for experienced users who want a quick way to create a basic framework for a design as a foundation for a more complex design database. You can also use the board wizard to import custom design data by way of user-defined templates and technology files.

A template file is an existing user-created .brd file containing customized data. Information that you should include in a .brd template file includes default parameter settings, company-default subclasses, and color-to-layer assignments.

⚠ The template file should *not* contain any data on ETCH/CONDUCTOR, PIN, or VIA classes.

The board wizard accepts the following data from a template file. Board units and board origin are data contained in the template file that can be replaced. The wizard cannot replace the following parameters, but they can be modified after you create a new layout:

- Drawing size
- Board outline
- Spacing constraints
 - Minimum line width
 - Minimum line to line spacing
 - Minimum line to pad spacing
 - Minimum pad to pad spacing

- Package and route keepins
- Grid definitions
- Cross-section definitions

If the template file contains only two ETCH/CONDUCTOR layers, the wizard lets you add more layers and defines them as routing layers or power planes. If additional layers are defined in the template, this functionality is disabled in the wizard.

If you import data using a template file and a tech file, note that the data in the tech file takes precedence over data brought in from the template. A tech file template should include constraint (DRC) rules and layer stack-up information. See the *Defining and Developing Libraries* user guide in your documentation set for details on technology files.

Templates and technology files that you can import into the design database should contain the following default parameter settings:

- Company-default subclasses
- Color-to-layer assignments
- Constraint (DRC) rules
- Layer stack-up information
- Mechanical (.bsm) symbols

If you choose not to load data from template or technology files, Board Wizard lets you input the data manually, from the wizard's user interface screens.

For procedural details, see the *Allegro PCB and Package Physical Layout Command Reference*.

Symbol

You create symbols for a design in the symbol editing mode. The tool appends the appropriate filename extension when you save a symbol.

There are two files associated with a symbol. The raw, unprocessed, drawing file has a .dra filename extension. When you choose *File – Create Symbol (create symbol)* command from the symbol editing mode, the .dra file is compiled into the appropriate binary file — Package (.psm), Format (.osm), Mechanical (.bsm), Shape (.ssm), or Flash (.fsm).

The layout editor automatically creates a symbol every time you save a drawing (.dra) when you are in the Symbol Editor. You no longer need to compile the symbol and save the drawing in two separate steps.

Set the environment variable, *no_symbol_onsave* to restore the legacy behavior and allow the layout editor to compile the symbol and save the drawing in two steps.

1. Choose *Setup – User Preferences* to display the User Preferences Editor.
2. Choose *Drawing* and then click the *no_symbol_onsave* environment variable.

See the *Defining and Developing Libraries* user guide in your documentation set for information about symbol files.

The symbol editor lets you create the following types of symbols:

Package Symbol

Creates a new component symbol such as an IC. The tool saves package symbols to the symbol library, by means of *File – Create – Symbol*, and appends the file name that you specify with a .psm extension.

Mechanical Symbol

Creates a drawing symbol such as a card edge connector or a board/design outline. The tool saves mechanical symbols to the symbol library and appends the file name that you specify with a .bsm extension.

Format Symbol

Creates a drawing symbol such as a legend or a company logo. The tool saves format symbols to the symbol library and appends the file name that you specify with an .osm extension.

Shape Symbol

Creates a drawing symbol such as a special shape for a padstack. The tool saves mechanical symbols to the symbol library and appends the file name that you specify with an .ssm extension.

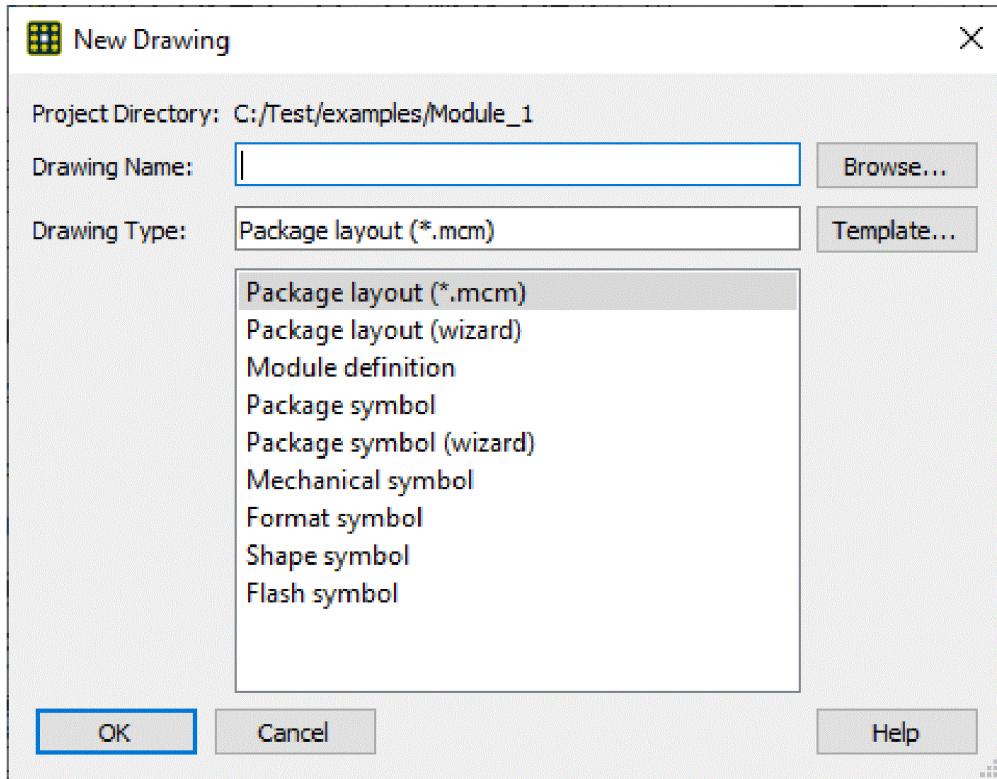
Flash Symbol

Creates a flash symbol such as a thermal pad for Raster formats. The tool saves flash symbols to the symbol library and appends the file name that you specify with an .fsm extension.

APD: Creating New Design

Design work, which entails symbol and layout creation, occurs within the context of a design that you create. Use the *new* command to specify type of drawing that you want to create: component or symbol (Figure 2-31).

Figure 2.46: New Drawing Dialog Box



Drawing types include:

- Component/multi-chip – Creates a design file (.mcm). A design file represents the drawing database. Use this file to perform such tasks as component placement, design routing, and other functions.
- Module definition – Creates a module file (.mdl). Module files are collections of physical entities (which can include other modules). Modules may or may not have logic (represented by nets, components, and so on) or a block (a collection of schematic information used by a schematic tool) associated with them. They can be permanently stored as module definition databases in library files.
- Symbol – Creates a symbol file. APD saves these databases as files with the .dra extension. This invokes the Symbol Editor, from which you can create the following types of symbols:
 - Component symbol – Manually creates a new component symbol such as a die or a discrete. APD saves component symbols to the symbol library and appends the file name that you specify with a .psm extension.
 - Component symbol (wizard) – Creates a new component symbol such as a die or a discrete using the Package Symbol Wizard.
 - Mechanical symbol – Creates a drawing symbol such as a card edge connector or a design outline. APD saves mechanical symbols to the symbol library and appends the file name that you specify with a .bsm extension.
 - Format symbol – Creates a drawing symbol such as a legend or a company logo. APD saves format symbols to the symbol library and appends the file name that you specify with an .osm extension.
 - Shape symbol – Creates a drawing symbol such as a special shape for a padstack. APD saves mechanical symbols to the symbol library and appends the file name that you specify with an .ssm extension.
 - Flash symbol – Creates a flash symbol (.fsm) used in various artwork processes.

Opening Existing Designs

You can open existing drawings in the following four ways:

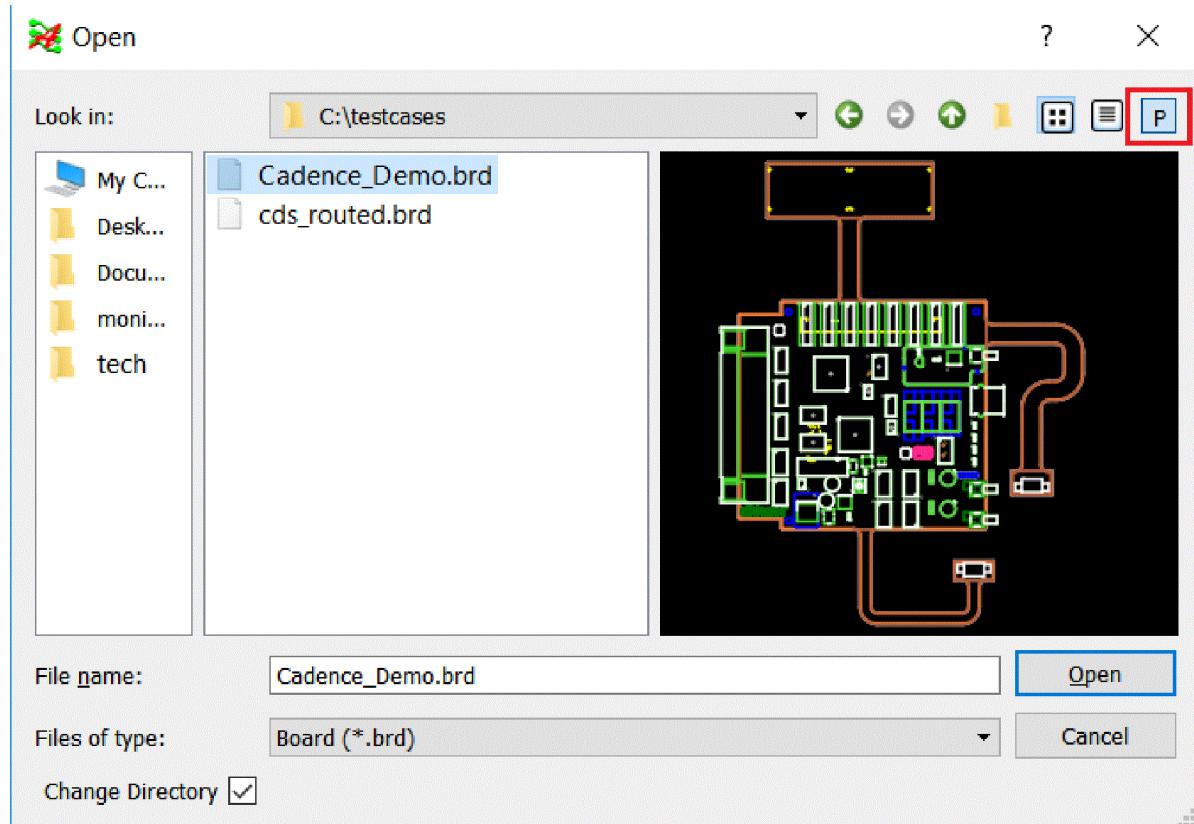
- From an operating system prompt, as described in [Starting the Layout Editor from an Operating-System Prompt](#)

- From Start page, as listed in the *Recent Designs* tab
- From within the layout editor using *File – Open* ([open](#) command)
- Drag and drop design file on the layout editor canvas

⚠ You are prompted to save any changes made to an open design before opening a new file, but may be prohibited from doing so if the database has been locked. For details on database locking, see [Protecting Files with Edit Locks](#).

You can display information for an existing drawing before opening it by using the Quickview window in the Open dialog box. Quickview provides a high-level graphic overview or a summary of properties of the database you select from the list. The information that appears is based on the icon you press in the dialog box. [Figure 2-33](#) is an example. Use preview button, located at the top right corner of the file browser dialog, to toggle the display of design quick view.

Figure 2.47: Quickview in the Open Dialog Box



For additional information on Quickview, see "[Using Data Browsers](#)" in Using the Layout Editor.

Saving Automatically

The layout editor lets you automatically save an active design or symbol at regular intervals when you set the *autosave* environment variable. When the tool saves a design, it automatically generates a file named `AUTOSAVE.brd` (a symbol is saved to a file named `AUTOSAVE.dra`) and places it in the directory that was active when you opened the tool. If you change directories, the tool saves the file to the original working directory. The saved file is kept after you have closed and saved the design or symbol and exited the software.

⚠ The autosave option is automatically disabled if you invoke the database locking command, `file_property`. For details on this feature, see [Protecting Files with Edit Locks](#).

If the autosave time is reached when a command or non-filled shape is active, the tool displays a message that reads "Save Pending." The save executes when the command is completed or when the shape is filled. If you have not executed a command since the last autosave, the tool does not resave the design.

Activating the Autosave Utility

- Set the *autosave* variable in the Environment Editor by choosing *Setup – User Preferences* (*enved* command). See [Managing Environment Variables](#) for details on environment variables.
or
- Before opening a design, execute the following command from the console window prompt:

```
set autosave
```

You can specify the interval at which checkpoint saves are made by using the `set` command and the *autosave_time* variable as follows:

```
set autosave_time = <time>
```

The `<time>` can be set from 10 to 300 minutes. The default is 30 minutes.

Changing the Default Name (AUTOSAVE) of the Generated File

- Running the *enved* command to display the User Preferences Editor dialog box and entering a new value for *autosave_name* in the *Autosave* Category
or
- Using the following command and the *autosave_name* variable

```
set autosave_name = <filename>
```

The tool lets you specify whether a database check is performed when a design or symbol is saved with the autosave facility.

Enabling a Database Check

- Set *autosave_dbcheck* in the User Preferences Editor dialog box
or
- Execute the following command from the console window prompt:

```
set autosave_dbcheck
```

Note that enabling the database check during autosave requires additional processing time. The default is *disabled*.

Disabling the Autosave Facility

- Uncheck *autosave* from the User Preferences Editor dialog box
or
- After opening a design, execute the following command from the command line:

```
unset autosave
```

Suppressing the Overwrite File Confirm

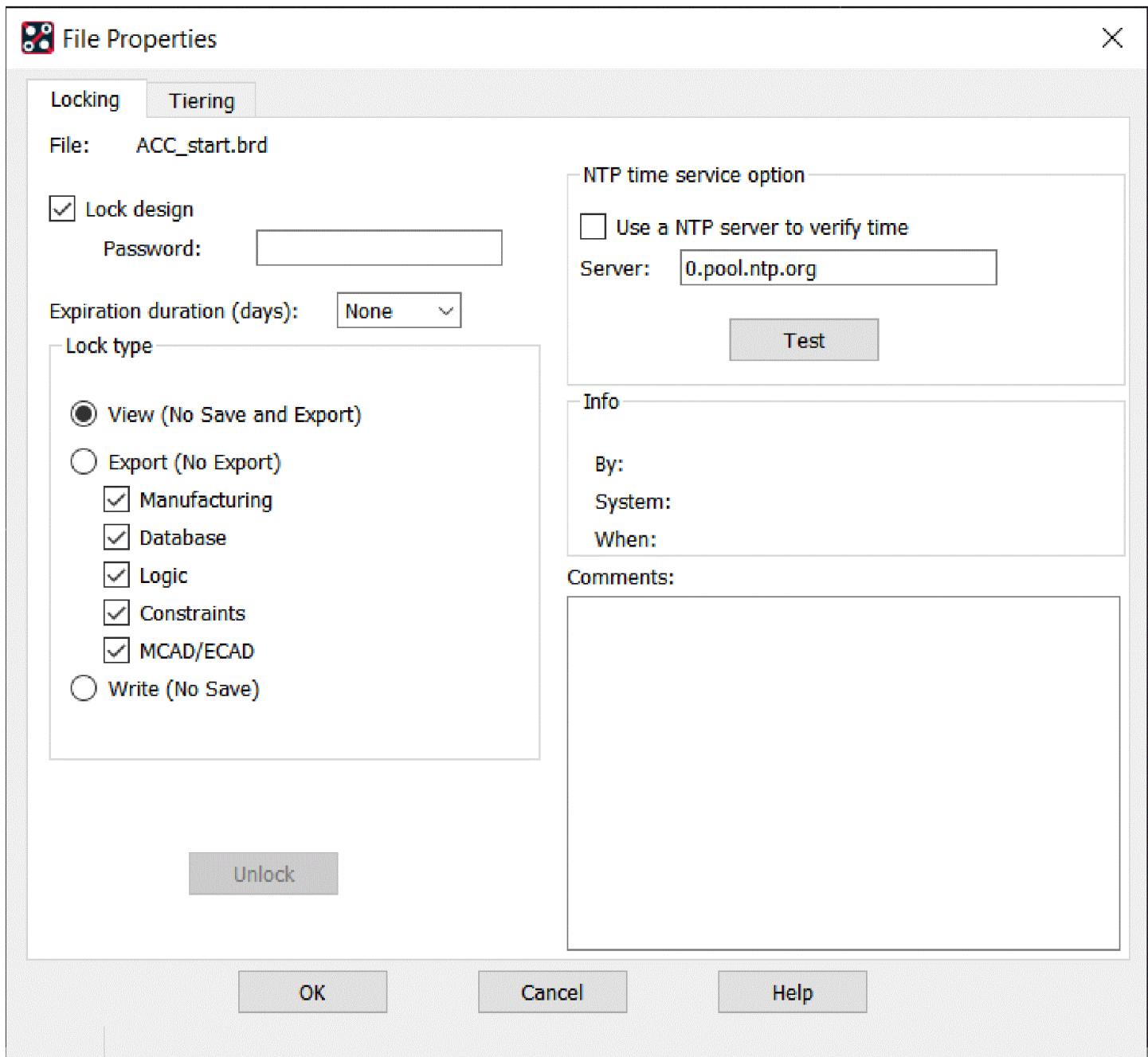
To disable the overwrite confirm that automatically appears when you save an existing file, disable the *noconfirm_savedb* environment variable in the *Drawing* category of the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* (*enved* command). No warning message displays even if saving will overwrite an existing database.

Saving to an Earlier Version

The databases are backward-compatible with their major version number (the number to the left of the dot). This means that databases created in or upreved to any revision within a major version (for example, to 14.1) can migrate between revisions of that version. You *cannot* save any major version to an earlier one, such as 15.x to 14.x, 14.x to 13.x, and so on.

Protecting Files with Edit Locks

You can secure any design database file by choosing *File – Properties* (*file_property* command) to set an optional password-protected database lock. Doing so marks the file as read-only in the database (as opposed to on the platform's operating system). This ensures that the design is not accidentally replaced by you or an unauthorized user when attempting to save over the file.



In addition, you can set database locking to disable the export of design data such as writing techfiles, exporting libraries, and creating modules. A set of export options are available as check boxes with *View* and *Export* locks. These options create different groups of export commands. You can select none, some or all the options. If an option is selected, the export commands belonging to that group are disabled.

Database locking also turns off the autosave environment variable. The locking mechanism does *not* prohibit you from performing an uprev of the database in batch mode; however, batch programs that open databases for writing, such as `netrev` and `netin`, are unable to perform their operations when the database is locked.

When a database lock has been set, editing the file results in an error message, warning the user that the database has been locked for saving. (Edit locking will *not* inform you if another user has the file open.) The lock can be disabled only by entering the password established when the file was locked or, if a password was not set, by unlocking it in the File Properties dialog box or through the `dbdoctor` command. For procedures on locking files through the user interface or at the system prompt, see *File – Properties* (`file_property` command) or *Tools – Database Update* (`dbdoctor` command), respectively, in the *Allegro PCB and Package Physical Layout Command Reference*.

ⓘ It is extremely important that you record any passwords used to lock databases. Cadence does not support the recovery of databases in a locked state due to forgotten passwords.

Because a design might be legitimately opened for updating by any number of users in a large, networked system environment, the File Property dialog box displays the name of the user who locked the file, when it was locked, and on which system it was locked. A comment field allows you to provide additional information. These comments, as well as the option for prohibiting design data export, cannot be altered when the file is locked.

File Types

The layout editor automatically attaches the appropriate extension to the base filename that you specify. These extensions indicate the following file types:

Extension	File Type
.art (default)	Artwork files. ⚠ You can change the default file extension of .art for artwork film filenames by setting the ext_artwork environment variable in the User Preferences Editor, available by choosing <i>Setup – User Preferences</i> (<i>enved</i> command).
.brd	Board file that represents the drawing database
.bsm	Library file that stores drawing or board symbols
.cio	Specifies a file containing a co-design die.
.cml	Used in component design, the Condensed Macro Library (.cml) file type stores the LEF data for those pins of a macro that impact your component design.
.dat	Data files.
.def	Used in component design, the Design Exchange Format (.def) file type is an industry-standard format developed by Cadence Design System for representing digital IC implementation data.
.dfa	Design for Assembly file.
.dpf	Design Partition file.
.dpm	Design Partition file.
.dps	Design Partition file.
.dra	Drawing file. You must create one of these before you create a symbol file. Later, this file is compiled into a binary symbol file.
.drl	NC drill output files. ⚠ You can change the default file extension of .drl for NC drill output filenames by setting the ext_drill environment variable in the User Preferences Editor, available by choosing <i>Setup – User Preferences</i> (<i>enved</i> command).
.fsm	Library file that stores flash symbols.
.jrl	A journal file which contains a record of events — menu picks, keyboard activity, and so on — which are recorded for each session in your layout editor. You can share this data with Cadence Usability staff to help us learn how you use the product, which will assist us in our efforts to improve the user interface.
.ldf	Used in component design, the LEF Definition file type defines libraries and the paths to the LEF files defined in them.
.lef	Used in component design, the Library Exchange Format (.lef) file type is an industry-standard format developed by Cadence Design System for representing digital IC implementation data.
.log	Log file that contains data on processes.
.mcm	Multi-chip module file (APD)
.mdd	Library file that stores module definitions.
.ncr	Output file in Excellon Format 2 for numerically controlled routers.
.osm	Library file that stores format symbols.

.pad	Padstack file.
.psm	Library file that stores package symbols.
.prm	Database parameter file that contains customized parameters exported from one design and imported into another for reuse.
.rou	Output ASCII text file in Excellon format for an NC router based on parameters set in the NC Parameters dialog box, available by choosing <i>Manufacture – NC – NC Route</i> (<code>ncroute</code> command).
.scf	A System Configuration File that specifies the relationship between the <code>.mcm</code> file and the <code>.cio</code> files. The file name of the <code>.scf</code> uses the base <code>.mcm</code> name followed by <code>_codesign.scf</code> . Together, these files are called a design link.
.scr	Script and macro files.
.ssm	Library file that stores shape symbols.
.tap	Output text files that contain NC drill data.
.txt	Text file, such as that used for parameters.

Opening a `.pad` file invokes the Padstack Tool. Opening a `.brd` file starts the Workspace Editor with the layout menu set. Opening a `.bsm`, `.osm`, `.psm`, `.fsm` for `.ssm` file starts the Workspace Editor with the symbol menu set.

When you finish with a `.dra` file in the symbol editor, choose *File – Create Symbol* (`create symbol` command). The tool converts the file to a binary, symbol type file.

The layout editor supports the storage of log files, journals reports, and artwork films in a subdirectory under the board file location. Three environment variables control the output locations:

- `ads_sdreport` – report location
- `ads_sdlog` – log file/journal location
- `ads_sdart` – artwork and NC output

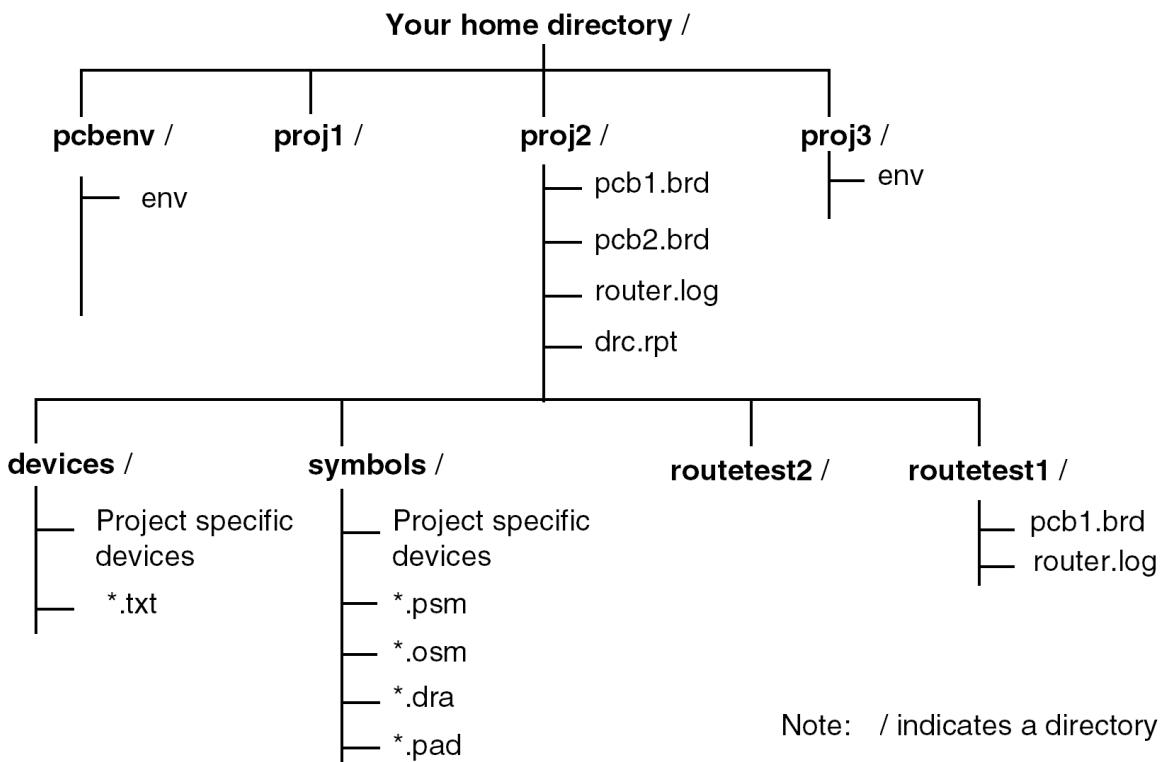
If a directory does not exist, the editor creates one.

You can access these environment variables when you choose *Setup – User Preferences*.

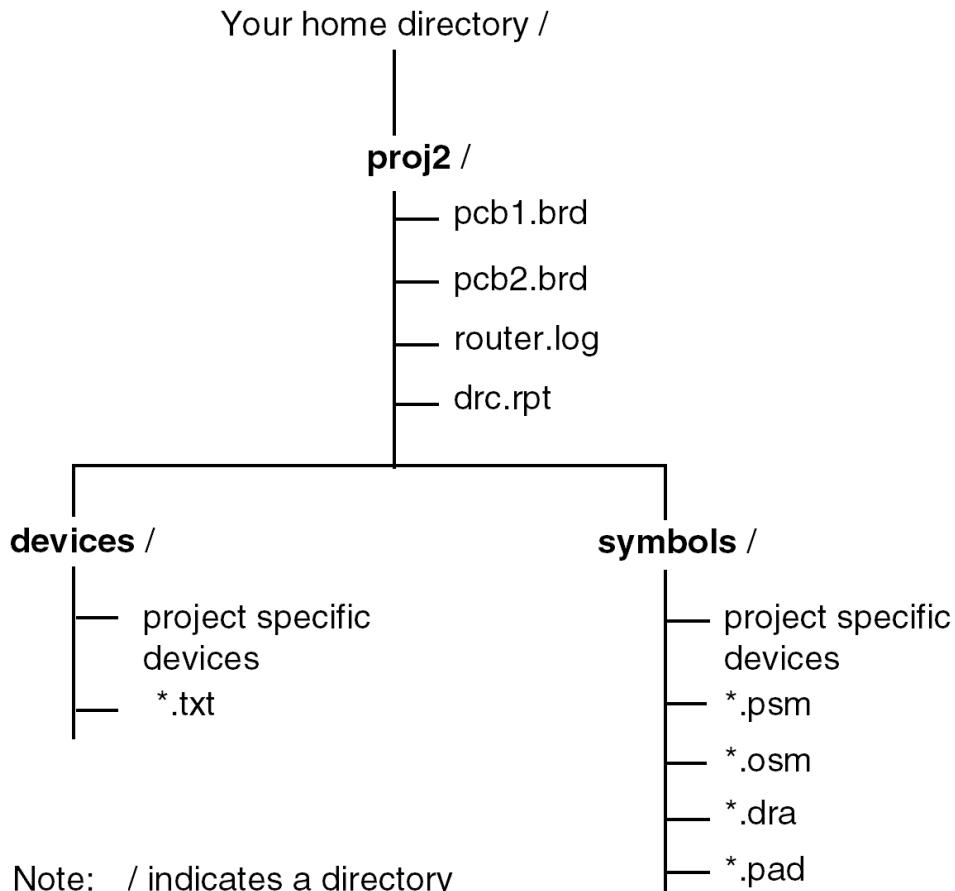
Setting Up a Working Directory Structure

Figure 2-34: shows a suggested directory structure for your projects. This structure lets you have several project directories (for example, `proj1` and `proj2`) and have subdirectories under each project.

Figure 2.48: Suggested Directory Structure for PCB Editor Projects



The symbols and devices directories beneath a project directory contain symbols and devices that are unique to that project. These subdirectories parallel the structure of the library directories supplied by the layout editor in `<install_dir>/share/lib/pcb_lib`.



A project can also contain other subdirectories, such as temporary directories for routing tests that let you run batch routes without replacing log or design files.

Manipulating Design Elements

The Allegro graphical user interface (GUI) adheres to most Microsoft Windows™ standards for pull-down menus, accelerator keys, mouse use, icons, and so on. The layout editor lets you execute commands in one of two methods:

- *command–then–element, or menu-driven editing mode*, in which you first choose a command then elements to be acted upon.
- *preselect use model, or noun-verb*, in which you choose elements to be acted upon, then the command, when you work in an application mode

Using the Mouse

Cadence recommends a three-button mouse. Using a three-button mouse eliminates the need to hold down the Control key while using the right mouse button to pan, zoom in, and zoom out.

Left Mouse Button

Use the left mouse button in conjunction with an active command to select graphic design elements such as lines, pads, and text. The selected feature is highlighted.

⚠ You can move a shape or void with the left mouse button if you enable the `shape_drag_move` design-level environment variable in the User Preferences dialog box, available by running the `enved` command. The shape commands also change the cursor to indicate the legal operation to perform.

You can also use this button to choose commands from menus, tabs, or icons. In dialog boxes with entry fields that list built-in options, the left mouse button can

be used in the data field to display and choose these options (for example, the *Options* window pane).

Keyboard Sequence	Functionality
Shift plus left mouse button	Adds elements to the selection set in an application mode
Control key plus left mouse button	Deselects items by pick.
Double-click left mouse button	Extends left mouse click for specific commands: <ul style="list-style-type: none"> • Using <i>Route – Connect</i> (add connect) to add traces, inserts a via. • Using <i>Edit – Vertex</i> deletes a vertex. Double-clicking the left mouse button on any edge of shape also selects it.

Middle Mouse Button

Press and hold the middle mouse button while moving the mouse in the direction you want to pan and use the view (zoom) features (see [Viewing a Design](#)). If you click the middle mouse button, the system either zooms in or out, based on the direction in which you move the cursor. If you move from top left to bottom right, the display zooms out. If you move from bottom right to top left, the display zooms in. In both cases, a rectangle that depicts the new zoom area appears. You can disable the zoom functionality by setting the environment variable *no_dynamic_zoom*. In Windows, for Wheel mouse devices (middle mouse button is a wheel), the middle mouse button must be defined so that the roam function works correctly. Access the Control Panel to open the Mouse Option Control and check the behavior.

Right Mouse Button

Application-mode and pre-select use model commands are accessible from a right mouse button pop-up menu based on the current selection set. The commands that populate an application mode pop-up menu depend on:

- Current application mode
- Design elements already in the selection set
- Design elements selectable at the current mouse position

Keyboard Sequence	Functionality
Shift + right click	Extends the functionality of the active command for two-button mouse devices, and allows roaming.
Ctrl + right click	Executes strokes. Using <i>Edit – Groups</i> (<i>groupedit</i> command), lets you delete an item from the group.
Double-click right mouse button	Choose and select right mouse pop-up if one is available in command. Drag and drop, click and select pop-up and move to item in pop-up. If you set the <i>no_dragpop-up</i> environment variable, then right-drag-click performs a stroke.

Keyboard Shortcuts

Keyboard shortcuts and accelerators let you perform a number of actions without using the mouse, including changing the view of the design and displaying dialog boxes from the user interface.

Select by Window

Create a selection rectangle by clicking the left mouse button to pick a corner for the rectangle, then holding the left mouse button and dragging the mouse. All applicable items with the rectangle are selected.

Select by Group

While using a command in the menu-driven editing mode, rather than the noun-verb (pre-select) use model, click the right mouse button to display the pop-up menu. Choose *Temp Group*. Choose the elements you want to group together. Each element you choose is highlighted. When you choose all the elements, right-click again to display the pop-up menu and choose *Complete*. All objects selected are added to a preselect buffer.

Deselect Support

In the menu-driven editing mode, rather than the noun-verb (pre-select) use model, use the `Control` key and left mouse button to deselect a selected object in temp group mode (in commands that support this option using the right mouse pop-up menu). To complete the selection, choose *Complete* from the right mouse pop-up menu. If you use the `Control` key while holding down the left mouse button, you can deselect multiple objects using a bounding box.

Viewing a Design

The easiest way to zoom in, zoom out, and move across the design workspace is using the middle mouse button. The button gives you access to all the zoom features available from the menu bar or keyboard commands (except `zoom in`, which is integrated into `zoom points`) without the need to make a menu selection or enter a command at the console window prompt. Use of the middle mouse button also enables you to roam or pan across a design.

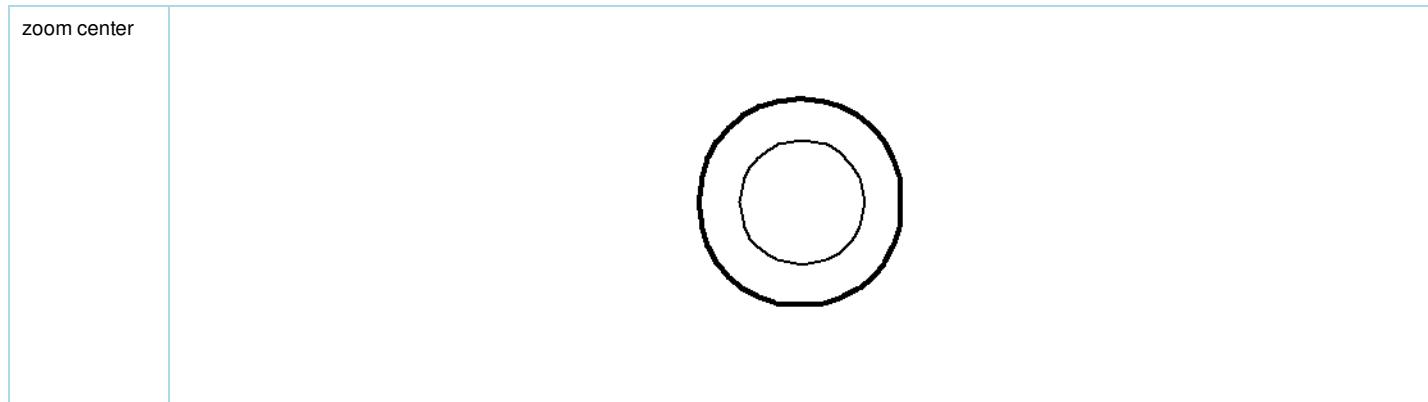
Roaming

Roaming or panning are the terms used to describe the action of moving across a design in the workspace. To pan a design:

- With the cursor inside the design workspace, click and hold the middle mouse button as you drag the cursor across the design. As long as the mouse button remains pressed, you can move all areas of the design into full view. You cannot drag the cursor outside the boundaries of the design.

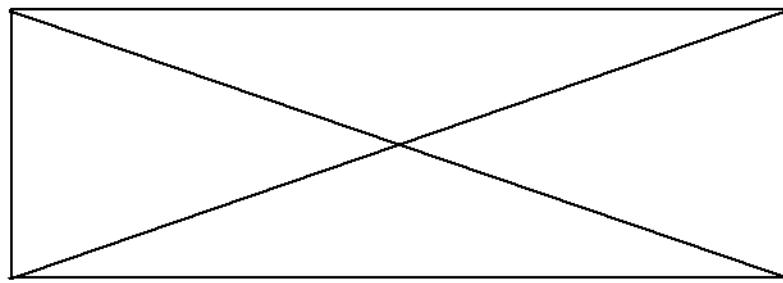
Zooming

Zoom functionality is dependent on the position of the cursor relative to its location when you first click the middle button (the "starting pick"). Movement of the cursor up or down, left or right of this coordinate determines what zoom function is active (as shown in Figure 2-35). Zoom center is the active zoom mode when the cursor is at its starting pick (dynamically displayed in the design as concentric circles). The mode you are in is displayed in the status bar and by way of dynamic shapes that bound the affected areas. The shape geometry associated with each command is:

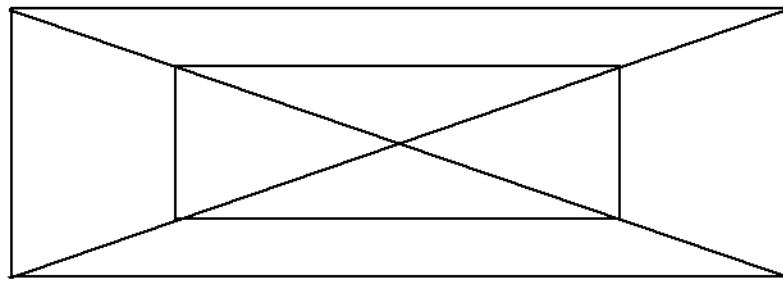


Getting Started with Physical Design
Getting Started--Viewing a Design

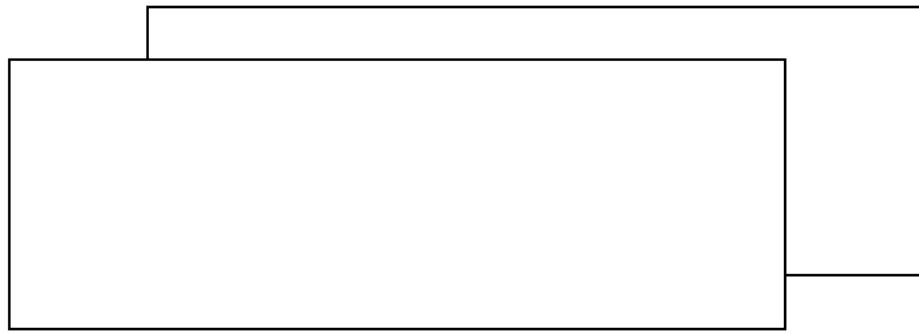
zoom points

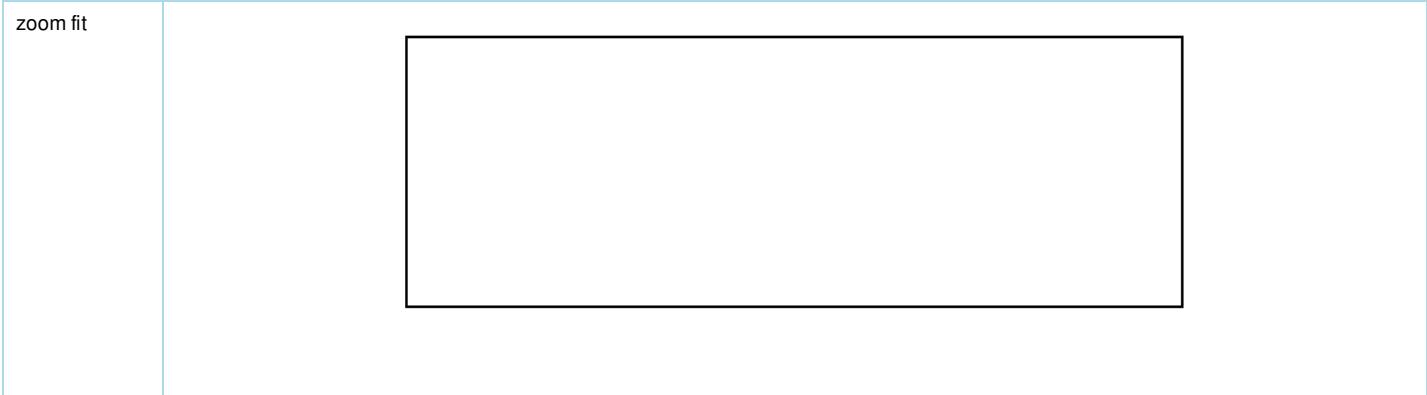


zoom out



zoom previous

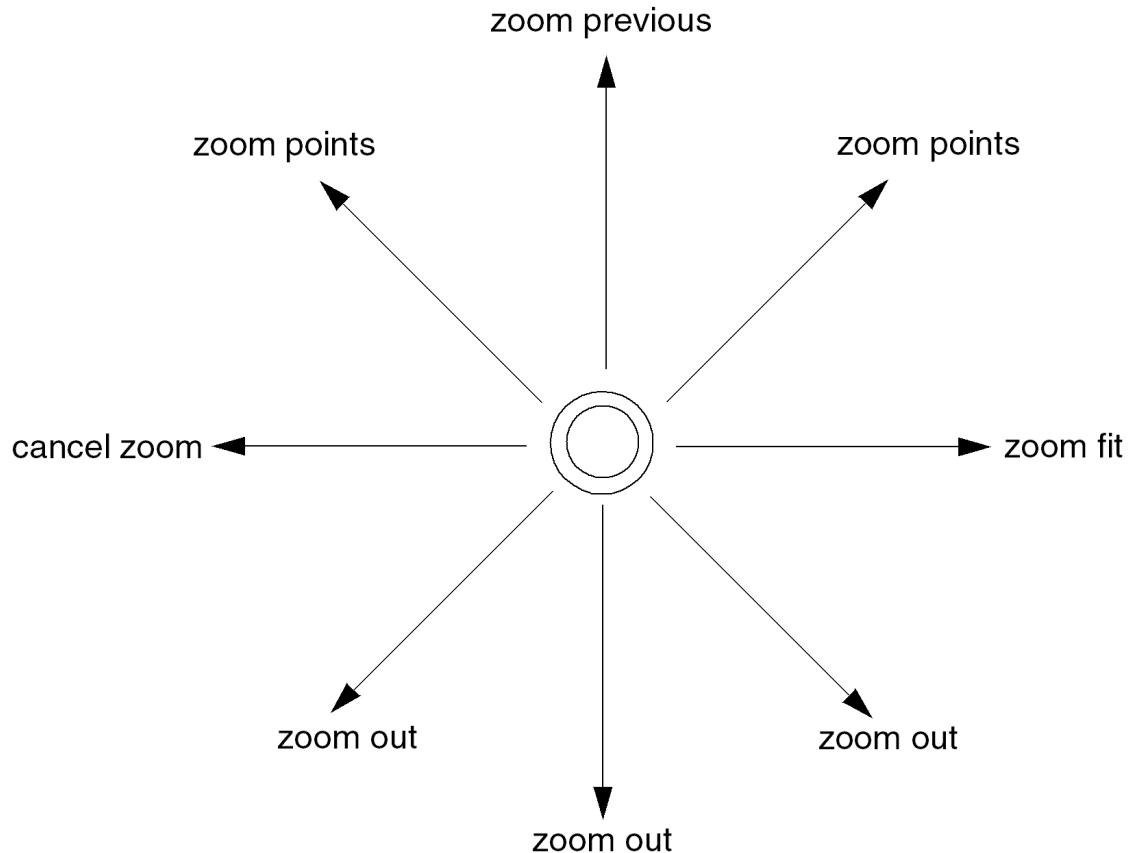




To enhance performance, `zoom out` repaints the design in a minimized draw mode. This "skeletal" view is maintained until the second click completes the zoom operation. While you are in this mode, the following conditions apply:

- Pins, vias, and ratsnests are not drawn
- Line segments are drawn without endcaps
- Lines are drawn in single pixel width
- Shapes are unfilled
- Only reference designator text is drawn
- Layer visibility settings are ignored (all layers are visible)

Figure 2.48: Zoom Modes Relative to Starting Pick



The zoom function remains active until you click the middle mouse button a second time. (Clicking the left mouse button also takes you out of zoom mode.)

If you prefer not to use the dynamic zoom features, you can disable the functionality by setting the environment variable *no_dynamic_zoom* in the User Preferences Editor. By setting this variable, middle mouse button functionality is limited to zooming in or zooming out.

For example, if you move the mouse pointer left, the design will appear to move to the right in the design window.



View Functions

You can control the view of a design by choosing commands from the *View* menu, or by using designated icons, function keys, keyboard accelerators, or mouse strokes. (See "Using Strokes and Associated Commands" in Using the Layout Editor for details on command strokes.)

The following list includes the ways you can zoom in or out on a design, or move the design in the design window.

- Zooming in on a design
- Zooming in on a specific section of a design
- Zooming out on a design
- Zooming out to a full view of a design
- Zooming out to a full view of a design, excluding legends and borders
- Centering an element in the design area
- Zooming back to the last previous window extents

Customizing the User Interface

You can customize your menus by adding, changing, or removing items. The default menus shipped with Allegro products are available at the following location:

<cdsroot>/share pcb/text/cuimenu

Allegro finds the menus with its *MENUPATH* environment variable in the User Preferences Editor, available by choosing *Setup – User Preferences* ([enved](#) command). You should not modify any other file type in this directory as only the menu files are supported for user modification.

As new products are added in a release; new menu files may be added, and Cadence may change the name of any menu file in a release.

For each graphics editor, separate menus exist for the drawing and symbol editors. Use these menus as starting points for customization. Set *MENUPATH* to <*a local directory*> plus \$*MENUPATH* in the local env file to ensure Allegro retrieves customized local menus first. After customization, deposit your version in the following location:

<cdsroot>/share/local pcb/menus

To customize the Cadence-supplied environment, use the operating-system variable *CDS_SITE*, which overrides the default site location:

<cdsroot>/share/local

See "Site Customization" in Managing Environment Variables, which describes the options available with *CDS_SITE*.

Putting the menu in this location lets all your users see this version at startup. To understand this file's format, see:

<cdsroot>/share pcb/examples/skill/DOC/FUNCS/axlMenuDoc.txt

Switching between the symbol and drawing editors reloads the menu, allowing you to perform test edits without restarting the graphics editor. If you have a tool with Skill access, you can also type the following at the command line:

```
skill axlUIMenuLoad("<menu file>")
```

Currently, no mechanism exists to customize the right mouse button pop-up items in the drawing canvas.

Changing Fonts

The layout editor lets you customize the look of the graphical user interface by changing the size and type of the fonts in the console, status, and *Options* windows, and in the Find window pane. This can be convenient if you find it difficult to read information presented in the default size and type.

To change fonts in the user interface:

1. Exit the layout editor, if you have it running.
2. Set the font variables in your environment file.

These variables can also be set in the System dialog box in the Control Panel.

<code>fontSize = -12</code>	where -12 represents the default font size. A larger negative number (for example -20) makes the font larger. Do not use positive numbers in this value.
<code>fontFace = helvetica</code>	where helvetica represents the default font type. Fonts available to you depend on your platform and any user-installed fonts. The value is always a font name.
<code>fontWeight = 500</code>	where 500 represents bolded type. Change the value to 300 to produce unbolded type.

3. Restart the tool.
4. Resize the window, if necessary, to display all information in the larger font size.

You can also change font variables in the User Preferences Editor dialog box by running the `enved` command. Note that you must restart the tool to see the change.

Command Browser

You can access the complete selection of keyboard commands through the **Command Browse** r. This dialog box lets you either run the command or view any online documentation associated with that feature. For procedures on using the Command Browser, see the *Allegro PCB and Package Physical Layout Command Reference*.

Optimizing the Display

The display features are optimized by way of the `display_raster_ops` environment variable. This variable is set to *on* by default. However, based on your hardware, you may notice slower performance while performing tasks that use extensive display capabilities — for example, via shoving while etch editing. If performance slows during these tasks, set the variable to *slow* using *Setup – User Preferences* (`enved` command). This setting disables display enhancement for tasks that bring complex displays into use (other tasks are unaffected).

To disable the `display_raster_ops` environment variable, set the variable to *off*.

Running Commands in the Background

This section is specific to the layout editors on UNIX workstations.

Normally, when you run a command from the command line, you cannot use the Design Window until the command is complete. When you type a command at an operating-system prompt in a UNIX shell window, you cannot use the shell window until the command completes. By running commands in the *background* you are able to continue using the design window or shell window.

While the background job is running, you can look at the contents of the output file with the UNIX commands `more` or `type`.

When a job completes, you are notified with a message in the console window.

Using the Layout Editor

This chapter describes generic operations that apply to a variety of processes and familiarizes you with the user interface and its relationship to them. Detailed descriptions of selection options in the various dialog boxes are available in the *Allegro PCB and Package Physical Layout Command Reference*. Some functionality this chapter describes may not be available in all versions of the layout editors.

Limits

The limits are as follows:

Database resolution	Inches, Centimeters, Mils, and Millimeters: 4 places Microns: 2 places
Maximum number of connections	No limit
Maximum design area size	The maximum design area supported varies based upon design accuracy. The more places of accuracy the smaller the maximum area supported. With 0 places of accuracy we support 20 million design units. For example, with mils and 0 decimal places of accuracy we support 20 million mils (or 20,000 inches). If you increase the design accuracy to 2 decimal places the maximum design extent allowed is 200,000 mils or 200 inches.

Getting Started with Physical Design
Using the Layout Editor--Limits

Maximum number of design layers: (signal, power plane, drafting and so on.)	200 maximum ETCH/CONDUCTOR layers; 200 maximum layers per class (for each class)
Minimum signal width	No limit
Maximum signal width	No limit
Number of connections per net	No limit
Maximum via size	No limit
Number of definable vias	No limit
Via types	Thru, Blind, Buried, Microvia
Maximum number of text strings	No limit
Characters per text item	1000
Maximum text height	No limit
Rotation resolution	millidegrees ($0.001 \deg$)

Maximum errors displayed	No limit
Minimum checking distance	No limit: whatever database resolution is

Unless otherwise indicated, the layout editor only supports uppercase characters. If you enter lowercase characters, the tool converts them to uppercase. Printable characters are generally any key on a standard keyboard with the exception of Tab, Backspace, Enter, function keys, Esc, and navigation keys (Arrows, Home, and so on.).

- ⓘ The exclamation mark ('!') is an invalid character. This character is used as field separator when data is extracted from the database. If you extract any field that contains a '!' character then incorrect data is resulted in the wrong columns. Moreover, if you attempt to save or open a file with '!' in the name, it is reported as illegal character.

Table 3.1: Acceptable characters

Field Name	Length	Acceptable Characters
file name	OS limit ²	no spaces
film name	18	filename ¹
device type	31	All printable except '
directory name	OS limit	OS limit ³
function designator	31 ⁴	All printable except '
package name	31 ⁴	a to z, 0 to 9, -, and _
padstack	31 ⁴	a to z, 0 to 9, -, and _

pin number	31	All printable except '
pin name	31^4	All printable except '
property value	1023	All printable except '
net name	31^4	All except \ and ' ⚠ Set environment variable <i>legacy_character_set</i> to allow backslash (\) in the net names.
reference designator	31	all except * and ' Avoid using period (.) in refdes names because period is used as a delimiter when referring to a pin: for example, <refdes>. <pin number>.
slot name	31^4	all except '
swap type	31	all except '
text lines	1023	except ! ¹
tolerance	1023	all except '
user part number	1023	all except '
value	1023	all except '

1

¹Allows lower case for general text unless on a special layer where it may adhere to more restrictive rules; for example, many layers show Refdes.

² File names adhere to operating systems restrictions except if they are stored in the database, where they assume the least common denominator. For example, a .psm file becomes a package symbol in the database so its least common denominator is the package name restriction. Spaces in the name are not supported. It is strongly suggested that you use lower case, especially for those names stored in the tool database.

³ Directory names follow operating system limitations. The layout editor supports spaces in directory names on Windows.

⁴ The default maximum number of characters is 31. You can set the initial length for new designs to a maximum length of 255 by using the *allegro_long_name_size* environment variable (choose *Setup – User Preferences* ([enved](#) command)). You can change the size in existing designs by choosing *Setup – Design Parameters* ([prmed](#) command) and specifying a new maximum for the *Long Name Size* parameter in the Design tab.

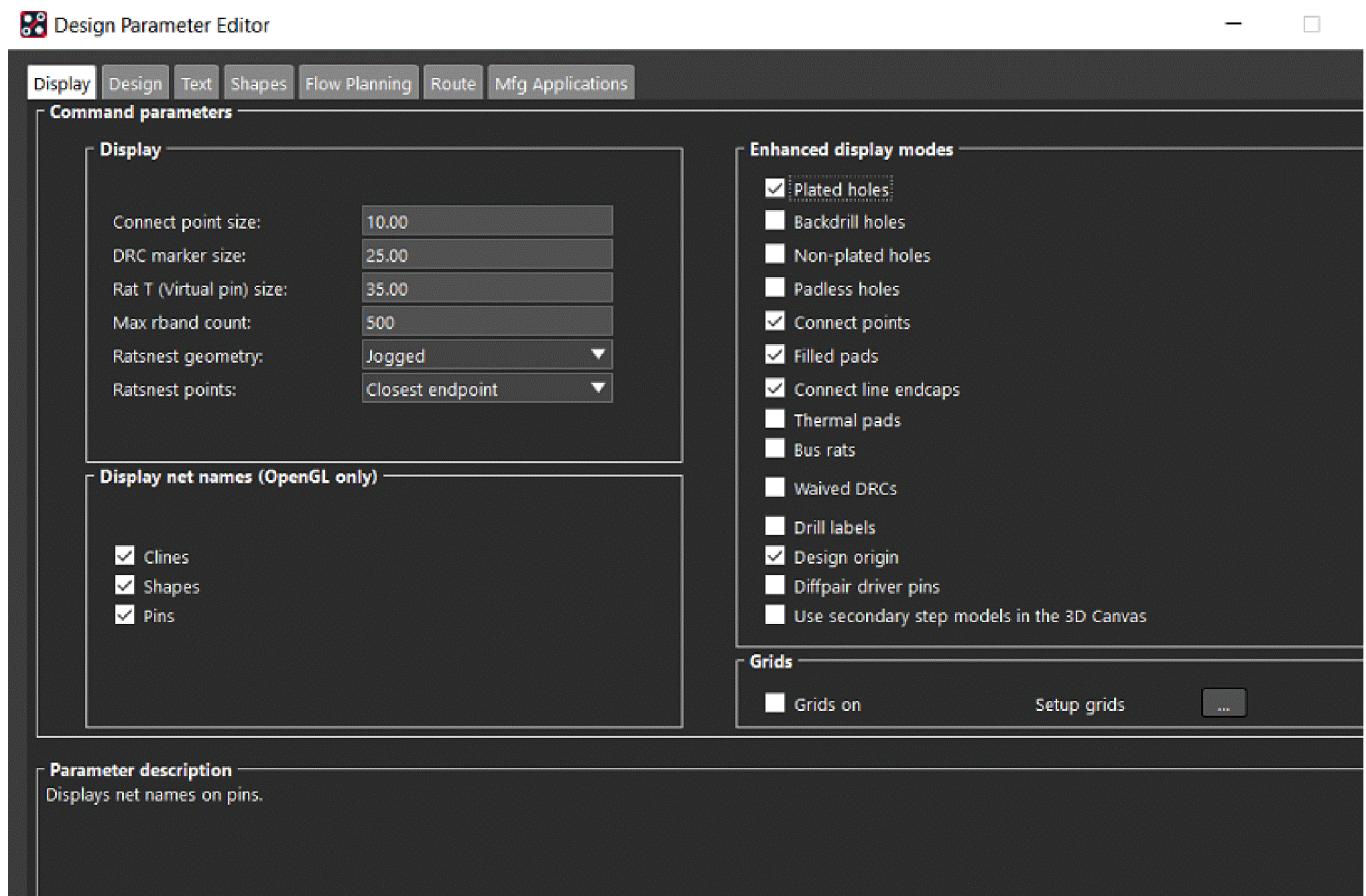
 For releases prior to 16.0, the environment variable *allegro_long_package_name* is only used as part of the uprev process to override the design's default name length limit.

Setting Drawing Parameters

You set drawing parameters in the *Display* and *Design* tabs of the Design Parameter Editor. Use *Setup – Design Parameters* ([prmed](#) command) to access the Design Parameter Editor or right-click in the pre-select use model and choose *Design Parameters* from the pop-up menu that appears.

The Design Parameter Editor organizes common parameters needed to set up a drawing, which entails specifying the following:

- Drawing parameters, including drawing extents, origin, type, and size; database accuracy; and user units
- Text size
- Grids
- Net names



⚠ You can reuse customized parameter settings from one design by exporting them to a database parameter file (.prm) with the *File – Export – Parameters (param out)* command. Then when you initially begin a design, import the .prm file with the *File – Import – Parameters (param in)* command. The **techfile** batch command can also be used to import or export database parameters.

Specifying Text Size

The *Text* tab of the Design Parameter Editor lets you specify the appearance of text in a design. For procedural information on formatting text, see the **define text** command in the *Allegro PCB and Package Physical Layout Command Reference*.

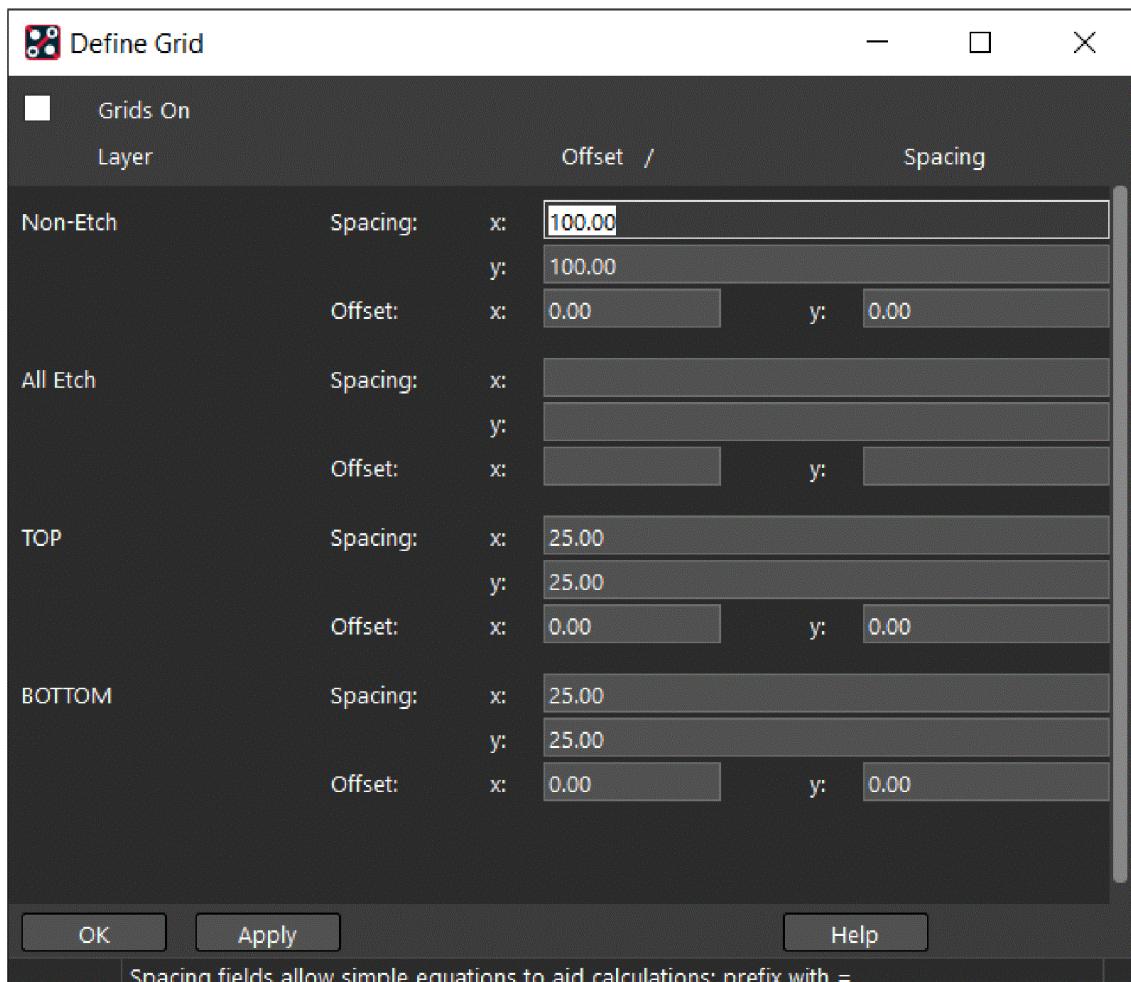
Specifying Grids

The *Display* tab of the Design Parameter Editor lets you access the *Define Grids* dialog box, where you set the x and y values for both ETCH/CONDUCTOR and non-ETCH/CONDUCTOR grids in a design. It also lets you customize the grid for each ETCH/CONDUCTOR layer in a design. For procedural information, see the [define grid](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

All drawings, except Autoplacement, interactive routing, and Autorouting use non-ETCH/CONDUCTOR grid. All non-ETCH/CONDUCTOR layers use the same, single-increment grid with the grid points spaced evenly across the design.

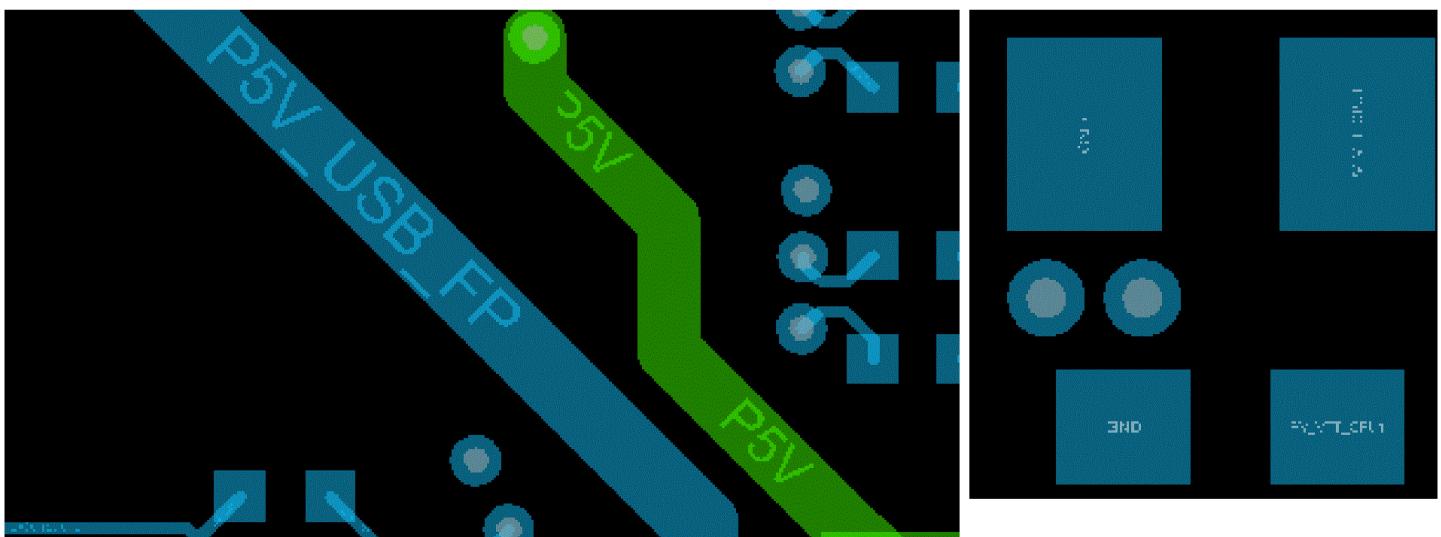
ETCH/CONDUCTOR grids are dedicated routing grids for both interactive and autorouting. You can use a separate x, y grid for each ETCH/CONDUCTOR layer in a design. In addition, you can set a single increment value for each ETCH/CONDUCTOR grid, or you can set different values for non-ETCH/CONDUCTOR grids and ETCH/CONDUCTOR grids.

You can enter values into the Grids Display dialog box to reset the point of origin for x and y, as well as the spacing between the grid points for x and y. The default point of origin for all layers is x=0, y=0. The default increment setting for non-ETCH/CONDUCTOR layers is x=100, y=100. For ETCH/CONDUCTOR layers, the default setting is x=25, y=25.

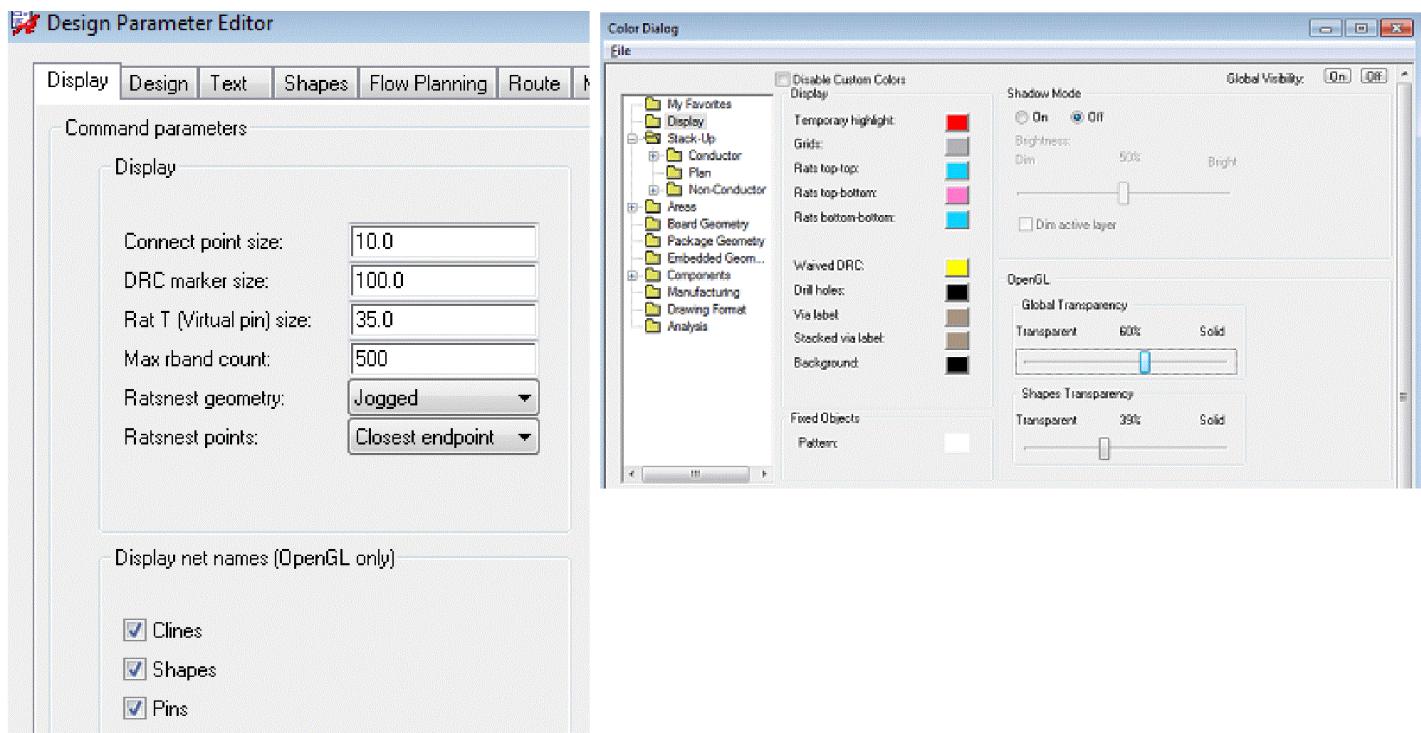


Displaying Net names

You can set the display for net names within the clines and bond wires, pins, and shapes. If set, when moving the wire bonds (push and shove) the cline segment attached to the shoved bond fingers and bond wires show the net name.



The *Display* tab of the *Design Parameter Editor* lets you set the display for net names. This feature is enabled by default. For displaying net names enable *OpenGL* and set transparency to less than *Solid* level in the *Color Dialog* box.



About Classes and Subclasses

In the Allegro tools, categories of drawing elements are called *classes*. Classes represent all types of visible items in the design. A few examples of classes are:

ETCH/CONDUCTOR	Represents pieces of copper forming electrical connections.
PINS	Represents defined pads and holes.
BOARD	Represents the physical outline of the design and other geometry related to the PCB.
PACKAGE	Represents the physical components of the design.

The parts of the drawing in each class are called *subclasses*. Each class can contain many subclasses, including some that you define.

Classes and subclasses identify how every element is to be used in a design. For example, *Add – Line* ([add line](#) command), used when Board is the active class, adds a simple geometric graphic element to a design. The same command, used when ETCH/CONDUCTOR is the active class, adds a connecting line of etch/conductor to the design because the command correlates the function with the class of element.

Subclasses allow a further degree of classification that allows the tool to treat data more specifically. For example, ETCH/CONDUCTOR has two pre-defined subclasses associated with it: Top and Bottom (thus eliminating the necessity of referring to element types by layer number). You also have the option of defining subclasses. (See [Creating User-Defined Subclasses](#).)

Table 3-2 lists groups of classes and their pre-defined subclasses. Note that the Allegro product you are running may not include all the classes/subclasses listed here. In addition, the subclasses in a design vary depending on layers added to or deleted from it.

For more information see, [Classes and Subclasses in Layout Editors](#).

To view colors assigned to the classes and subclasses in the design, choose *Display – Color/Visibility* ([color192](#) command), described in the *Allegro PCB and Package Physical Layout Command Reference*.

Classes and Subclasses		
Group	Class	Subclasses

Geometry	Board	OUTLINE	PLATING_BAR
		ASSEMBLY_NOTES	TOOLING_CORNERS
		DIMENSION	PLACE_GRID_TOP
		PLACE_GRID_BOTTOM	TOP_ROOM
		BOTTOM_ROOM	BOTH_ROOMS
		SWITCH_AREA_TOP	SWITCH_AREA_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		ASSEMBLY_DETAIL	SOLDERMASK_TOP
		SOLDERMASK_BOTTOM	OFF_GRID-AREA
		ASSEMBLY_TOP	ASSEMBLY_BOTTOM
Package	Package	PLACE_BOUND_TOP	PLACE_BOUND_BOTTOM
		PIN_NUMBER	PAD_STACK_NAME
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		BODY_CENTER	SOLDERMASK_TOP
		SOLDERMASK_BOTTOM	DISPLAY_TOP
		DISPLAY_BOTTOM	MODULES
		DFA_BOUND_TOP	DFA_BOUND_BOTTOM
		PASTEMASK_TOP	PASTEMASK_BOTTOM
		SHAPE PROBLEMS	NOGLOSS_ALL
Manufacturing	Manufacturing	PHOTOPILOT_OUTLINE	NOGLOSS_BOTTOM
		NOGLOSS_TOP	BACKDRILL-FLAG-TOP
		NCLEGEND (combines former NCDRILL_LEGEND and NCDRILL FIGURE)	BACKDRILL-FLAG-BOT

		NO_GLOSS_INTERNAL	PROBE_TOP
		PROBE_BOTTOM	AUTOSILK_TOP
		AUTOSILK_BOTTOM	NO_PROBE_TOP
		NO_PROBE_BOTTOM	SHAPE PROBLEMS
		FIXTURE_BOTTOM	FIXTURE_TOP
	Drawing -----	OUTLINE	TITLE_BLOCK
		TITLE_DATA	REVISION_BLOCK
		REVISION_DATA	
Stack-Up	DRC	TOP	BOTTOM
		THROUGH ALL	PACKAGE_TOP
		PACKAGE_BOTTOM	I
	ETCH	TOP	BOTTOM
	Anti-ETCH	TOP	BOTTOM
		INTERNAL LAYERS	THROUGH ALL
	Pin	TOP	BOTTOM
		SOLDERMASK_TOP	SOLDERMASK_BOTTOM
		PASTEMASK_TOP	PASTEMASK_BOTTOM
		FILMMASKTOP	FILMMASKBOTTOM
	Via	TOP	BOTTOM
		SOLDERMASK_TOP	SOLDERMASK_BOTTOM
		PASTEMASK_TOP	PASTEMASK_BOTTOM
		FILMMASKTOP	FILMMASKBOTTOM
Components	Refdes	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM

		DISPLAY_TOP	DISPLAY_BOTTOM
Component 	ASSEMBLY_TOP	ASSEMBLY_BOTTOM	
	SILKSCREEN_TOP	SILKSCREEN_BOTTOM	
	DISPLAY_TOP	DISPLAY_BOTTOM	
Device Type 	ASSEMBLY_TOP	ASSEMBLY_BOTTOM	
	SILKSCREEN_TOP	SILKSCREEN_BOTTOM	
	DISPLAY_TOP	DISPLAY_BOTTOM	
Tolerance 	ASSEMBLY_TOP	ASSEMBLY_BOTTOM	
	SILKSCREEN_TOP	SILKSCREEN_BOTTOM	
	DISPLAY_TOP	DISPLAY_BOTTOM	
User Part Number 	ASSEMBLY_TOP	ASSEMBLY_BOTTOM	
	SILKSCREEN_TOP	SILKSCREEN_BOTTOM	
	DISPLAY_TOP	DISPLAY_BOTTOM	
Areas	Route Keepin	THROUGH ALL	
Route 	THROUGH ALL	TOP	
	BOTTOM		
Via Keepout 	TOP	BOTTOM	
	THROUGH ALL		
Package Keepin	THROUGH ALL		
Package Keepout	THROUGH ALL	TOP	
	BOTTOM		

Creating User-Defined Subclasses

You can create user-defined ETCH/CONDUCTOR and non-ETCH/CONDUCTOR subclasses in the layout editor. The ETCH/CONDUCTOR subclasses identify the layers or cross-section of the design. There are several non-ETCH/CONDUCTOR subclasses that you can create, including the following:

- Board
- Component Value
- Device Type
- Drawing Format
- Manufacturing
- Analysis
- Package Geometry
- Ref Des
- Tolerance
- User Part Number

Use the [define subclass](#) command to create a subclass. Alternatively, use the *Setup – Subclasses* menu command.

For information on creating both types of subclasses, see the [define subclass](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

User-Defined Subclass Mapping for DFM Checks

The mask type and layer association of a non-etch subclass to its corresponding DRC subclasses is determined by the subclass names:

- `SOLDER`, `PASTE`, or `SILKSCREEN` substring: Indicates that the DRC subclass is to be treated as soldermask, pastemask, or silkscreen, respectively.
- `_TOP` or `_BOTTOM` suffix: Indicates that the subclass is to be associated with the `TOP` or `BOTTOM` layer, respectively.

However, if a user-defined non-etch subclass does not follow the defined naming criteria, a configuration file is used to identify the `TOP` or `BOTTOM` layer association and the subclass type (mask or silkscreen).

A sample configuration file, `dfmUserDefinedSubclassMapping.txt`, is available at the following location: `<installation_directory>/share/pcb/dfm`.

Creating Custom Subclass Mapping

To create custom subclass mapping, follow these steps:

1. Copy the sample configuration file to the working directory.
2. Add custom subclasses with mapping details as instructed in the configuration file.
3. If the same rules need to be applied at the project, site, or organizational level, copy this file to the `CDS_SITE` location.

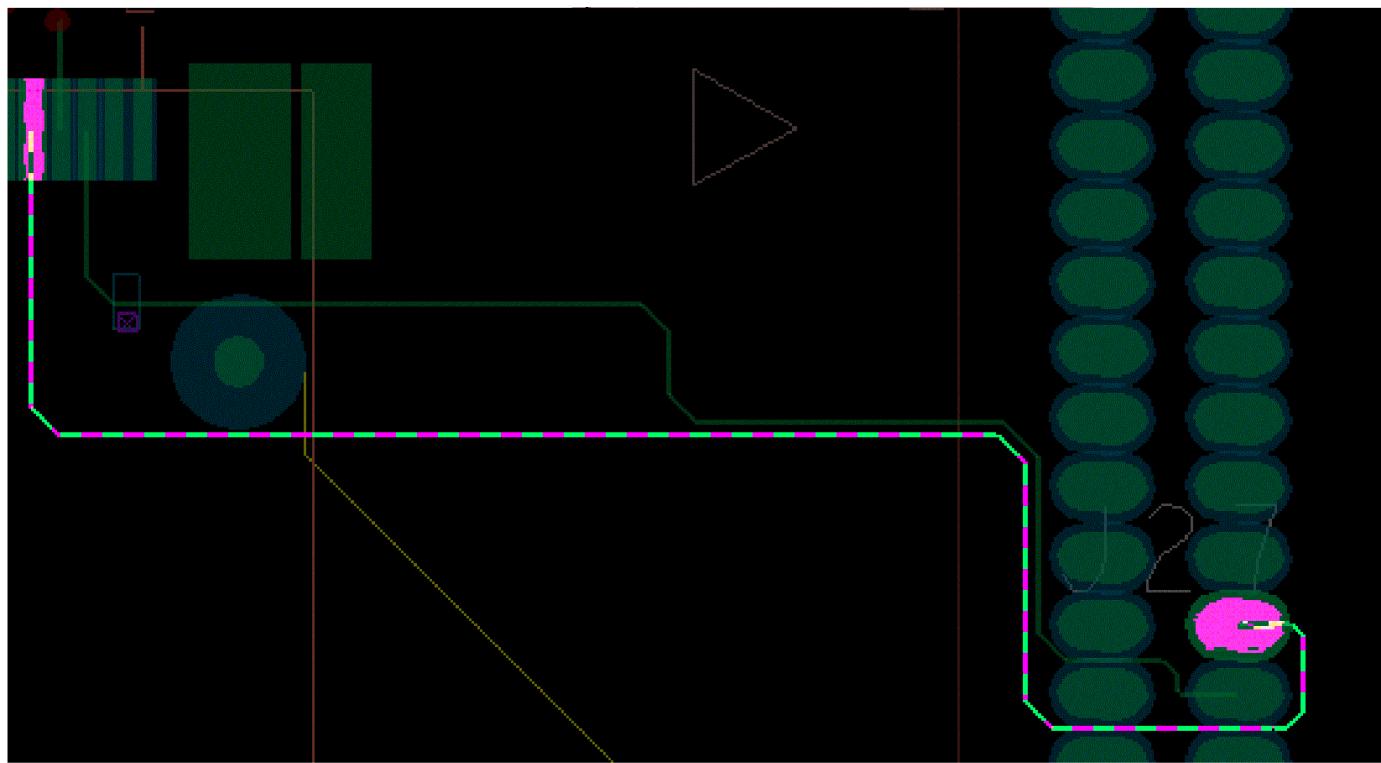
Working with Highlighting and Coloring

Accentuate an element by using the following methods:

- Use a highlight pattern comprising the element's base subclass color and the temporary highlight color defined in the *Display* category of the Color dialog box.
- Override the base subclass color with a custom color but without applying a highlight pattern.
- Assign a custom color plus a highlight pattern.

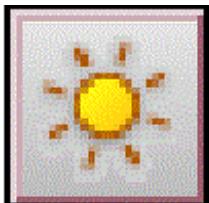
Highlighting Design Elements

Highlighting accentuates certain elements, often nets, with a pattern—or striping—rather than a color, to locate them more easily during debugging. Once the element becomes highlighted in the design canvas, its name also displays in a bold font in the *Nets* section of the Color dialog box.



⚠️ Striping is only visible when the `display_nohilitefont` variable is disabled.

- Menu: *Display – Highlight*.



- Icon:
- Color Dialog: Select a cell (net only) in the *Nets* grid, then right-click and choose *Set Highlight State*.
- Pre-selection mode: Hover over an element, then right-click and choose *Highlight*.

Assigning Colors to Design Elements

Colors can be assigned to Nets, Symbols, Pins, DRCs, Groups, and Functions using the following methods:

- Menu: *Display – Assign Color* ([assign color](#) command), set the Find window pane, and

choose a color from the *Options* window pane.

- Icon: Placeholder for new Icon
- Color Dialog: Use the *Nets* grid to color Net elements.
- Pre-selection mode: Hover over an element, right-click and choose *Assign Color*, then choose a color from the palette that displays.

When you choose *Display – Assign Color*, the following displays in the Options window pane, where you assign a custom color and also choose a highlight pattern.



When you right-click and choose *Assign Color* from the pop-up menu, the following palette displays, from which you can assign a color as well as highlight an element:



The element's color changes in the design canvas and in the *Nets* section of the Color dialog box.

Unassigning Colors

Color overrides can be removed by using the dehighlight command.

- Menu: *Display – Dehighlight*; set the Find window pane, or work with elements the Options window pane.



- Icon:
- Color Dialog: Select a cell in the *Nets* grid, then right-click and choose *Clear Custom Color*.
- Pre-selection mode: Hover over an element, then right-click and choose *Dehighlight*.

The Color Dialog Box

The Allegro layout editors support a palette of 192 modifiable colors, 96 of which appear at one time in a primary color palette, which is the Cadence default, and another 96 which appear in secondary palette, used for customization. The first 24 positions are reserved for colors used in pre-Release 16.0 databases. Choose *Display – Color/Visibility* ([color192](#) command) to display the *Color* dialog box, which comprises the *Layers* and *Nets* grids.

The *Layers* grid primarily controls the color and visibility settings of classes and subclasses, along with levels of transparency for the design and shapes. Use the *Layers* grid to also control shadow dimming, highlighting, ratsnest display, waived DRCs, and drill holes. You can create your own unique colors or palettes that may be saved to external `.col` files and then applied to other designs. You can also search for and display subclass layers by entering text in the *Filter layers* field.

The *Nets* grid allows each element of a net, including clines, pins, vias, shapes, and rats, to be uniquely color coded to differentiate them from other nets or net elements on a layer.

The *Favorites* tab allows you to maintain and use your favorite layers.

The *Visibility Pane* tab lets you customize the *Visibility* window pane.

You can add subclass layers to the *Favorites* or *Visibility Pane* tabs from the pop-up menu. In each of the tab, hovering over a color swatch under *Available Colors* displays the class and subclass where the color is used.

You can re-use customized layer or net colors defined in one design in other designs by creating a database parameter (`.prm`) file with *File – Export – Parameters* ([param out](#) command) and choosing to include the *Color Layer* and *Color Net* parameters.

Using the Layers Grid

Use the *Layers* section of the *Color* dialog box for the following tasks:

- [Assigning Subclass Colors and Enabling Visibility](#)
- [Controlling Ratsnest Colors](#)
- [Setting Graphics Transparency](#)
- [Creating My Favorites’ Folder](#)
- [Customizing Design Colors](#)

Figure 3.1: Layers Grid of the Color Dialog Box



Assigning Subclass Colors and Enabling Visibility

The *Layers* grid lets you assign color and visibility to individual subclasses or to quickly enable or disable color and visibility settings for entire subclasses. The color boxes allocate color across a column or row, as Figure 3-2 shows. The white boxes control visibility.

Figure 3.2: Assigning Colors and Enabling Visibility

The diagram illustrates the 'Layers' grid with various controls for visibility and color assignment. A dotted vertical line on the left separates controls from the grid itself. On the left, four text labels with arrows point to specific features: 'Choose this box to enable visibility globally' points to the top-left white box; 'Choose this box to enable visibility of entire column' points to the second column's header; 'Choose this color box to apply color across entire row' points to the first row's header; and 'Choose this box to enable visibility of entire row' points to the bottom row's header. The grid itself has rows labeled 'Subclasses' (All, Top, Gnd, Sig1, Sig2, Vcc, Vcc1, Sig3, Sig4, Gnd1, Bottom) and columns labeled 'All', 'Pin', 'Via', 'Etch', 'Drc', 'Anti Et', and 'Bndry'. The 'Etch' column contains many checked boxes, while other columns have mostly unchecked boxes. The 'Sig3' row is highlighted in red, and the 'Vcc1' row is highlighted in green.

Subclasses	All	Pin	Via	Etch	Drc	Anti Et	Bndry
All	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Top	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Gnd	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sig1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sig2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Vcc	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Vcc1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sig3	<input checked="" type="checkbox"/>						
Sig4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Gnd1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Bottom	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

There may be colors assigned to subclasses suitable for re-use on other subclasses. Similar versions of the color may exist in the color palette, so to source the exact color, hover over the color assigned to a subclass, then right-click and choose *Select Color*. This outlines the color used in the palette, even toggling between the primary and secondary palettes if necessary.

Figure 3.3: Sourcing Colors for Reuse

Hover over this color box, right-click and choose *Select Color*

Subclasses	All	Pin	Via	Etch	Drc	Plan	Anti
All							
Top							
Gnd							
In1							
In2							
Gnd1							
Bottom							
Soldermask_Top							

This color box becomes active



Controlling Ratsnest Colors

To differentiate the display of ratsnests, a side-centric coloring scheme is available using the following options in the *Display* folder.

- *Rats top-top*: Specifies the color of ratsnest lines that connect top-side only components (start-end pin on top).
- *Rats top-bottom*: Specifies the color of ratsnest lines (one pin on top, other on bottom).
- *Rats bottom-bottom*: Specifies the color of ratsnest lines that connect bottom-side only components (start-end pin on bottom).

Controlling the Visibility of Individual Elements with Shadow Mode

To highlight specific elements in a design without affecting the visibility settings of that object's entire subclass, use the Shadow mode feature in the *Display* folder. Shadow Mode is used with the `hilight` and `dehilight` commands, as well as various interactive commands. When you enable Shadow Mode, the following occurs:

- The *Brightness* setting slide bar moves to its last applied percentage of brightness. The initial default percentage setting is 50%.
- The colors in the *Color* section dim to the chosen percentage of brightness in the slide bar. This allows you to "preview" how the colors in the design display if you click *Apply* or *OK*.
- *Dim active layer* lets you dim the active layer of a design. Dimming the active layer if it contains a large number of elements displayed normally (non-highlighted) can increase the effectiveness of Shadow Mode. You can dim the active layer with the check box in the *Color* dialog box or in the *Options* tab when shadow mode is turned on.
- The design elements of the current active drawing dim to the percentage of brightness set in the slide bar.

Shadow Mode Display Options

With Shadow Mode active, design elements display in the following ways:

- Normal. elements on the active layer of a design remain unaffected by Shadow Mode unless you select the *Dim active layer* in the *Options* tab.
- Highlighted, either permanently by way of the `hilight` command, or temporarily when you run an interactive command. In this state, elements are unaffected by Shadow Mode. Elements affected or added by a current interactive command are temporarily highlighted while the command is active. For example, if you run `add connect` with Shadow Mode on, the elements highlighted include:
 - Interconnecting pins
 - Existing etch/conductor being tied into
 - Connect lines, vias, and DRCs

When you complete the command, the added/affected elements are dimmed.

- Dim. The elements are unaffected by the conditions described above. The degree of dimming

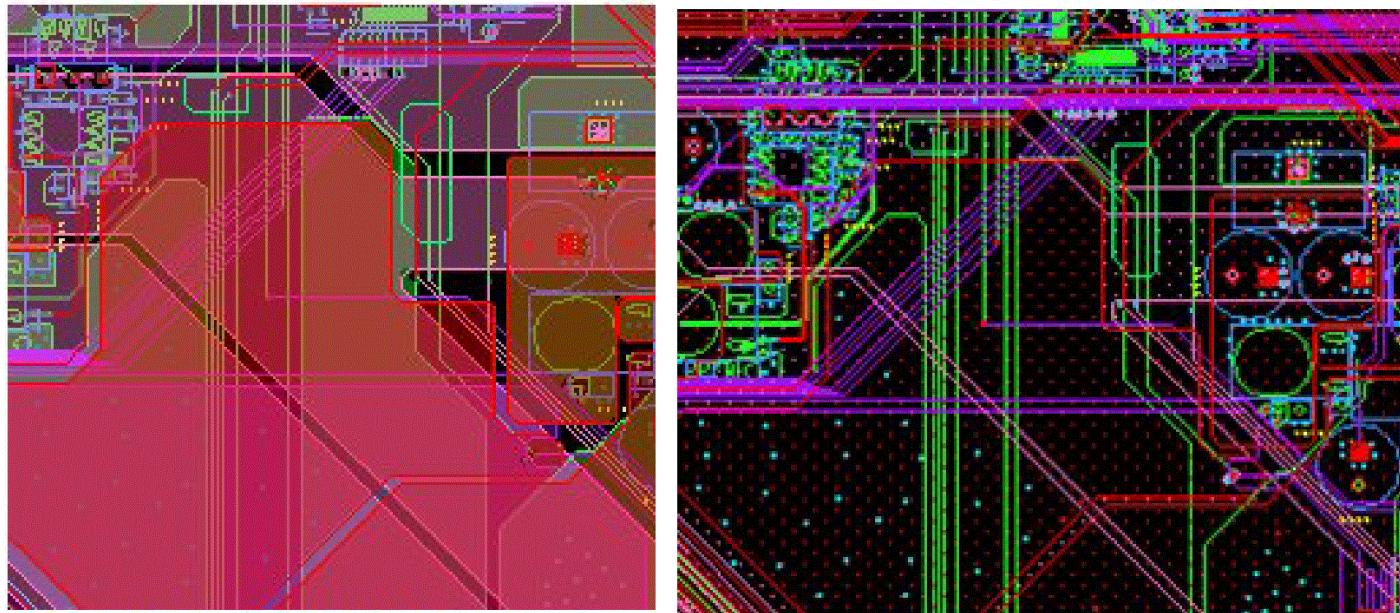
depends on the percentage of brightness set in the Color dialog box.

You can set global Shadow Mode parameters through the use of keyboard commands entered at the command window prompt, allowing you to assign function keys or toolbars to the dimming controls. For information on the syntax for setting Shadow Mode at the command prompt, see the [shadow](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Setting Graphics Transparency

OpenGL affords graphics transparency, which can be controlled at the global or shape level. A slide bar tailors the display from traditional solid to hollow fill, as Figure 3-4 shows. Transparent graphics allow more layers to display, including plane layers, that often block the graphics on other layers.

Figure 3.4: OpenGL Enabled (left) and OpenGL Disabled (right)

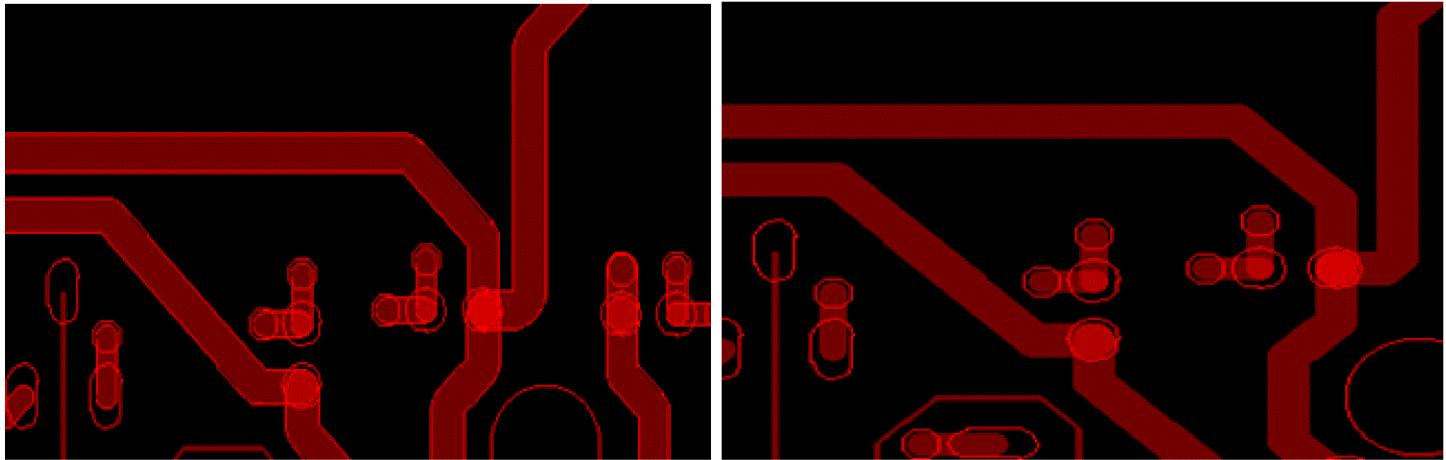


OpenGL is enabled by default. You can disable it using the environment variable *disable_opengl* in the User Preferences Editor dialog box.

Running Allegro with OpenGL requires a workstation with CPU board with at least 128 MB of memory and 128-bit bus interface. Only the 2D mode is supported. OpenGL requires higher-level graphics cards for best performance.

To display polyoutlines as Figure 3-5 shows, set the environment variable *draw_etch_outline* in the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* ([enved](#) command).

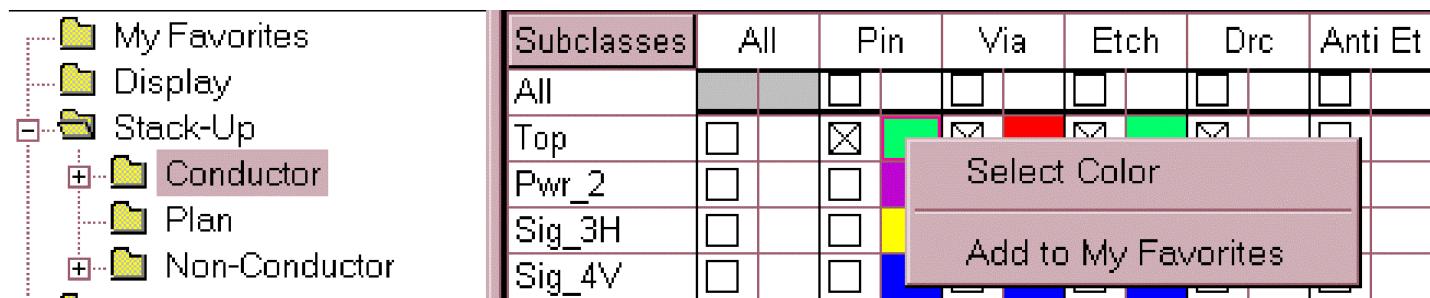
Figure 3.5: Polyoutline (l) and No Polyoutline (r)



Creating My Favorites' Folder

Use the *My Favorites'* folder to store frequently accessed subclasses where either the visibility or color changes often. Hover your cursor over the color box associated with a subclass, right-click and choose *Add to My Favorites*. The subclasses are copied, rather than moved, to the *My Favorites'* folder.

Figure 3.6: My Favorites



Remove a subclass from the *My Favorites'* folder by hovering your cursor over the color box, right-clicking, and choosing *Remove from My Favorites*.

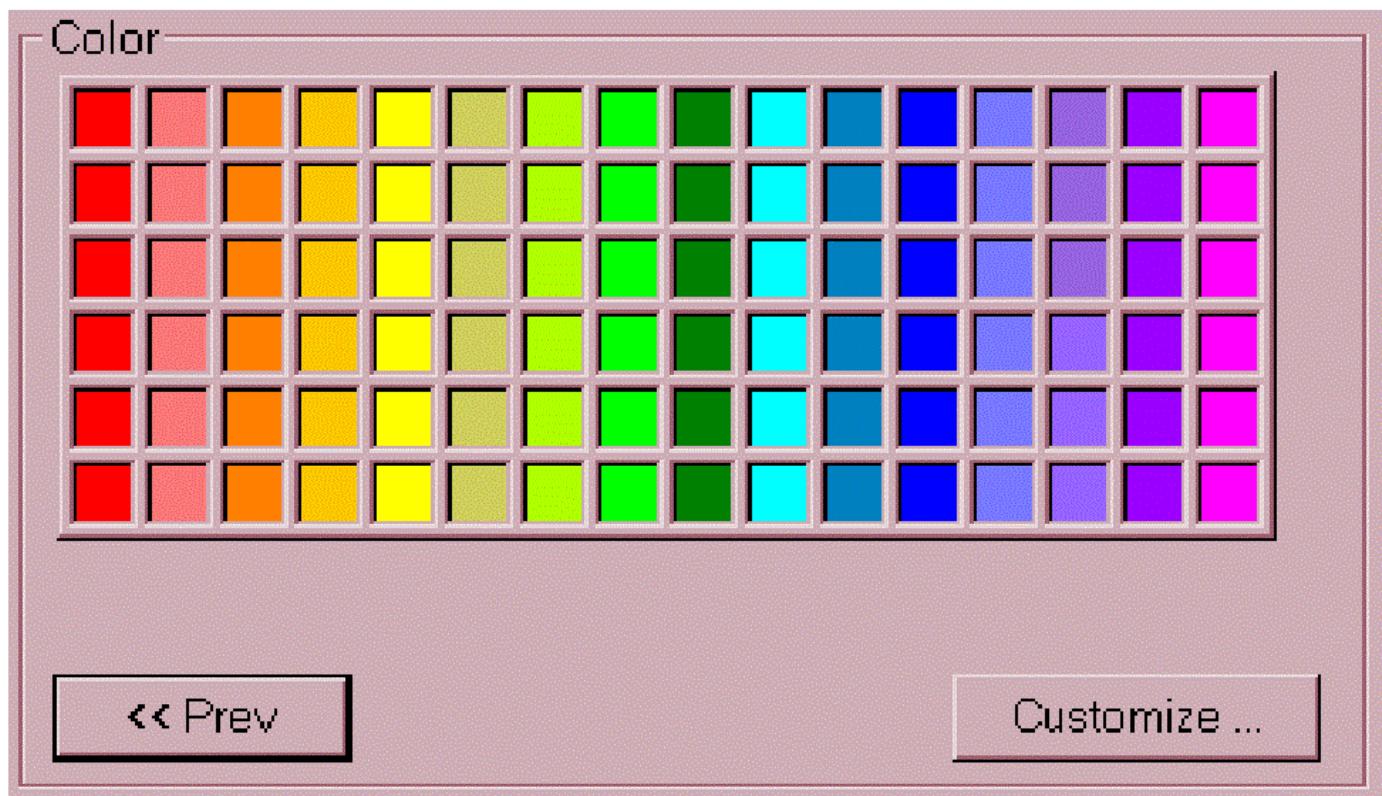
Saving and Reusing Color Palettes

When a design initially opens, the default color palette displays (Figure 3-7), which comprises an array of 16 x 6 colors. The first column comprises popular colors typically used in designs. This palette can always be reloaded using *File – Load Default Cadence Color Palette*.

Figure 3.7: Default Cadence Color Palette



Clicking *Next* displays the secondary palette, used for customization of colors, in Figure 3-8 :
Figure 3.8: Secondary Color Palette



A color palette may be customized and saved to an external .col file using *File – Save Color Palette*. You can then apply a unique color palette to other designs using *File – Load Color Palette*.

Customizing Design Colors

You can customize shades and hues of any color with the *Customize* button, which displays a Color dialog box shown below in Figure 3-9. After moving the control on the vertical sliding bar for luminosity away from the extremes of white or black, you can move the crosshair around the spectrum. All the fields in the dialog box reflect the correct number for the color in the crosshair. You can also type values in the fields to choose a color, click *Add to Custom Colors*, and then *OK*.

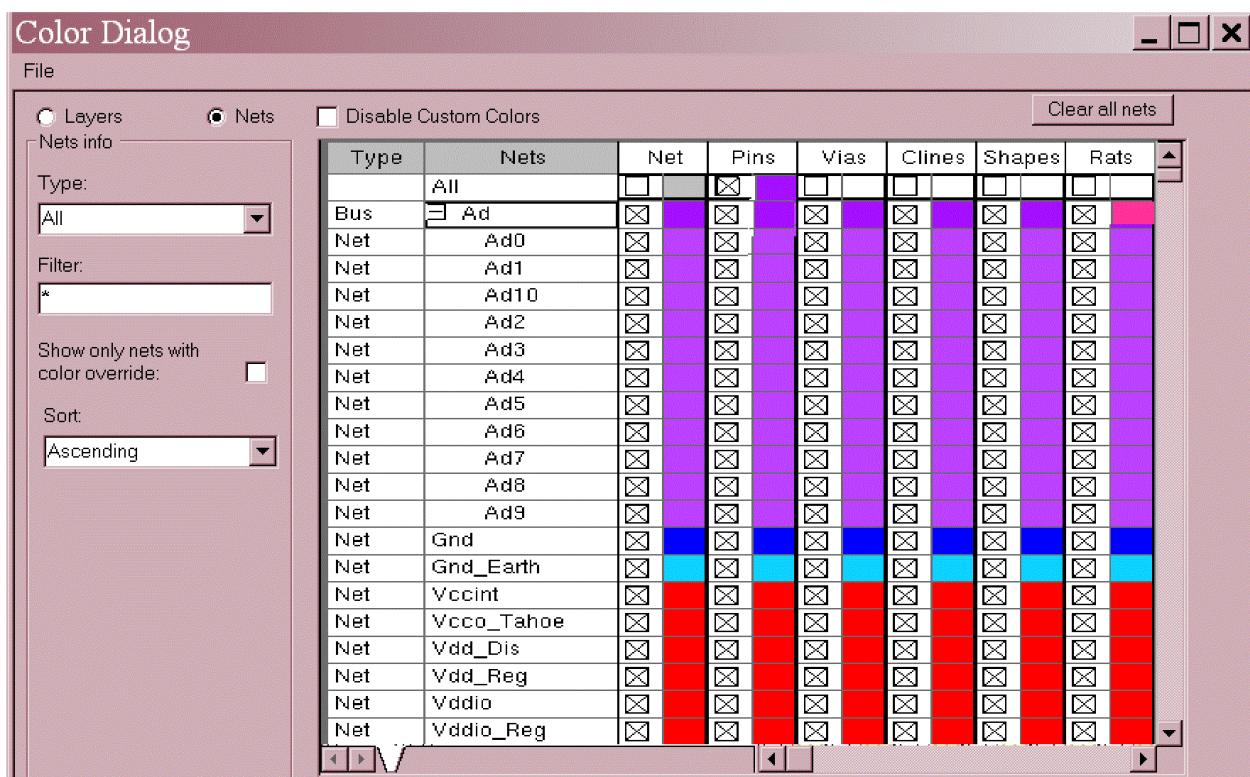
Figure 3.9: Color Customization dialog box



Using the Nets Grid

Database elements may be displayed using either the class/subclass color or a single color assigned to an element, also known as a custom color. To assign a custom color to an entire net or to its pins, vias, clines, shapes, or rats, you use the Nets grid. Assigning a custom color automatically enables the custom color state for that element as well, meaning that the custom color displays in the design canvas. You can also define how the custom color displays the element using a combination of states—none, highlight or custom color state, or highlight plus custom color state—all of which may be set independently.

Figure 3.10: Nets Grid



Highlighting is set or unset by right clicking and choosing *Set Highlight State* or *Clear Highlight State*, respectively.

To prevent custom color from displaying in both the Nets tree and the design, right click and choose *Clear Custom Color*. (A color box without a color assigned to it has no custom color state.) These custom color and highlighting states affect the display of the element as follows:

Element has custom color?	Highlight state?	Custom color state?	Element displays using...
Yes	No	No	Class/subclass color
Yes	Yes	No	Highlighted
Yes	Yes	Yes	Highlighted
Yes	Yes	No	Custom color, no highlighting
No	Yes	No	Highlighted with temp highlight color
No	No	No	Class/subclass color

Net Color Inheritance

Net elements can inherit highlighting and custom color from their parent. Inheritance can apply to specific elements as well. When a custom color is assigned to pins of an Xnet, for instance, all the pins of nets that belong to that Xnet inherit the custom color. A custom color or state of specific objects may also be overridden. For example, pins of a net can be assigned a color that differs from that of the net.

Saving Visibility Assigned to Classes and Subclasses

A color visibility view saves the visibility assigned to classes and subclasses as a collection of layer visibility settings that you can apply to subsequent designs using the *Views* field on the *Visibility* window pane. You save the settings in a file with a *.color* extension, stored in the directory specified by the *viewpath* variable, accessible in the *Paths-Config* folder in the *Categories* section of the *User Preferences* dialog box. A color view also displays film record visibility settings stored in the current design.

Color views (*.color* files) display in the *Views* field as *File: <name>*. Film record names display there as *Film: <name>*, unless you suppress the film record names from the list of color views in the *Visibility* window of the control panel. Suppress these names by selecting the *color_nofilmrecord* environment variable in the [The User Preferences Editor](#). Restart the layout editor for changes to the variable's value to take effect.

You can do the following tasks, all of which are described in the *Allegro PCB and Package*

Physical Layout Command Reference:

- Create or change a color visibility view, using *View – Color View Save* ([colorview create](#) command)
- Delete a color visibility view, described under *View – Color View Save* ([colorview create](#) command)
- Load a color visibility view, using the [colorview load](#) command
- Apply the previous color visibility view, using *View – Color View Restore Last* ([colorview restore](#) command)

APD+: Highlighting Sets of Pins

Highlighting objects in a package under design allows for quicker recognition of objects and, as a result, easier, more efficient design. The current IC Packaging tools provide the capability to permanently highlight nets, functions, symbols, or pins. Because functions are not widely used inside package designs, they rarely provide additional capability over symbol-based highlighting. Also, the current pin highlighting must be done based on window picks or by using the *Find By Name* advanced filtering techniques.

In Release 15.7, this base highlighting was extended with the `rats by layer` command, which allowed you to change the permanent highlight color of nets, and change the visibility of ratsnests for nets, based on the primary routing layer assignments.

With this release, you can supplement the current capabilities with enhanced pin highlighting, making such operations as pin swapping to optimize routability easier. With this feature, you can quickly highlight sets of pins in the design based on characteristics which otherwise would require you to make individual selections in the design. These include pin use, swap code, or padstack

For additional information on highlighting sets of pins, see the [advanced highlight](#) command.

Plotting a Design

The method by which the layout editor plots a design to a plotter or printer differs according to which platform you are on (UNIX or Windows) and which tools you run.

- The layout editors on Windows use Windows Print Manager for controlling printing operations. For information on installing a driver that supports a printer or plotter, consult the Microsoft Windows documentation.

- The layout editor on UNIX uses the `allegro_plot` program, which is based upon the Cadence corporate plotting package, *plotServ*.

Windows does not support `allegro_plot`. If you create an Intermediate Plot (IPF) file, which is a representation of a the tool database, you can copy it to a UNIX workstation that runs `allegro_plot` or to third-party plotting software.

- On either platform, the tool lets you import IPF files or create them for export using the [load plot](#) and [create plot](#) commands, which are detailed in the *Allegro PCB and Package Physical Layout Command Reference*.

See the *Preparing Manufacturing Data* user guide in your documentation set.

Working with Text

You can add, edit, and delete text in a drawing. Text can provide additional information about the design or it can be included as labels that are attached to graphic elements. This section describes:

- Defining text characteristics
- Adding text to drawings
- Editing text in drawings

Defining Text Characteristics

You can define the size and spacing characteristics of text that appears in the drawing. You can assign text parameters to up to 16 text blocks, which makes it easy to specify the appearance of text that you subsequently add to a design. You specify the text parameters as you add the text or label.

For procedures on defining text parameters, see the *Text* tab of the Design Parameter Editor. Use *Setup – Design Parameters* ([prmed](#) command) to access the Design Parameter Editor or right-click in the preselect use model and choose Design Parameters from the pop- up menu.

Adding Text to Drawings

You can use text in drawings as simple notes and as logical labels of elements. Labels include reference designators, device type, value, tolerance, and user part number.

 **Add – Text** ([add text](#) command) does not let you enter an exclamation point (!) in a design database, since `extracta` uses that character as a field delimiter. Be aware of the possible consequences of this condition if you read into the database a file that contains an exclamation point.

Some label commands require not only the data for text location and content, but also the identity of the element to be labeled, such as labeling placement room areas in the layout.

Use the `add text` command to annotate design elements. Use *Layout – Labels* menu selection (in Symbol mode) to add text labels (Ref Des, Device, Value, Tolerance, User Part Number) to symbols.

For procedures on adding text to a design, see *Add – Text* ([add text](#) command) in the *Allegro PCB and Package Physical Layout Command Reference*.

Editing Existing Text or Labels

You can edit text in a drawing. If the text is a reference designator label, editing the text changes the reference designator in the database. This can have other side effects, as explained in this section. You cannot edit a device type label in a drawing, because it redefines the logical structure of the component.

In general, when you edit text, the tool:

- Highlights the text and displays the text cursor on the first character location of the text string.
- Replaces the existing text.
- Lets you select another text string for editing.

 You cannot enter an exclamation point (!) in the layout database, since `a_extract` uses that character as a field delimiter. If an exclamation point is part of existing text that you are editing from an older version of the tool, be aware that `edit text` cannot replace that character if removed.

For procedures on editing text, see *Edit – Text* ([text edit](#) command) in the *Allegro PCB and Package Physical Layout Command Reference*.

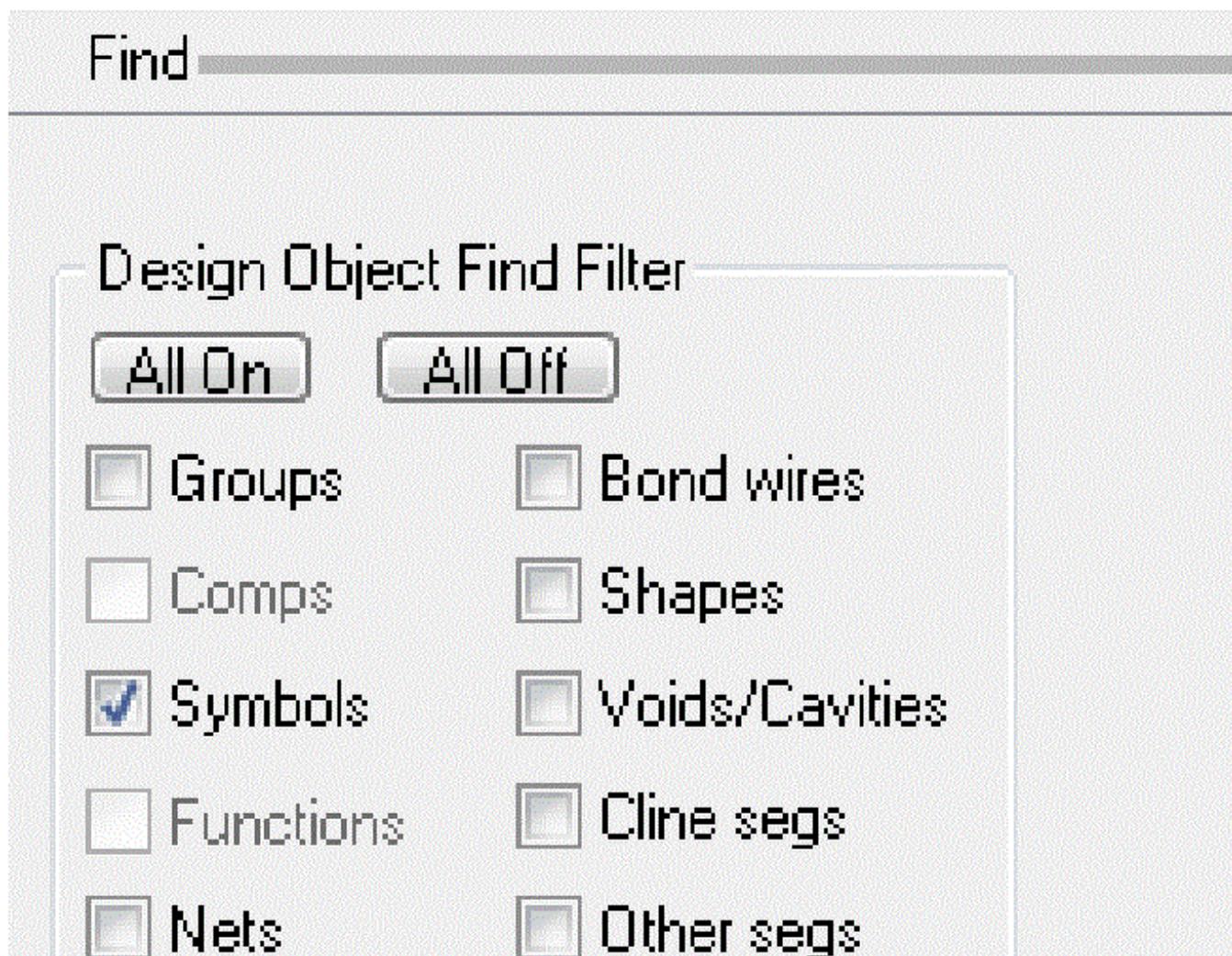
Finding Design Elements

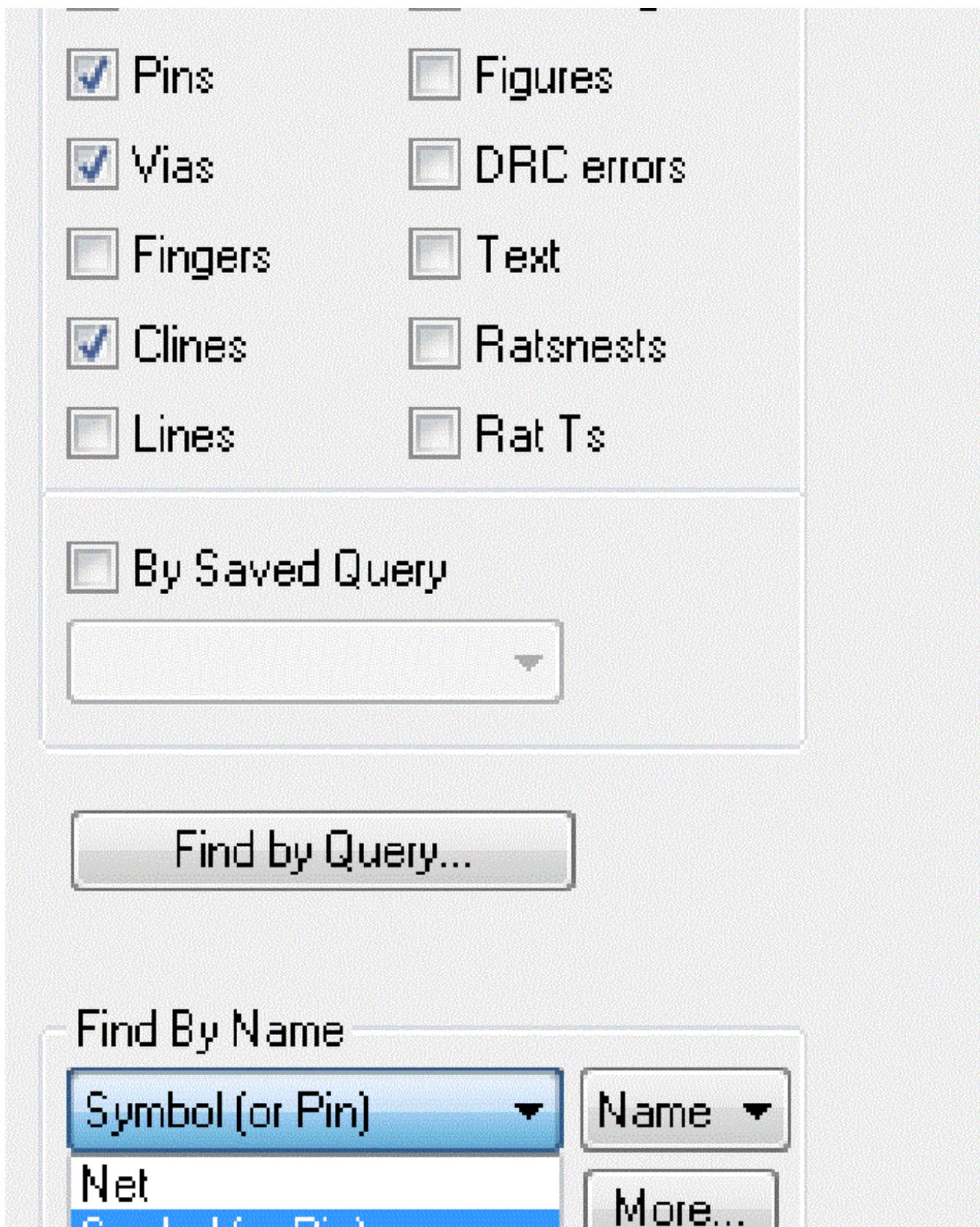
The Find Filter Window Pane

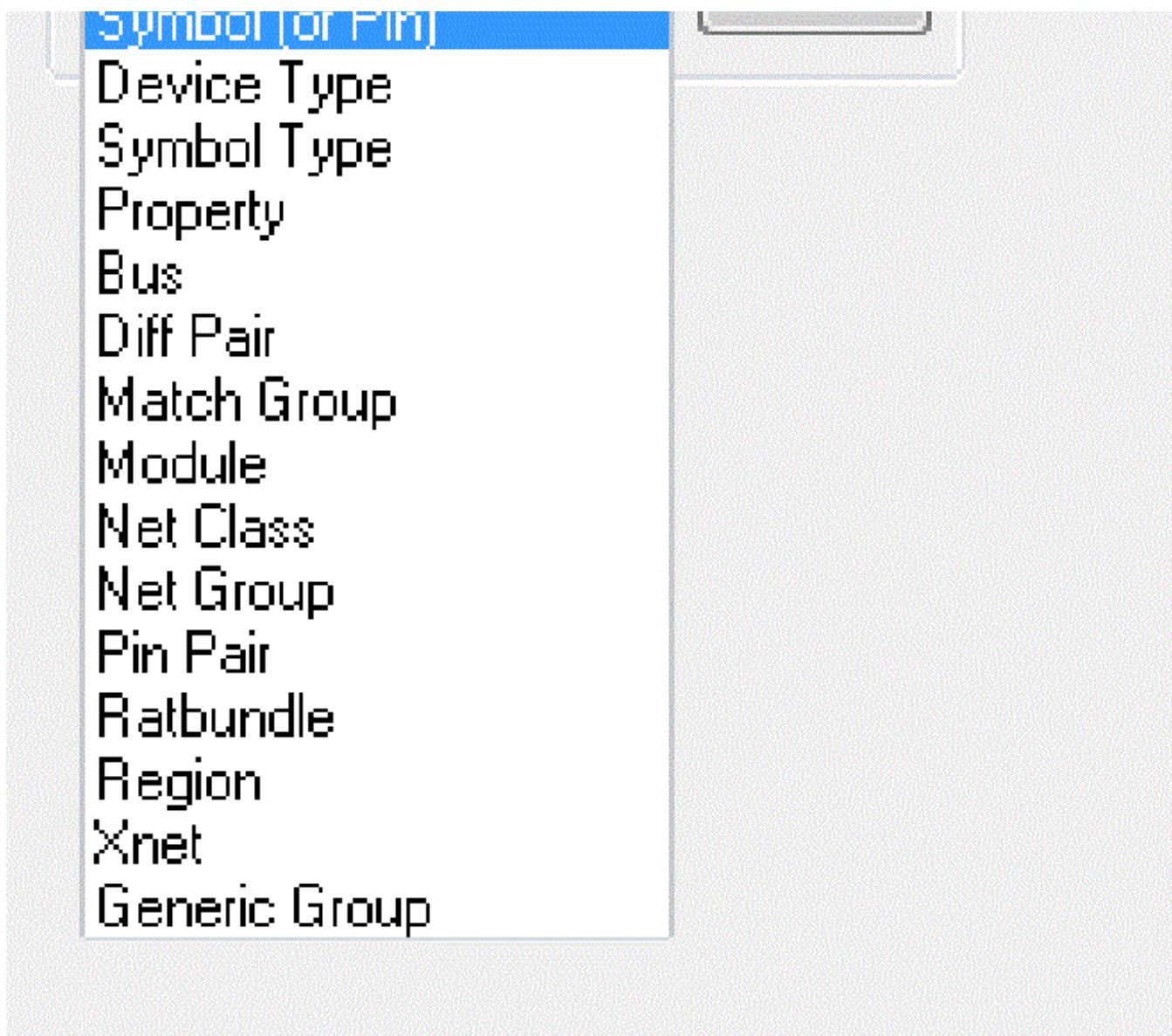
The *Find Filter* lets you specify design elements the active command affects. When you run an interactive command, such as *Edit – Move* ([move](#) command), the *Find Filter* displays the elements the command requires.

In pre-select use model, to refine your selection set and confine your work to a particular element type, such as all nets, you can also right-click and choose the *Superfilter* temporarily to disable the *Find Filter*. When you are using Superfilter, an icon appears in the lower right corner of the status bar.

Figure 3.11: Find Filter Window Pane







In menu-driven editing mode, the elements in the Find filter available for the active command are in bold text and have their check boxes chosen. The elements available for selection depend on the active command.

You can select or deselect any elements by clicking the check box on or off, or you can select/deselect all the elements with the *All On/All Off* buttons.

If you try to find an invalid element type, the layout editor displays the following message:

<element types> are not selectable at this time.

Name Function Failed.

Determining the Element Selection Hierarchy

The layout database maintains a hierarchy of elements to simplify the selection process. When you choose an element, the tool chooses the highest level element that is associated with that selection. If you disable the higher level elements, such as connect lines or nets, the tool chooses lower level elements, such as line segments.

For example, a pin can be part of a function, net, symbol, component, or group. When determining the proper element to highlight, the tool uses the following hierarchy:

- Groups
- Components
- Symbols
- Functions
- Nets
- Pins

Two primary methods allow you to locate design elements in the tool: *Display – Element* ([show element](#) command) and *Display – Property* ([show property](#) command). Both let you find elements by name or property, but do so in different ways.

Using Show Element

You can use *Display – Element* ([show element](#) command) with the Find filter to locate and identify design elements by property, name or in a list file. You can further refine a find operation by entering a value for the element you want to find. You perform these operations using the Find By Name or Property dialog box.

Finding Elements by Name or Property

With *Display – Element* ([show element](#) command) active, click More in the Find filter to display the Find by Name or Property dialog box, which lists all available object types for chosen elements.

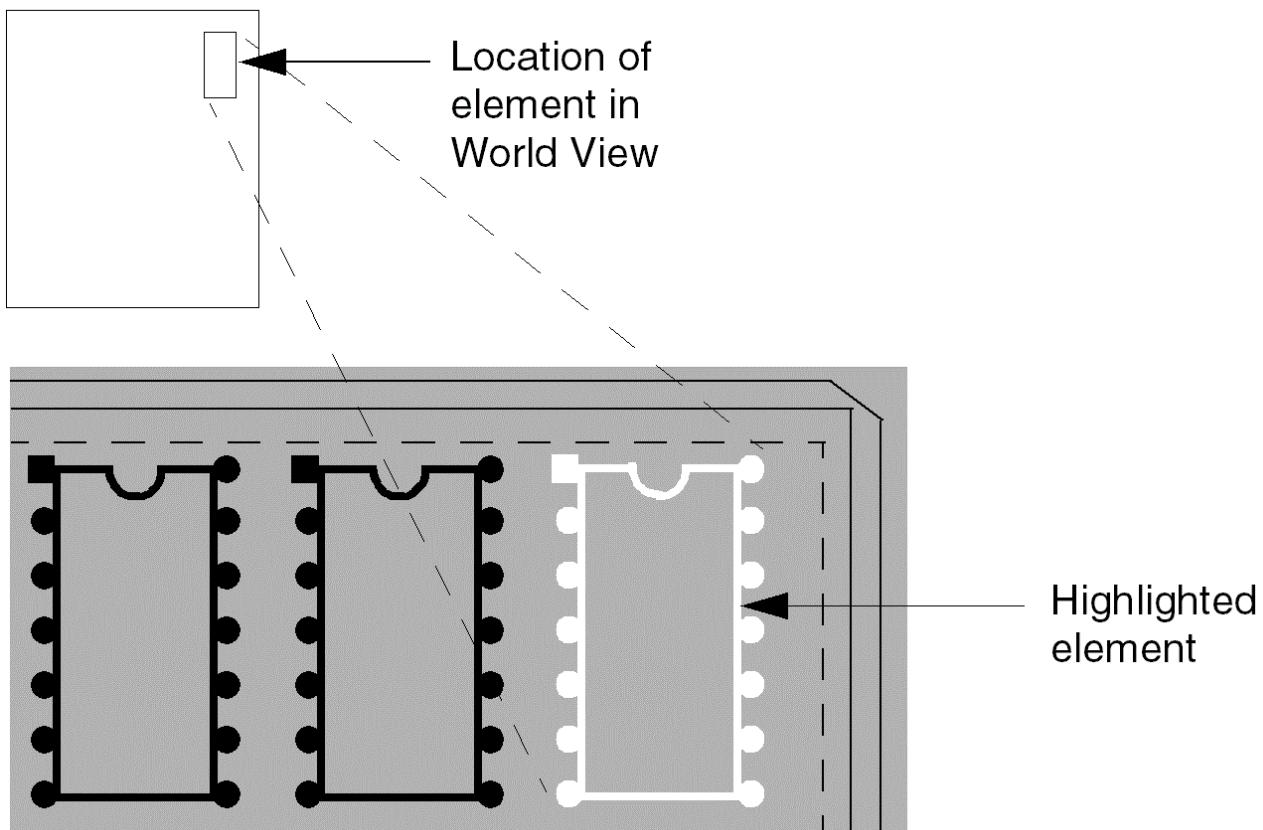
Depending on the object type you select, the Find By Name or Property dialog box allows you to identify an element that you want to find by listing those elements by object type. You can then choose individual items and then by clicking Apply:

- Display the Show Element dialog box on the element(s).

- Display the location of the element(s) in the World View area of the UI.
Highlight the chosen element(s) in the design area

If you know the name of the element that you want to locate (such as U13), you can find it by entering its designation in the Name Filter field and selecting the appropriate object type from the menu.

Figure 3.12: Example Result of Find by Name



Using Show Property

Unlike *Display – Element* ([show element](#) command), *Display – Property* ([show property](#) command) is not used with the Find Filter, though it can help you locate elements in a design. When you run the command, the Show Property dialog box appears.

By selecting a property (sorted by property or element) and pressing the appropriate Show button, you can display a definition of the property or its value relative to the object to which it is attached. The Name and Value fields let you qualify an element further. When you enter a name or value, the tool searches only for those elements that match both the Name and Value that you entered, and

that are valid for the active command.

Using Find by Property from the Console Window Prompt

You can also use the console window prompt to find elements by property. The Find Filter must be activated with elements that allow property assignments.

To use *Find by Property* from the console window prompt:

- At the console window prompt, type

```
find property name <property value>
```

All elements are chosen for the active command that have the defined property name and value.

You can use wildcard characters for both the property name and value. The property name field is not case-sensitive.

Finding by Name from the Command Window Prompt

You can also use the prompt in the command window to find elements by name. You must activate the Find Filter with elements that appear in the design.

When you use the command line at the console window prompt, you can enter character strings, including the element type plus a name or list file, and wildcard characters. Character strings are not case-sensitive.

Table 3-3 lists keywords, the way in which the tool matches that keyword, and an example of each keyword type.

Find by Name Commands		
Keyword	Match	Sample Value
Net	Net that matches name	data1
Symbol(or Pin)	Symbol instance that matches refdes	U34
	Symbol pin that matches refdes.pin	U34.1

Device Type	Component or symbol instances that match device type—components are chosen if the command allows; otherwise, symbols are chosen	74LS74
Symbol Type	Symbol instances that match symbol name	dip14
Property	Property that matches name	NET_SHORT
Bus	Bus that matches name	CON_HDD_BUS
Diff Pair	Diff Pair that matches name	DP1
Match Group	Match Group that matches name	MG1
Module	Module that matches name	MD1
Net Class	Net Class that matches name	CLS_1
Net Group	Net Group that matches name	NG1
Pin Pair	Pin Pair that matches name	U40.3 :U49.4
Ratbundle	Ratbundle that matches name	RAT1
Region	Region that matches name	REGION_1
Xnet	Xnet that matches name	XNET_L1
Generic Group	Generic Group that matches name	GR1

You must enter the keyword exactly as it appears in the drop-down list in the Find Filter. In other words, type `comp` or `symp` instead of component or symbol. If you enter multiple names, put a space between the element names. If the element name contains a space, put quotation marks around it.

For example, the following command selects the nets MEM17, DATA4, and CLOCK for processing.

```
net mem17 data4 clock
```

Likewise, when you enter multiple lists, you must put a space between each list file. For example, the following command selects all components in the files `U.1st` and `R.1st` for processing.

```
list comp U(.lst) R(.lst)
```

Using Wild Cards

The tool lets you use wild card characters when you try to find elements by name or by list. Table 3-4 lists the valid wild card characters.

Table 3.2: Valid Wild Card Characters

Wildcard	Match Description	Example
*	Any number of characters	name* = name1, name12, name ANY
?	Any single character	name? = name1, nameA name? ≠ name12, name ANY

Highlighting Chosen Elements

When you select elements by group or window, the tool lets you specify the temporary highlight color. Table 3-5 summarizes the way in which the tool highlights element types:

How the Layout Editor Highlights Element Types	
Element Type	Highlight
Net	All vias, connect lines, shapes, frects, ratsnests, and pins on a chosen net
Component Instance	Symbol instance linked to chosen component instance. Only placed components can be highlighted. Symbols highlighted by symbol and component instance appear the same on the display.
Ratsnest	Chosen ratsnest line
Function Instance	All pins for the chosen function instance

Finding Elements by Using the pick Commands

In addition to using the mouse to highlight elements in a drawing, you can use [pick](#) or [ipick](#) commands to enter x, y coordinates for the elements as described in the *Allegro PCB and Package Physical Layout Command Reference*.

Using Temporary Group Mode

When you run an interactive command in temporary group mode, you can identify elements by name, list, pick, window, or any combination of these until you click right and choose *Complete* from the pop-up menu. Temporary group mode is available only in menu-driven editing mode.

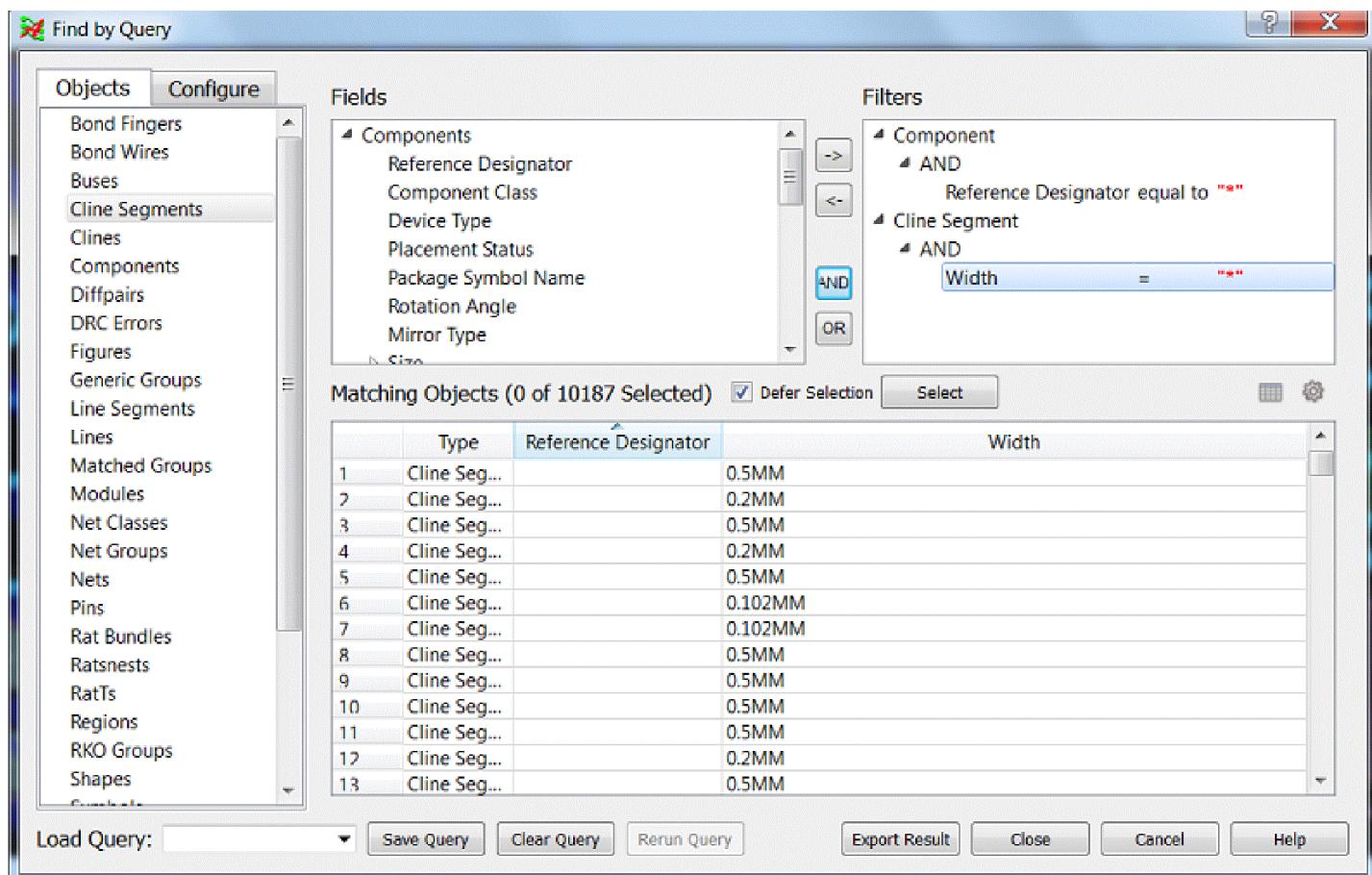
To deselect elements that you select in temporary group mode:

- Press *Ctrl* and click the mouse button.
If you are working with a congested board and multiple elements are chosen by a pick, the elements that you do not deselect go into the reject buffer.

Finding Objects by Query

You can use this method to issue a query to find the specific objects (nets, clines, shapes, voids, and so on) based on their attribute values in the database. The query returns all the objects that match a specified filter or criteria. You can define the criteria in the *Find By Query* dialog box available in the *Find* filter window pane.

Figure 3.13: Finding Objects by Query



The *Objects* tab displays list of different types of objects that can be modified in the *Configure* tab. You can add required objects and their attributes in the *Fields* for creating a query. The *Filters* section defines the query that will be used to extract the information.

The search results are displayed on the fly in the *Matching Objects* section. The *Configure Search Table* icon lets you identify the fields you want to see in the result table.

You have options to save query in a file (.qfnd), clear query, and load query. If inactive for some time, the *Rerun Query* button becomes active to refresh the results.

You can export the query in standard XML or CSV formats.

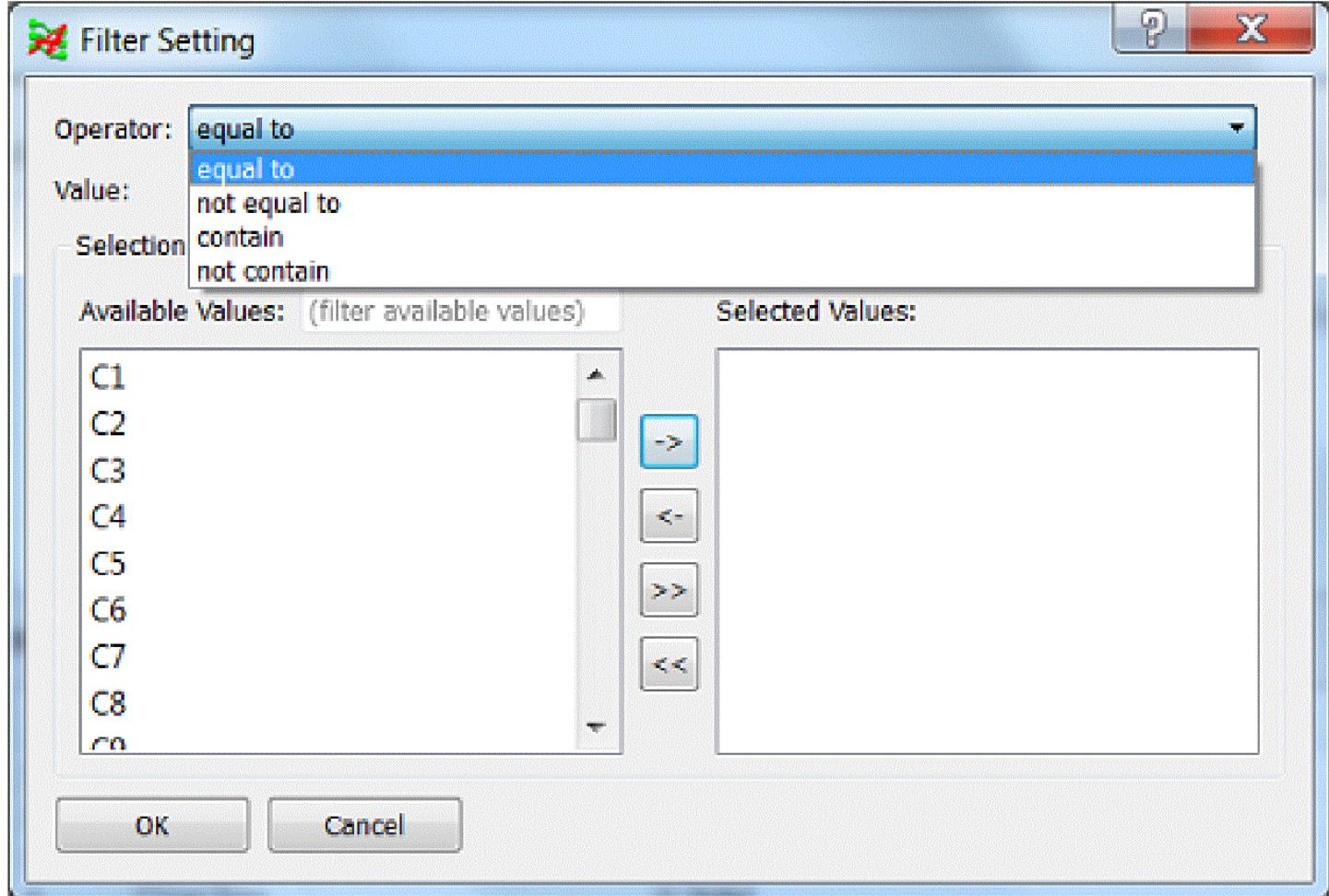
Defining a Query

Choose objects from the *Objects* tab and add them to the *Fields* section by double-clicking or by using drag and drop method. Move the required attributes to the *Filters* section. You can create query based on AND and OR logical conditions. You can specify multiple objects and their attributes for filtering.

In the *Filters* section, the attributes are added with null values representing by "/*". Selecting an

attribute opens *Filter Setting* dialog box where you can specify its value. You can either directly set values or choose from the list of available values that are present in the database. You can also specify a logical expression for determining the attribute value.

Figure 3.14: Filter Settings

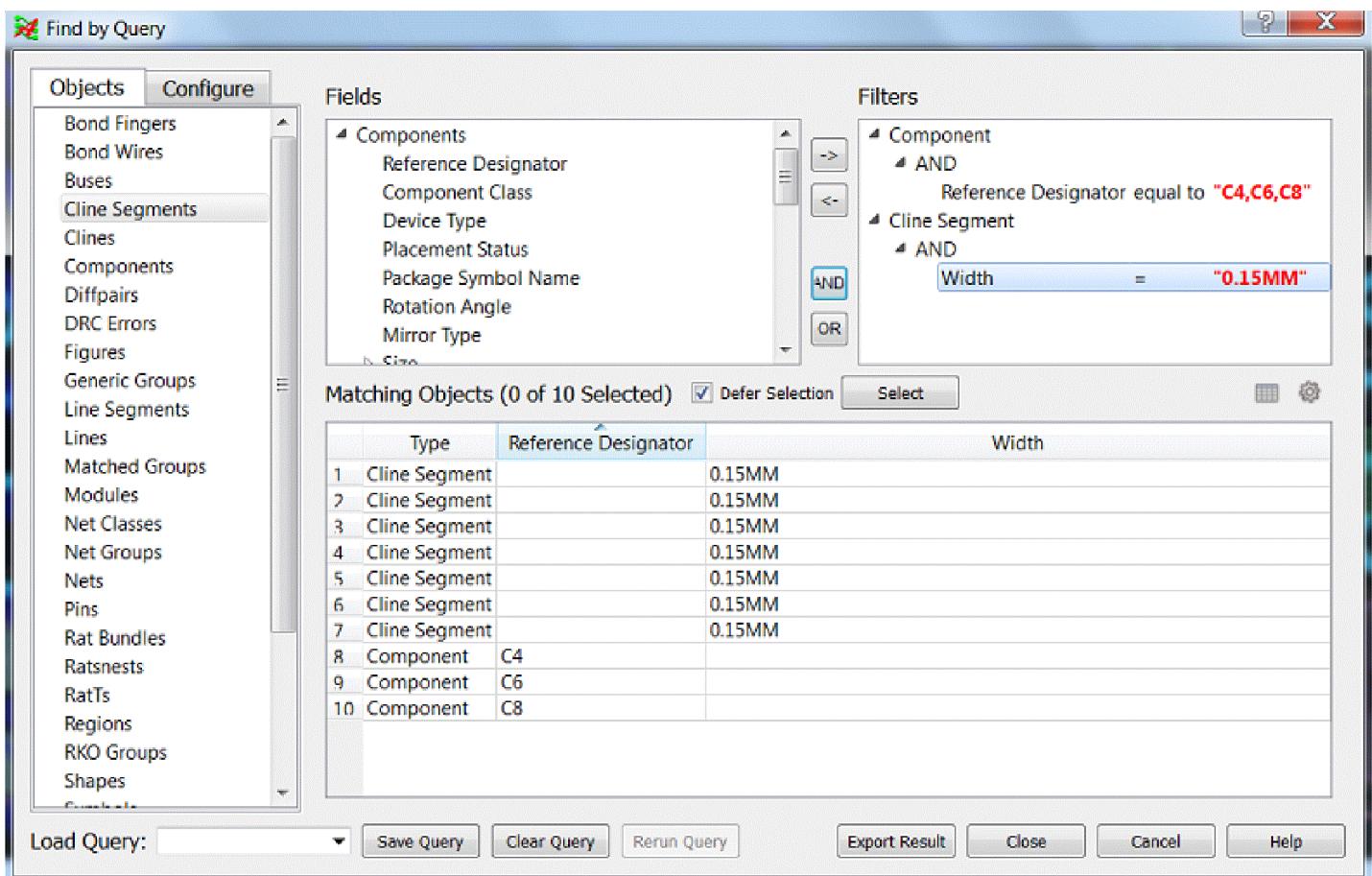


Viewing Matching Objects List

The search results are displayed on the fly in the *Matching Objects* table. The total number of matching results updates based on the changes you made in the *Filters* section.

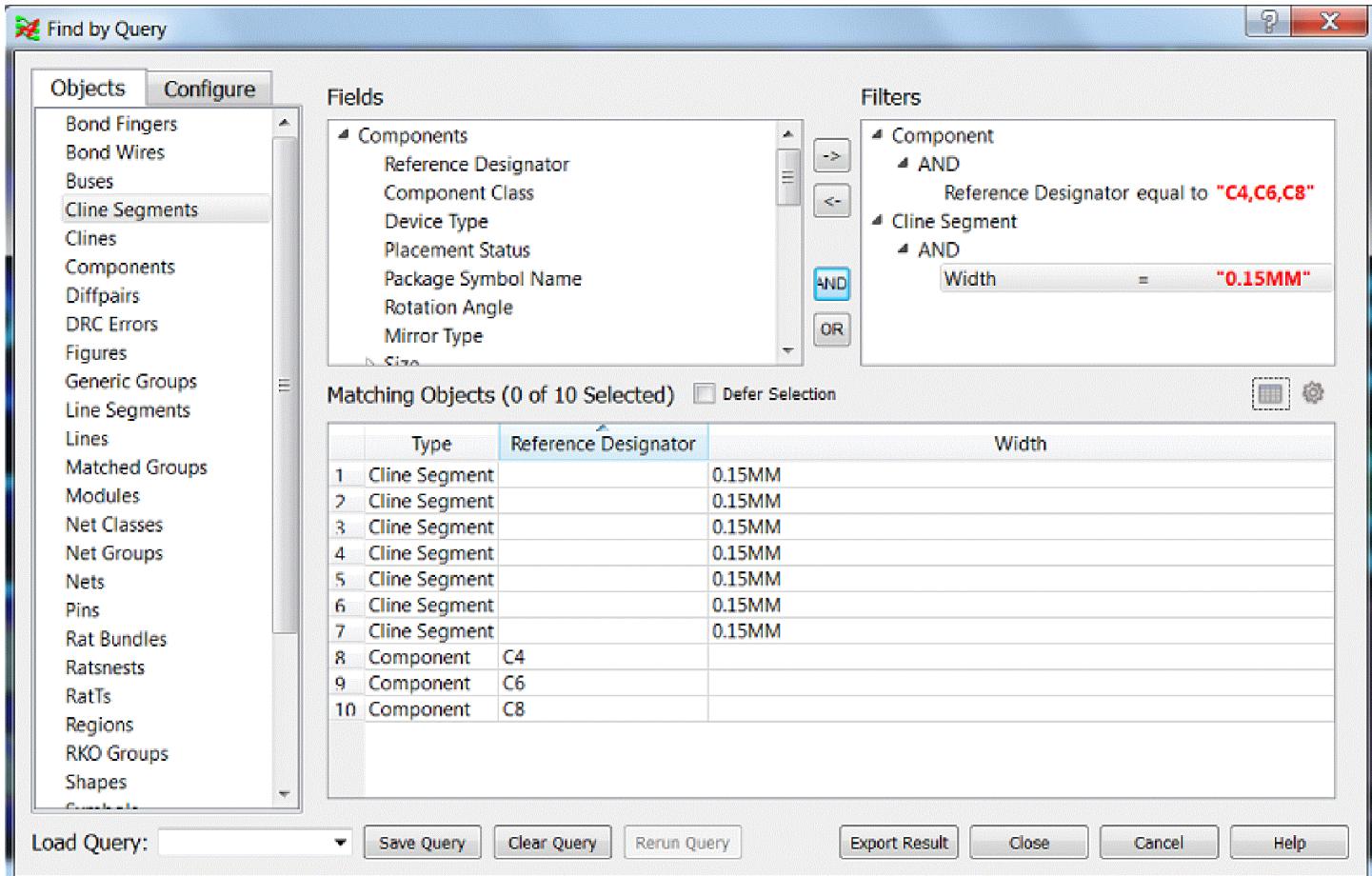
Figure 3.15: Matching Objects List

Getting Started with Physical Design
Using the Layout Editor--Finding Design Elements



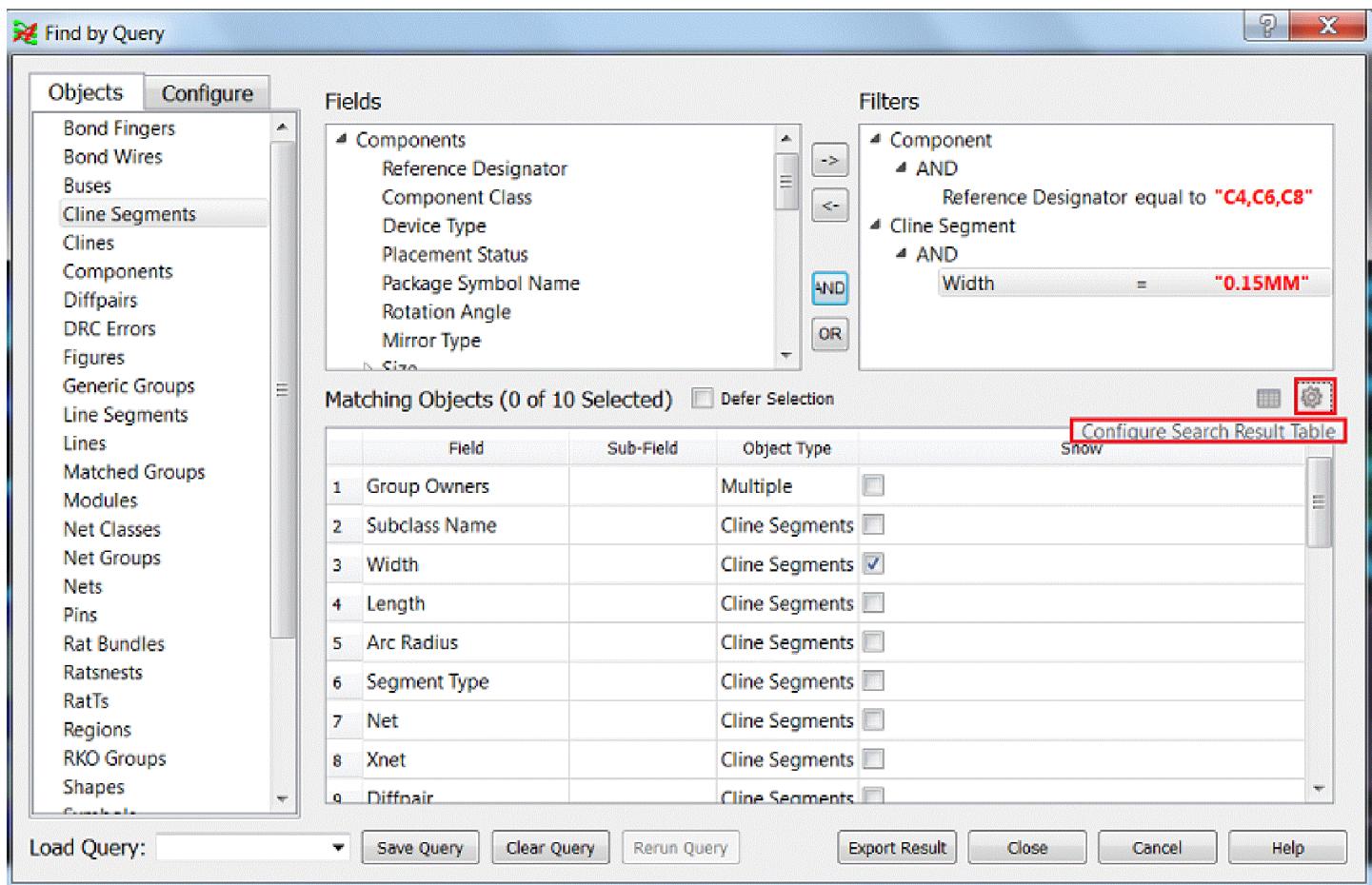
A default set of attributes are always displayed for every object type in the *Matching Objects* table.

Figure 3.16: Default Display of Matching Objects



The *Configure Search Table* icon lets you identify the fields you want to see in the result table. You can enable or disable the corresponding check boxes to show/hide them from the *Matching Objects* table.

Figure 3.17: Configuring Display of Matching Objects

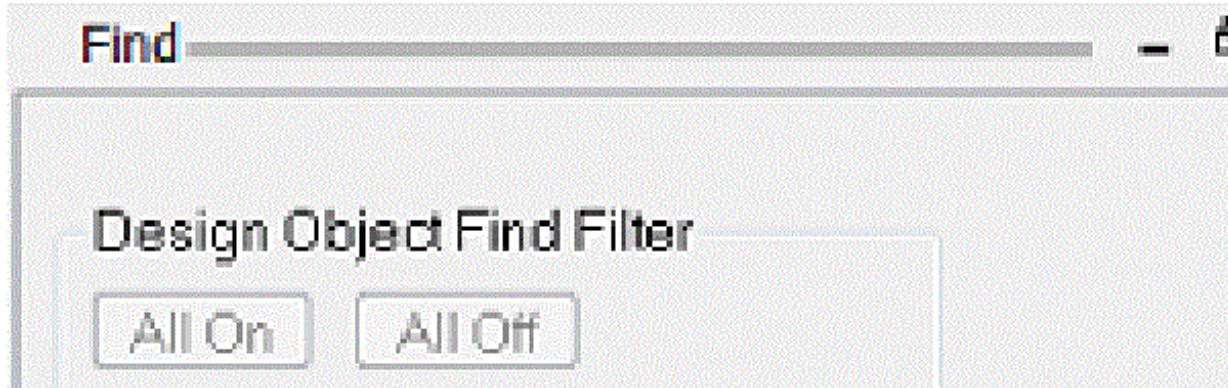


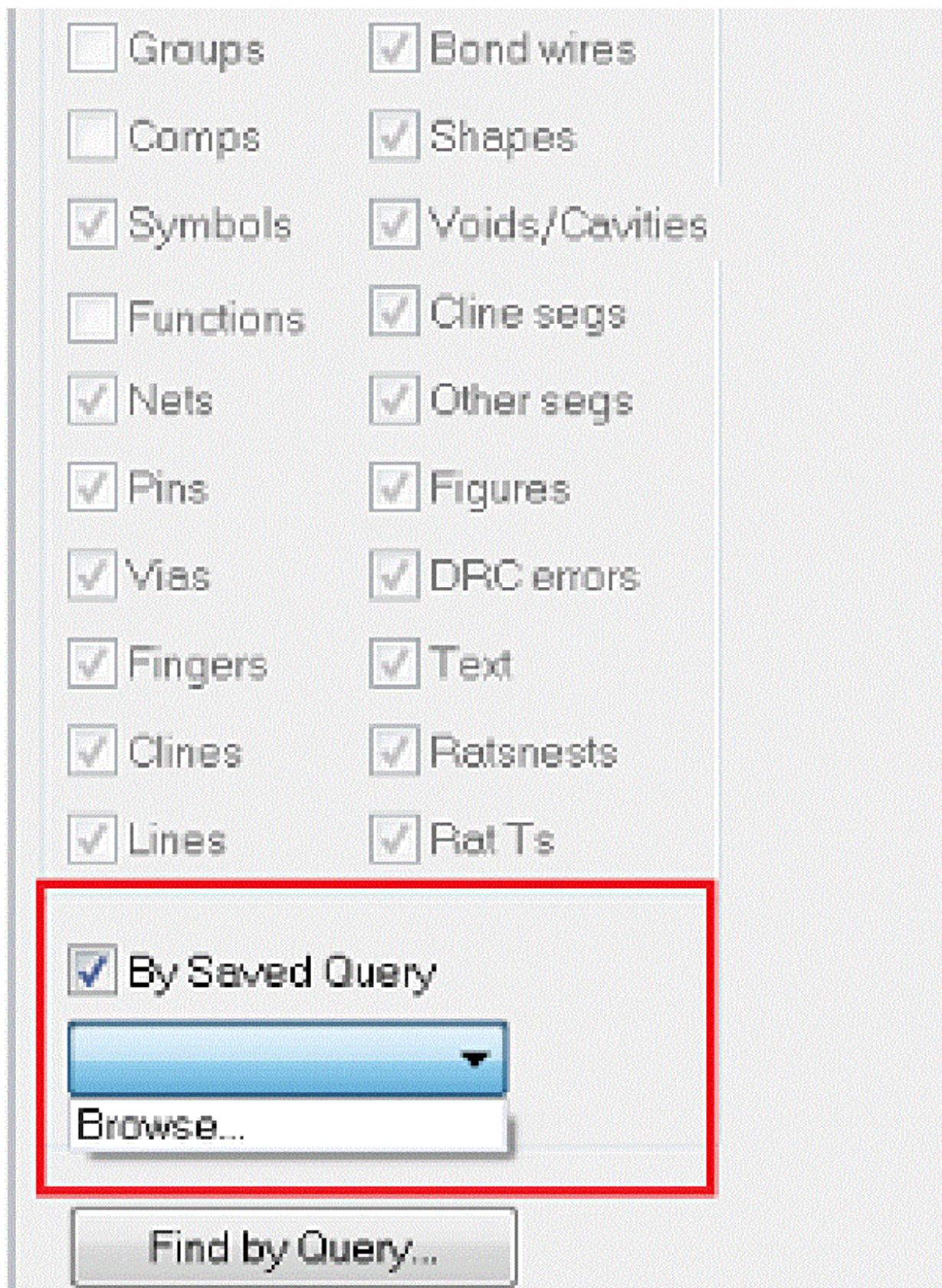
Saving Query

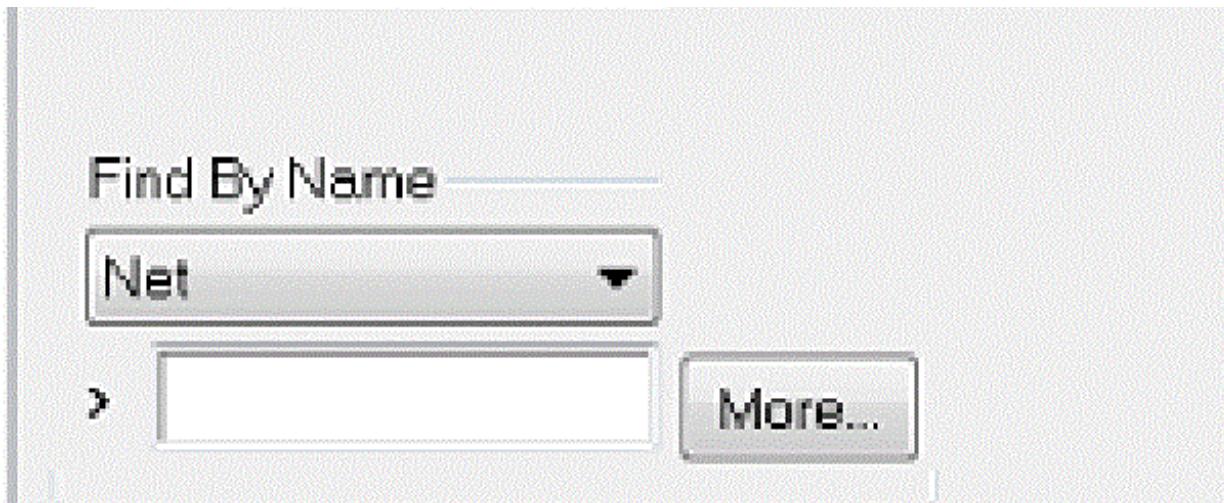
You have options to save query, clear query, and load query. If inactive for some time, the *Rerun Query* button becomes active to refresh the results. You can also export the query in an XML or CSV format.

The current query settings are saved as FNDQ(.qfnd) file. Once saved the query can be executed again by browsing it through the *By Saved Query* option in the *Find Filter*.

Figure 3.18: Running Saved Query







Displaying Matching Objects in the PCB Editor

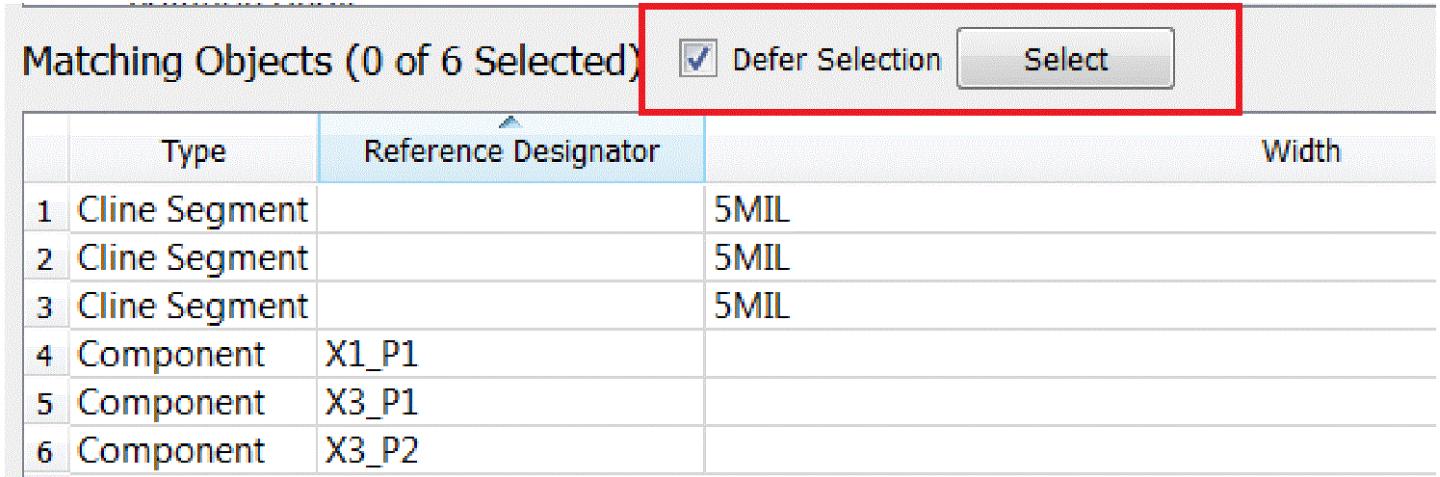
When you click the search result in the *Matching Objects* table, the command:

- Selects the objects to be acted upon by the active command; for example, `property edit`.

⚠ Different commands result in specific behaviors. For example, if you are running `property edit`, the *Edit Properties* and *Show Properties* dialog boxes are displayed; if you are running `move`, the selected objects are selected for editing.
- Displays the location of the objects in the *WorldView* area of the UI.
- Highlights the selected objects in the design area of the UI.

In the result table, select any row right-click and choose *Select All* to highlight all the objects at a time.

If required, you can delay the selection of objects in the design canvas by enabling *Defer Selection* checkbox. A *Select* button becomes available that holds the selection. The selected object is not processed until the button is clicked.



The screenshot shows a table titled "Matching Objects (0 of 6 Selected)". The table has three columns: "Type", "Reference Designator", and "Width". There are six rows of data. The first three rows are "Cline Segment" objects, each with a width of "5MIL". The last three rows are "Component" objects, with reference designators X1_P1, X3_P1, and X3_P2 respectively. At the top of the table, there are two buttons: "Defer Selection" with a checked checkbox and "Select". A red box highlights these two buttons.

Matching Objects (0 of 6 Selected)		
Type	Reference Designator	Width
1 Cline Segment		5MIL
2 Cline Segment		5MIL
3 Cline Segment		5MIL
4 Component	X1_P1	
5 Component	X3_P1	
6 Component	X3_P2	

Using this functionality you can choose multiple objects from the *Matching Objects* table and view them together or send them as an input to a command in a single step.

Finding Buses in Composer/Allegro Design Entry HDL or System Connectivity Manager and Allegro PCB Editor

When you draw a schematic in Composer, Allegro Design Entry HDL or System Connectivity Manager, you can identify groups of nets as buses. The Find Filter lets you use this bus identification to process nets that are members of the buses. In Composer and Allegro Design Entry HDL or System Connectivity Manager, each net in a bus has a bus name, followed by a number that is enclosed in angle brackets. This number specifies the bit position in the bus. For example, a four-bit data bus can consist of the nets DATA<0>, DATA<1>, DATA<2>, and DATA<3>.

Identifying Buses

When you choose *File – Import Logic* ([netin](#) command) and choose *Design entry HDL* from the Import Logic dialog box, each bus is assigned a **BUS_NAME** property and value that matches its net name. For example, in the bus described in the preceding section, each net receives a **BUS_NAME** property with DATA as the assigned value.

The net name assigned is the original bus name plus the associated number without the angle brackets. For example, the corresponding tool net names for the four-bit data bus are DATA0, DATA1, DATA2, and DATA3.

This association between the net name and the bus name lets you use the Find by Name function to identify the net and by using *Edit – Properties* ([property edit](#) command) to add the **BUS_NAME** property interactively.

Bus Selection Syntax

You can specify designated bus nets on the command line in the command console window or, if you choose *Nets* in the Find Filter, in the Name field.

To specify a group of nets in a bus:

- Enter the bus name and a bit subscript field using the following formats:

<bit>	Specifies a single bit of the bus. For example, DATA<3> defines net DATA3.
<bit1:bit2>	Specifies a subrange of bits. For example, DATA<3:1> defines nets DATA3, DATA2, and DATA1. (The order of this subrange does not matter; DATA<3:1> is the same as DATA<1:3>.)
<bit_list>	Specifies a list of bit subscript fields that can have either of the preceding formats. Separate each list with a comma. For example, DATA<1:3,7,10:12> defines bits 1, 2, 3, 7, 10, 11, and 12.

In each of these formats, angle brackets delimit the bit subscript field; the *bit* variable specifies a bit number and must be an integer greater than or equal to zero. If you leave the angle brackets empty, the tool chooses all nets of the bus. To choose bus members, the bus name must match the net name and bit number exactly.

The following command chooses the DATA1, DATA3, DATA4, DATA5, DATA6, and DATA7 nets for processing.

```
net data<1,3:7>
```

In addition, you can assign a BUS_NAME to nets that do not have a bit number in the name or that match the bus name, but that can be found by using the *busname<>* syntax. For example, if you assign the BUS_NAME property DATA to the DATA0, DATA1, DATA2, and DATA3 nets and enter the following command in the Name field, you select all the nets.

```
net data< >
```

Using Buses

The following menu selections/commands accept bus names:

- *Display – Highlight* ([hilight](#) command)
- *Display – Dehighlight* ([dehilight](#) command)
- *Display – Element* ([show element](#) command)

- *Display – Property* ([show property](#) command)
- *Edit – Change* ([change](#) command)
- *Edit – Delete* ([delete](#) command)

You can also use the select by bus name option to expedite the following operation:

- Highlighting the bus nets
- Assigning placement weights to a bus by defining the WEIGHT property on bus nets
- Routing buses before the other nets by setting the ROUTE_PRIORITY property on bus nets

Highlighting and Dehighlighting Design Elements

The layout editor lets you highlight and dehighlight eligible elements to accentuate and easily locate them in the design canvas with *Display – Highlight* ([highlight](#) command) and *Display – Dehighlight* ([dehighlight](#) command).

Another means of highlighting elements is *Display – Assign Color* ([assign color](#) command). You can quickly assign both custom color and highlighting to an element without requiring the use of the *Color* dialog box and *Display – Highlight*. Changing the color or highlighting with this command automatically updates the color and highlighting information in the *Nets* section of the *Color* dialog box as well.

These three commands also function in the pre-selection use model, in which you choose an eligible element first, then right-click and execute the command.

Automating Design Tasks with Scripts and Macros

If you find yourself repeating certain design tasks on a regular basis, you can create scripts and macros to automatically perform those tasks.

While you can use both scripts and macros across multiple drawings, scripts always start and end at the same coordinate, whereas a macro lets you start at a different coordinate each time you use the macro. Every action included in the macro takes place relative to the starting point.

Scripts are useful when performing repetitive tasks such as setting up fields in dialog boxes, adding elements to multiple databases at the same location, and duplicating drawings.

For information about procedures for using scripts, see *File – Script* ([script](#) command) in the *Allegro PCB and Package Physical Layout Command Reference*.

Using Environment Commands with Scripts

You can modify the behavior of script recording and replaying through the use of environment commands entered at the console window prompt.

For information on using environment commands in scripts, see the [ifvar](#) and [ifnvar](#) commands in the *Allegro PCB and Package Physical Layout Command Reference*.

Displaying Connectivity

The layout editor uses ratsnest lines to display the connectivity in a design. These lines show the logical connections between pins, lines, or vias that are on the same net.

For information on displaying ratsnest lines, see *Display – Show Rats – All* ([rats all](#) command), *Display – Show Rats – Components* ([rats component](#) command) or *Display – Show Rats – Nests* ([rats net](#) command) in the *Allegro PCB and Package Physical Layout Command Reference*.

Using Data Browsers

Data browsers are dialog boxes that present elements of the type required by the current command. You can select elements listed in a data browser, but you cannot delete, rename, or otherwise control the type of data displayed. Data browsers list all named elements in a design or within libraries outside the design, based on parameters that you set in the dialog box.

Displaying Quickview Information

Data browsers support quick views of the database elements that you select from the list in the dialog box. Quickview lets you see a graphic preview of a database or a selection of the properties that make up the database.

 Graphics are not available for padstacks; only text regarding the padstack name, type, units, accuracy, and geometry is available.

Supported databases include the following file types:

.brd	.bsm
.mcm	.dra

.osm	.mdd
.ssm	.psm
.fsm	.dfa
.dpf	

File browsers that open scripts, logs, and other text files do not support quickviews.

⚠ Older databases must be upreved to version 14.0 (or subsequent versions) with Qvupdate before you can display them in quickview.

By selecting one of the two quickview buttons, you can view different data associated with the selection:

- *Text*

The Text button displays text information, such as the information for a package symbol.

Name: SSOP28

Type: Allegro Symbol

Units: MILS

Accuracy: 2

Pins: 28

- *Preview*

The preview button displays a simple graphic of the database, the image of which depends on the type of database you are viewing.

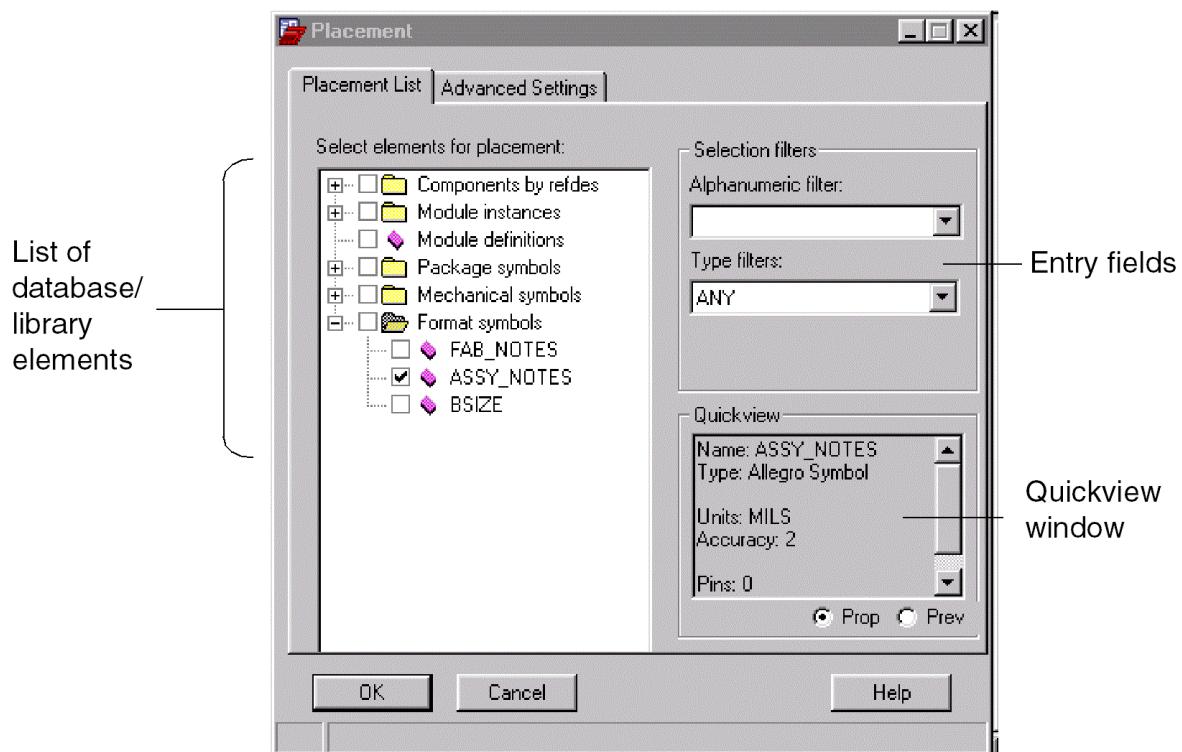
- Quickviews of .brd and .mdd databases display a board outline, package keepin, or a rectangle of the drawing extents, and a chosen set of the largest pin-count components in the database.
- Quickviews of symbols display a symbol outline and the number of pins on the symbol. If the symbol contains a large number of pins, the quickview does not display all of them. (But that information can be derived from the text view.)

 Try opening and saving symbols that do not show quickviews.

Figure 3-19 shows the data browser that opens when you choose *Place – Manually* ([place manual](#)

command) and a quickview of the properties of the chosen object. The title bar reflects the object type you are browsing.

Figure 3.19: Data Browser for Manual Placement



If Quickview cannot display the preview or the properties of the element, a "Not Available" message appears in the quickview window.

Using Qvupdate to Display Quickview Information

This stand-alone program lets you update footprint information in design (.brd), drawing (.dra), padstack (.pad), or module (.mdd) databases that were created prior to release 14.0 so that text and graphics associated with them can be displayed in the Quickview window of file/library browsers. Without running Qvupdate, such information can be displayed in Quickview only by opening the pre-14.0 database in the editor's graphic environment and replacing the database using *File – Save (save* command). Qvupdate lets you update the footprint information for all pre-14.0 libraries in one operation though the use of the * wildcard character.

⚠ Qvupdate does not update symbols; you must update corresponding .dra files. Qvupdate automatically generates symbols from the .dra file.

Note the following conditions:

- Saving pre-14.0 databases in batch mode does not update the footprint information.
- Running uprev does not add the Quickview data to a layout database.
- Databases that were created prior to release 13.0 may have to be upreved before running Qvupdate.

For procedural information on using qvupdate see the [qvupdate command](#) in the *Allegro PCB and Package Physical Layout Command Reference*.

Database and Library Selections

In default mode (Database), data browsers list all the elements in the design's database. You can also view all named elements in the editor libraries when you check *Library*. The elements listed in Library mode may sometimes include items already in the design. This is because database items remain displayed in the list box when the library option is checked.

If an object in the database has the same name as an object in the library but contains different content, the database object takes precedence in the data browser; that is, the database object is chosen.

When you check the *Library* option, it reopens in Library mode for the duration of the design session, or until you de-select the library option.

To choose a database object:

1. Choose an application that prompts you for data by opening a data browser. (Specific instances are covered in the appropriate sections of this user guide.)
2. If the object you are looking for is not listed in the design's database, click *Library* to get a listing of all elements in the library.
You can filter the elements displayed in the list box by typing a string (partial object name) and a "wildcard" character in the field. For example:
 - Type FLAT* to display all object names that begin with FLAT.
 - Type FLAT*x to display all object names that begin with FLAT and end with x.
 - Type FLAT ?, where ? represents any single character.

Data browsers remember filters that you enter in the field. They can be reviewed by clicking the arrow button to the right of the field.

- Highlight a filter by clicking on it or by using the up-arrow/down-arrow keys on the

keyboard.

- Close the filter history menu by clicking the arrow button.

3. Select the object name you want to place in the design using one of these methods:

- Choose the object name.

The object name is highlighted and appears in the field.

- Type the object name in the field.

The data browser searches the design database, then the library files for the object. If the name you are looking for is in the library, the *Library* check box turns on to indicate the object's location.

- Double-click on the object name.

The object is chosen and the data browser closes.

4. Do one of the following:

- Choose *OK*.

The data browser closes and the chosen valid object is ready to be placed in the design. (*OK* does not close the browser until a valid object name is chosen.)

or

- Choose *Cancel* to close the data browser without placing an object.

Using Strokes and Associated Commands

You can run certain commands using predefined patterns of mouse strokes that you draw in the Design window. The layout editor interprets the pattern as a command and executes the command when you complete the stroke.

You can use the layout editor default `allegro.strokes` file located in the `$cdsroot\share\pcb\text` directory or you can create your own file using the Stroke Editor.

The layout editor looks for `.strokes` files in this order:

1. Current working directory
2. `\pcbenv` directory
3. `$cdsroot\share\pcb\text` directory

If you create a new `.strokes` file, store it in your current working directory or in the `\pcbenv` directory.

If you do not create a new strokes file, the Stroke Editor places a copy of the default `allegro.strokes` file in your `pcbenv` directory.

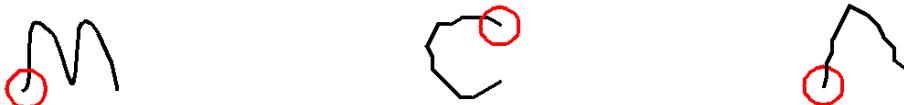
To create a `.strokes` file, or edit an existing `.strokes` file, see the [stroke editor](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Default `.strokes` File

The following table shows the strokes and associated commands in the default `allegro.strokes` file.

Using the default `allegro.strokes` file, you can:

- Execute the `zoom world` command by drawing a W stroke anywhere on the design.
- Zoom into an area of a design by drawing a Z stroke in the specified area of the design.
- Move, copy, and delete by drawing the M, C, and D strokes respectively. The stroke selects the object under the first point of the stroke, shown here as circles in the patterns.



For more information on the commands listed in this section, see the appropriate sections of the *Allegro PCB and Package Physical Layout Command Reference*.

Running Commands Using Strokes

To run commands using strokes:

1. In the Design Window, place the cursor over the object you want to move, copy, or delete, or over the area you want to zoom into. (You can draw the `world` command anywhere in the Design window.)
2. Press and hold down the `Control` key and the right mouse button *at the same time* to make a stroke.
As you move the mouse, you see the pattern being drawn.
3. When the stroke is complete, release the right mouse button.
If the layout editor recognizes the stroke, the associated command runs. If it does not recognize the stroke, the layout editor displays the following message:

Stroke not recognized.

You must enter strokes in the same direction in which they were created either in the default `.strokes` file or a customized file. This means that if you are creating your own `.strokes` file, you can have two strokes that look the same but issue different commands.

For example, if two strokes appear as diagonal lines, one can represent the `vertex` command, and the other the `delete vertex` command. The difference is that one stroke is drawn from upper left to lower right and the other from lower left to upper right.

- ⚠** You can set the `no_dragpopup` environment variable by choosing *Setup – User Preferences*. By default, you must hold down the `Ctrl` key and depress the right mouse button at the same time when using strokes. Setting this environment variable lets you depress the right mouse button and drag the mouse when using strokes. With this option, however, you lose the ability to choose popup menu items by pressing the right mouse button and dragging the mouse. Instead, you have to click twice with the right mouse button: once to see the popup and a second time to select a popup item.

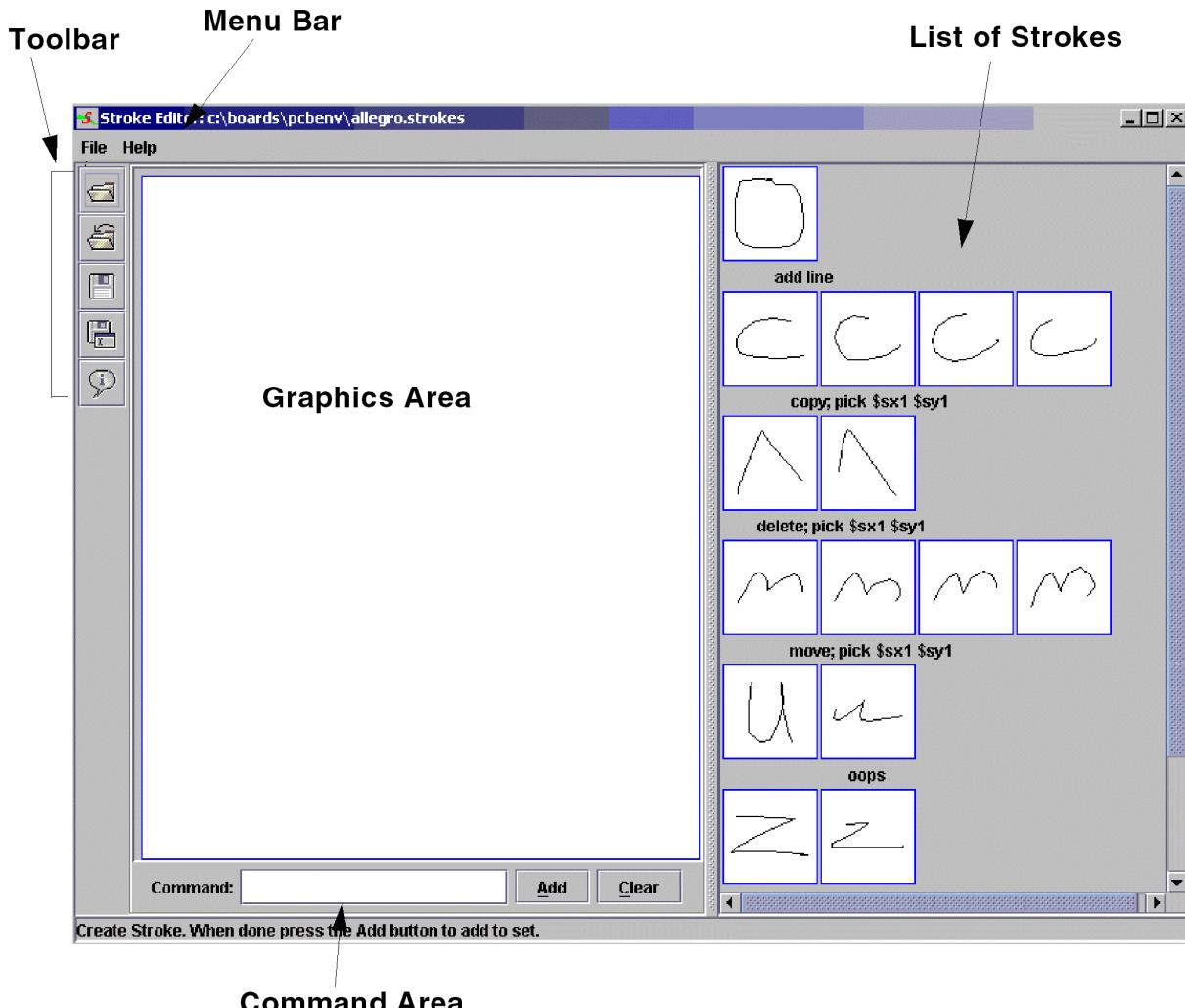
To specify a file containing your own strokes instead of using the default strokes file, see the [strokefile](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

The Stroke Editor

In addition to using the default `.strokes` file that shipped with the software, you can create stroke definitions with associated commands using the Stroke Editor. You can store these files on your system.

Figure 3-20 shows the Stroke Editor.

Figure 3.20: Stroke Editor



The Stroke Editor has these features:

- **Menu Bar** – Located below the title bar, the menu bar provides options for opening, closing, and saving files, saving a file with another name, and getting help for the layout editor and the Stroke Editor.
- **Toolbar** – Located on the left side of the Stroke Editor window, the icons provide the same options as the menu bar items.
- **Graphics Area** – Located on the left side of the window, the Graphics Area is the white portion of the window where you can draw a stroke.
- **Command Area** – Located below the Graphics Area, the Command Area lets you enter a command and associate it with the stroke shown in the Graphics Area. You can also clear existing strokes in the Graphics Area.

- **List of Strokes** – Located at the right side of the Stroke Editor window, the List of Strokes includes all the strokes and associated commands in the file.

 For the `move`, `copy`, and `delete` commands, a notation states `pick $xs1 $sy1`. This means that the stroke selects the object under the first point of the stroke.

To create a stroke file, or edit an existing stroke file, see the [stroke editor](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Defining Aliases

The alias feature lets you define a command vocabulary and create shorthand for commands you use most often. You can also program function keys (on most keyboards) to execute commands to increase speed and ease of work.

The alias is an alternative way of entering the command, but it does not disable the full commands. You can still use the standard form of the command.

This section describes how to establish an alias for typed entries and for function keys. Note that aliases work only in the layout editor, not at the operating system level.

A command alias entered at the command prompt is active only for the current work session. When you exit the layout editor and return to the operating system, aliases are lost.

To use command aliases repeatedly:

- Define and save them in a local environment file as described in [Managing Environment Variables](#).

Some default command aliases are provided with the layout editor. The sample global environment file lists the default aliases for the function keys and for the typed commands.

 `a` is used as an alias for `alias`.

You have several options at the keyboard. You can:

- Use standard commands.
- Use the default aliases.
- Define aliases for personal use.
- Define temporary aliases for an individual work session by entering the alias command at the console window prompt.

- Establish aliases in a local environment file that remain in effect at every login until you change the environment file.

For information on creating aliases, see the [alias](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

For information on deleting aliases, see the [unalias](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Assigning Function and Control Keys

The layout editor function and control keys take advantage of the capability provided by the native windowing systems in which you can execute your layout editor. This section describes the function and control keys.

 Some keyboards may not support all function key assignments.

You can assign layout commands to any function key that the editor can access through a native windowing system. The layout editor defines function keys from F1 through F10 and SF1 through SF10. Layout editors allow the use of Shift, Control, Alt keys, and navigation keys (Home, Up arrow, Esc, and so on) in combination with function keys to create aliases. Most alphanumeric keys can be used with control and shift key modifiers in creating aliases.

 The Alt key is not supported with the alpha-numeric since it is used by the Windows system to access the menu items via the keyboard.

 Control key modifiers C, V, and X are reserved for copy, paste, and cut tasks.

The following table shows you some examples of function and modifier key combinations:

Modifier	Indicator	Example
Shift	S	SF2
Control	C	CF2
Alt	A	AF2
Control	~	~N

You can combine modifiers in different ways. The following list of the by the following:

Modifier Combination	Example
Control-Shift F2	CSF2
Alt-Shift F2	ASF2
Control-Alt F2	CAF2
Control-Alt-Shift F2	CASF2
Control-Shift Z	~SZ
Shift-Up Arrow	SUp
Control-Up Arrow	CUp

Check the sample environment file for a list of the predefined function and control key aliases. The alias settings only apply to current session and are not saved to local environment file.

 Alias changes do not effect programs launched from layout editor such as, import logic, refresh symbol.

See the [alias](#) command in the *Allegro PCB and Package Physical Layout Command Reference* for examples on assigning function keys.

Layout Editor Limits and Accepted Characters

The limits are as follows:

Database resolution	Inches, Centimeters, Mils, and Millimeters: 4 places Microns: 2 places
Maximum number of connections	No limit

Maximum design area size	The maximum design area supported varies based upon design accuracy. The more places of accuracy the smaller the maximum area supported. With 0 places of accuracy we support 20 million design units. For example, with mils and 0 decimal places of accuracy we support 20 million mils (or 20,000 inches). If you increase the design accuracy to 2 decimal places the maximum design extent allowed is 200,000 mils or 200 inches.
Maximum number of design layers: (signal, power plane, drafting and so on.)	200 maximum ETCH/CONDUCTOR layers; 200 maximum layers per class (for each class)
Minimum signal width	No limit
Maximum signal width	No limit
Number of connections per net	No limit
Maximum via size	No limit
Number of definable vias	No limit
Via types	Thru, Blind, Buried, Microvia
Maximum number of text strings	No limit
Characters per text item	1000

Maximum text height	No limit
Rotation resolution	millidegrees (0.001 deg)
Maximum errors displayed	No limit
Minimum checking distance	No limit: whatever database resolution is

Unless otherwise indicated, the layout editor only supports uppercase characters. If you enter lowercase characters, the tool converts them to uppercase. Printable characters are generally any key on a standard keyboard with the exception of Tab, Backspace, Enter, function keys, Esc, and navigation keys (Arrows, Home, and so on.).

- ⓘ The exclamation mark ('!') is an invalid character. This character is used as field separator when data is extracted from the database. If you extract any field that contains a '!' character then incorrect data is resulted in the wrong columns. Moreover, if you attempt to save or open a file with '!' in the name, it is reported as illegal character.

Table 3.3: Acceptable characters

Field Name	Length	Acceptable Characters
file name	OS limit ²	no spaces
film name	18	filename ¹
device type	31	All printable except '
directory name	OS limit	OS limit ³

function designator	31^4	All printable except '
package name	31^4	a to z, 0 to 9, -, and _
padstack	31^4	a to z, 0 to 9, -, and _
pin number	31	All printable except '
pin name	31^4	All printable except '
property value	1023	All printable except '
net name	31^4	All except \ and ' ⚠ Set environment variable <i>legacy_character_set</i> to allow backslash (\) in the net names.
reference designator	31	all except * and ' Avoid using period (.) in refdes names because period is used as a delimiter when referring to a pin: for example, <refdes>. <pin number>.
slot name	31^4	all except '
swap type	31	all except '
text lines	1023	except ! ¹
tolerance	1023	all except '
user part number	1023	all except '
value	1023	all except '

¹Allows lower case for general text unless on a special layer where it may adhere to more restrictive rules; for example, many layers show Refdes.

² File names adhere to operating systems restrictions except if they are stored in the database, where they assume the least common denominator. For example, a *.psm* file becomes a package symbol in the database so its least common denominator is the package name restriction. Spaces in the name are not supported. It is strongly suggested that you use lower case, especially for those names stored in the tool database.

³ Directory names follow operating system limitations. The layout editor supports spaces in directory names on Windows.

⁴ The default maximum number of characters is 31. You can set the initial length for new designs to a maximum length of 255 by using the *allegro_long_name_size* environment variable (choose *Setup – User Preferences* ([enved](#) command)). You can change the size in existing designs by choosing *Setup – Design Parameters* ([prmed](#) command) and specifying a new maximum for the *Long Name Size* parameter in the Design tab.

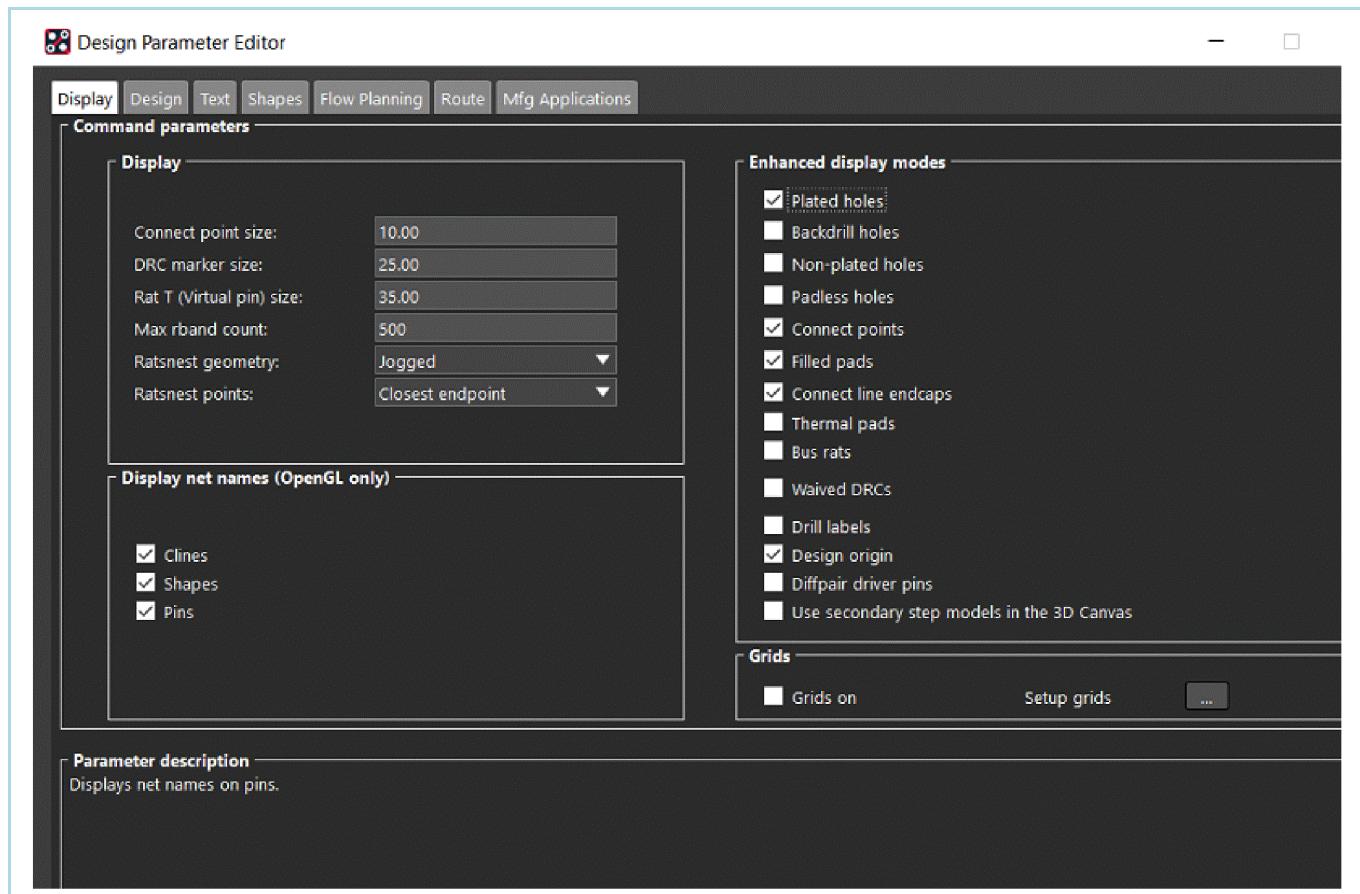
Setting Drawing Parameters

You set drawing parameters in the *Display* and *Design* tabs of the Design Parameter Editor.

1. choose *Setup – Design Parameters* ([prmed](#) command) to access the Design Parameter Editor or right-click in the pre-select use model and choose *Design Parameters* from the pop-up menu that appears.

The Design Parameter Editor organizes common parameters needed to set up a drawing, which entails specifying the following:

- Drawing parameters, including drawing extents, origin, type, and size; database accuracy; and user units
- Text size
- Grids
- Net names



⚠ You can reuse customized parameter settings from one design by exporting them to a database parameter file (.prm) with the *File – Export – Parameters (param out)* command). Then when you initially begin a design, import the .prm file with the *File – Import – Parameters (param in)* command. The `techfile` batch command can also be used to import or export database parameters.

Specifying Text Size

1. Use the *Text* tab of the Design Parameter Editor to specify the appearance of text in a design. For procedural information on formatting text, see the `define text` command in the *Allegro X PCB and Package Physical Layout Command Reference*.

Specifying Grids

1. Use the *Display* tab of the Design Parameter Editor to open the *Define Grids* dialog box, where you set the x and y values for both ETCH/CONDUCTOR and non-

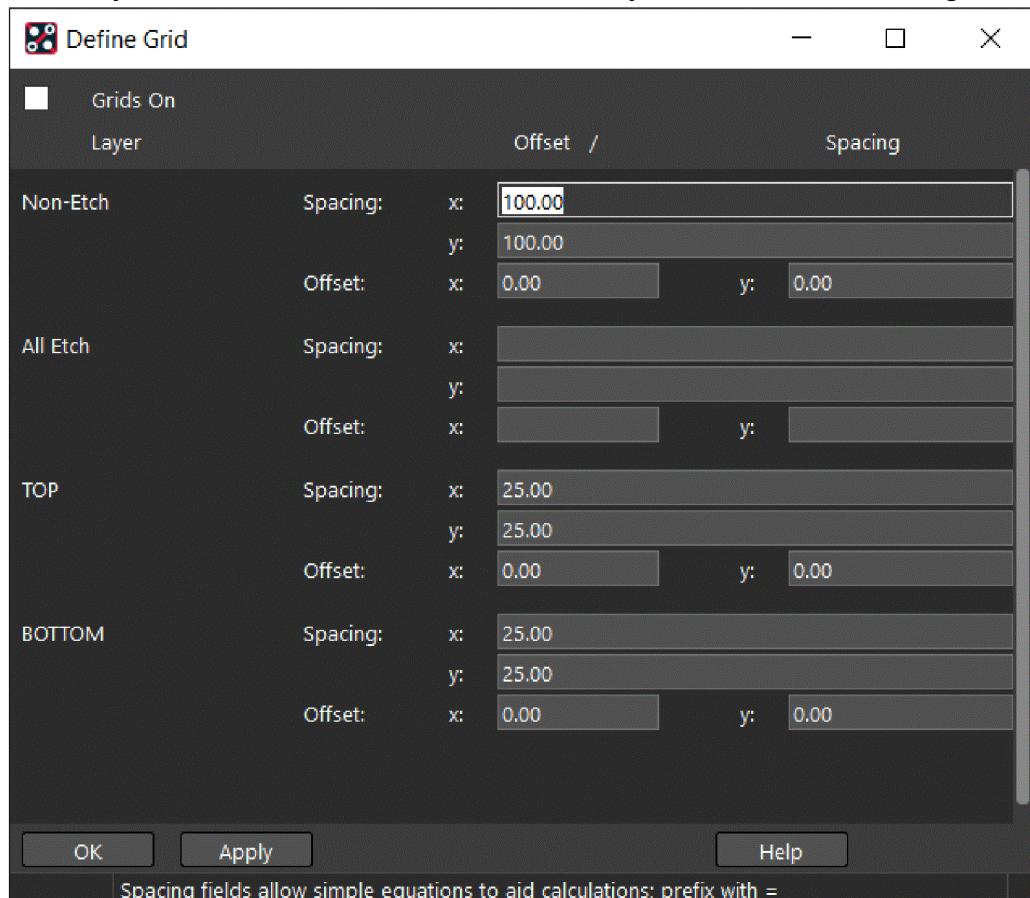
ETCH/CONDUCTOR grids in a design.

The dialog box also lets you customize the grid for each ETCH/CONDUCTOR layer in a design. For procedural information, see the [define grid](#) command in the *Allegro X PCB and Package Physical Layout Command Reference*.

All drawings, except Autoplacement, interactive routing, and Autorouting use non-ETCH/CONDUCTOR grid. All non-ETCH/CONDUCTOR layers use the same, single-increment grid with the grid points spaced evenly across the design.

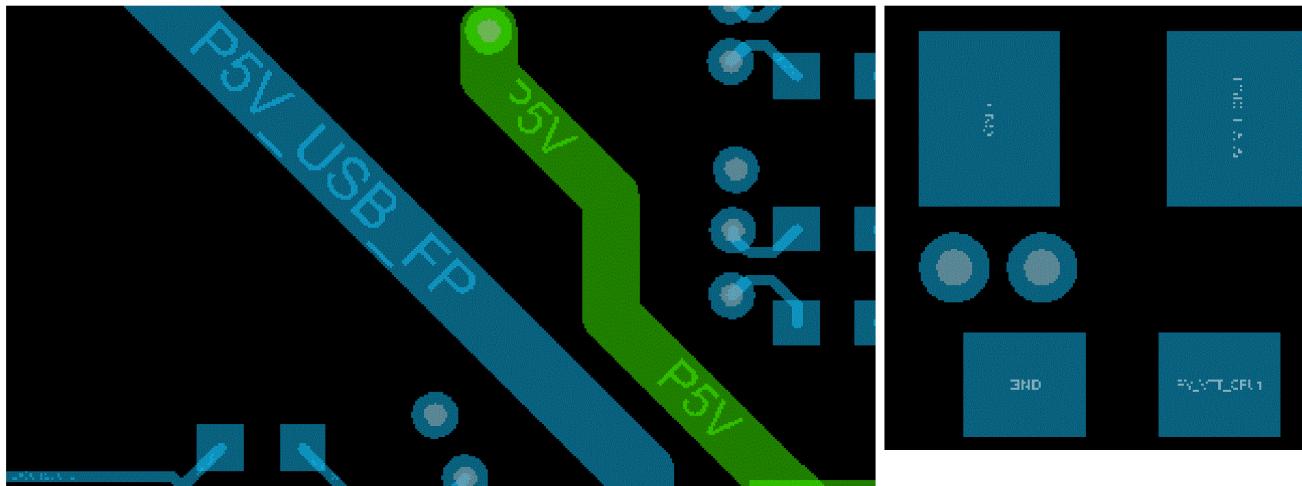
ETCH/CONDUCTOR grids are dedicated routing grids for both interactive and autorouting. You can use a separate x, y grid for each ETCH/CONDUCTOR layer in a design. In addition, you can set a single increment value for each ETCH/CONDUCTOR grid, or you can set different values for non-ETCH/CONDUCTOR grids and ETCH/CONDUCTOR grids.

You can enter values into the Grids Display dialog box to reset the point of origin for x and y, as well as the spacing between the grid points for x and y. The default point of origin for all layers is x=0, y=0. The default increment setting for non-ETCH/CONDUCTOR layers is x=100, y=100. For ETCH/CONDUCTOR layers, the default setting is x=25, y=25.

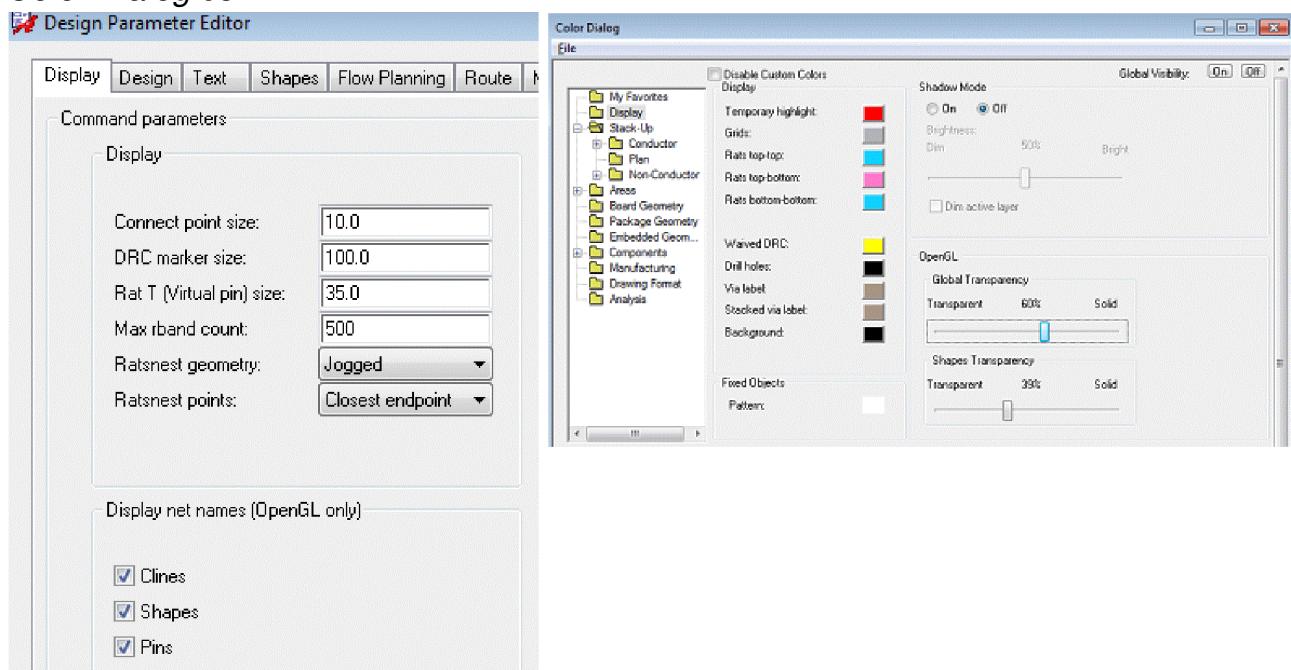


Displaying Net names

You can set the display for net names within the clines and bond wires, pins, and shapes. If set, when moving the wire bonds (push and shove) the cline segment attached to the shoved bond fingers and bond wires show the net name.



1. Use the *Display* tab of the *Design Parameter Editor* to set the display for net names.
2. For displaying net names, enable *OpenGL* and set transparency to less than *Solid* level in the *Color Dialog* box.



Classes and Subclasses in Layout Editors

In the layout editors, categories of drawing elements are called *classes*. Classes represent all types of visible items in the design. A few examples of classes are:

ETCH/CONDUCTOR	Represents pieces of copper forming electrical connections.
PINS	Represents defined pads and holes.
BOARD	Represents the physical outline of the design and other geometry related to the PCB.
PACKAGE	Represents the physical components of the design.

The parts of the drawing in each class are called *subclasses*. Each class can contain many subclasses, including some that you define.

Classes and subclasses identify how every element is to be used in a design. For example, *Add – Line* ([add line command](#)), used when Board is the active class, adds a simple geometric graphic element to a design. The same command, used when ETCH/CONDUCTOR is the active class, adds a connecting line of etch/conductor to the design because the command correlates the function with the class of element.

Subclasses allow a further degree of classification that allows the tool to treat data more specifically. For example, ETCH/CONDUCTOR has two pre-defined subclasses associated with it: Top and Bottom (thus eliminating the necessity of referring to element types by layer number). You also have the option of defining subclasses. ([See Creating User-Defined Subclasses.](#))

Table 3-2 lists groups of classes and their pre-defined subclasses. Note that the Allegro product you are running may not include all the classes/subclasses listed here. In addition, the subclasses in a design vary depending on layers added to or deleted from it.

For more information see, [Classes and Subclasses in Layout Editors](#).

To view colors assigned to the classes and subclasses in the design, choose *Display – Color/Visibility* ([color192 command](#)), described in the *Allegro X PCB and Package Physical Layout Command Reference*.

Classes and Subclasses			
Group	Class	Subclasses	
Geometry	Board	OUTLINE	PLATING_BAR
		ASSEMBLY_NOTES	TOOLING_CORNERS
		DIMENSION	PLACE_GRID_TOP
		PLACE_GRID_BOTTOM	TOP_ROOM
		BOTTOM_ROOM	BOTH_ROOMS
		SWITCH_AREA_TOP	SWITCH_AREA_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		ASSEMBLY_DETAIL	SOLDERMASK_TOP
		SOLDERMASK_BOTTOM	OFF_GRID-AREA
		ASSEMBLY_TOP	ASSEMBLY_BOTTOM
	Package	PLACE_BOUND_TOP	PLACE_BOUND_BOTTOM
		PIN_NUMBER	PAD_STACK_NAME
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		BODY_CENTER	SOLDERMASK_TOP
		SOLDERMASK_BOTTOM	DISPLAY_TOP
		DISPLAY_BOTTOM	MODULES
		DFA_BOUND_TOP	DFA_BOUND_BOTTOM
		PASTEMASK_TOP	PASTEMASK_BOTTOM
		SHAPE PROBLEMS	NO_GLOSS_ALL
Manufacturing	Manufacturing		

		PHOTOPILOT_OUTLINE	NOGLOSS_BOTTOM
		NOGLOSS_TOP	BACKDRILL-FLAG-TOP
		NCLEGEND (combines former NCDRILL_LEGEND and NCDRILL FIGURE)	BACKDRILL-FLAG-BOT
		NOGLOSS_INTERNAL	PROBE_TOP
		PROBE_BOTTOM	AUTOSILK_TOP
		AUTOSILK_BOTTOM	NO_PROBE_TOP
		NO_PROBE_BOTTOM	SHAPE PROBLEMS
		FIXTURE_BOTTOM	FIXTURE_TOP
Drawing Elements	OUTLINE		TITLE_BLOCK
	TITLE_DATA		REVISION_BLOCK
	REVISION_DATA		
Stack-Up	DRC	TOP	BOTTOM
		THROUGH ALL	PACKAGE_TOP
		PACKAGE_BOTTOM	I
ETCH	TOP		BOTTOM
	TOP		BOTTOM
Anti-ETCH	TOP		BOTTOM
	INTERNAL LAYERS		THROUGH ALL
Pin	TOP		BOTTOM
	SOLDERMASK_TOP		SOLDERMASK_BOTTOM
	PASTEMASK_TOP		PASTEMASK_BOTTOM
	FILMMASKTOP		FILMMASKBOTTOM
Via	TOP		BOTTOM

		SOLDERMASK_TOP	SOLDERMASK_BOTTOM
		PASTEMASK_TOP	PASTEMASK_BOTTOM
		FILMMASKTOP	FILMMASKBOTTOM
Components	Refdes	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		DISPLAY_TOP	DISPLAY_BOTTOM
	Component Value	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		DISPLAY_TOP	DISPLAY_BOTTOM
	Device Type	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		DISPLAY_TOP	DISPLAY_BOTTOM
	Tolerance	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		DISPLAY_TOP	DISPLAY_BOTTOM
	User Part Number	ASSEMBLY_TOP	ASSEMBLY_BOTTOM
		SILKSCREEN_TOP	SILKSCREEN_BOTTOM
		DISPLAY_TOP	DISPLAY_BOTTOM
Areas	Route Keepin	THROUGH ALL	
	Route	THROUGH ALL	TOP
		BOTTOM	
	Via Keepout	TOP	BOTTOM
		THROUGH ALL	

Package Keepin	THROUGH ALL	
Package Keepout	THROUGH ALL	TOP
	BOTTOM	

User-Defined Subclass Creation

You can create user-defined ETCH/CONDUCTOR and non-ETCH/CONDUCTOR subclasses in the layout editor. The ETCH/CONDUCTOR subclasses identify the layers or cross-section of the design. There are several non-ETCH/CONDUCTOR subclasses that you can create, including the following:

- Board
- Component Value
- Device Type
- Drawing Format
- Manufacturing
- Analysis
- Package Geometry
- Ref Des
- Tolerance
- User Part Number

Use the [define subclass](#) command to create a subclass. Alternatively, use the *Setup – Subclasses* menu command.

For information on creating both types of subclasses, see the [define subclass](#) command in the *Allegro X PCB and Package Physical Layout Command Reference*.

User-Defined Subclass Mapping for DFM Checks

The mask type and layer association of a non-etch subclass to its corresponding DRC subclasses is determined by the subclass names:

- `SOLDER`, `PASTE`, or `SILKSCREEN` substring: Indicates that the DRC subclass is to be treated as soldermask, pastemask, or silkscreen, respectively.
- `_TOP` or `_BOTTOM` suffix: Indicates that the subclass is to be associated with the `TOP` or `BOTTOM` layer, respectively.

However, if a user-defined non-etch subclass does not follow the defined naming criteria, a configuration file is used to identify the `TOP` or `BOTTOM` layer association and the subclass type (mask or silkscreen).

A sample configuration file, `dfmUserDefinedSubclassMapping.txt`, is available at the following location: `<installation_directory>/share/pcb/dfm`.

Creating Custom Subclass Mapping

To create custom subclass mapping, follow these steps:

1. Copy the sample configuration file to the working directory.
2. Add custom subclasses with mapping details as instructed in the configuration file.
3. If the same rules need to be applied at the project, site, or organizational level, copy this file to the `CDS_SITE` location.

User-Defined Subclass Mapping for DFM Checks

The mask type and layer association of a non-etch subclass to its corresponding DRC subclasses is determined by the subclass names:

- `SOLDER`, `PASTE`, or `SILKSCREEN` substring: Indicates that the DRC subclass is to be treated as soldermask, pastemask, or silkscreen, respectively.
- `_TOP` or `_BOTTOM` suffix: Indicates that the subclass is to be associated with the `TOP` or `BOTTOM` layer, respectively.

However, if a user-defined non-etch subclass does not follow the defined naming criteria, a configuration file is used to identify the `TOP` or `BOTTOM` layer association and the subclass type (mask or silkscreen).

A sample configuration file, `dfmUserDefinedSubclassMapping.txt`, is available at the following location: `<installation_directory>/share/pcb/dfm`.

Creating Custom Subclass Mapping

To create custom subclass mapping, follow these steps:

1. Copy the sample configuration file to the working directory.
2. Add custom subclasses with mapping details as instructed in the configuration file.
3. If the same rules need to be applied at the project, site, or organizational level, copy this file to the `CDS_SITE` location.

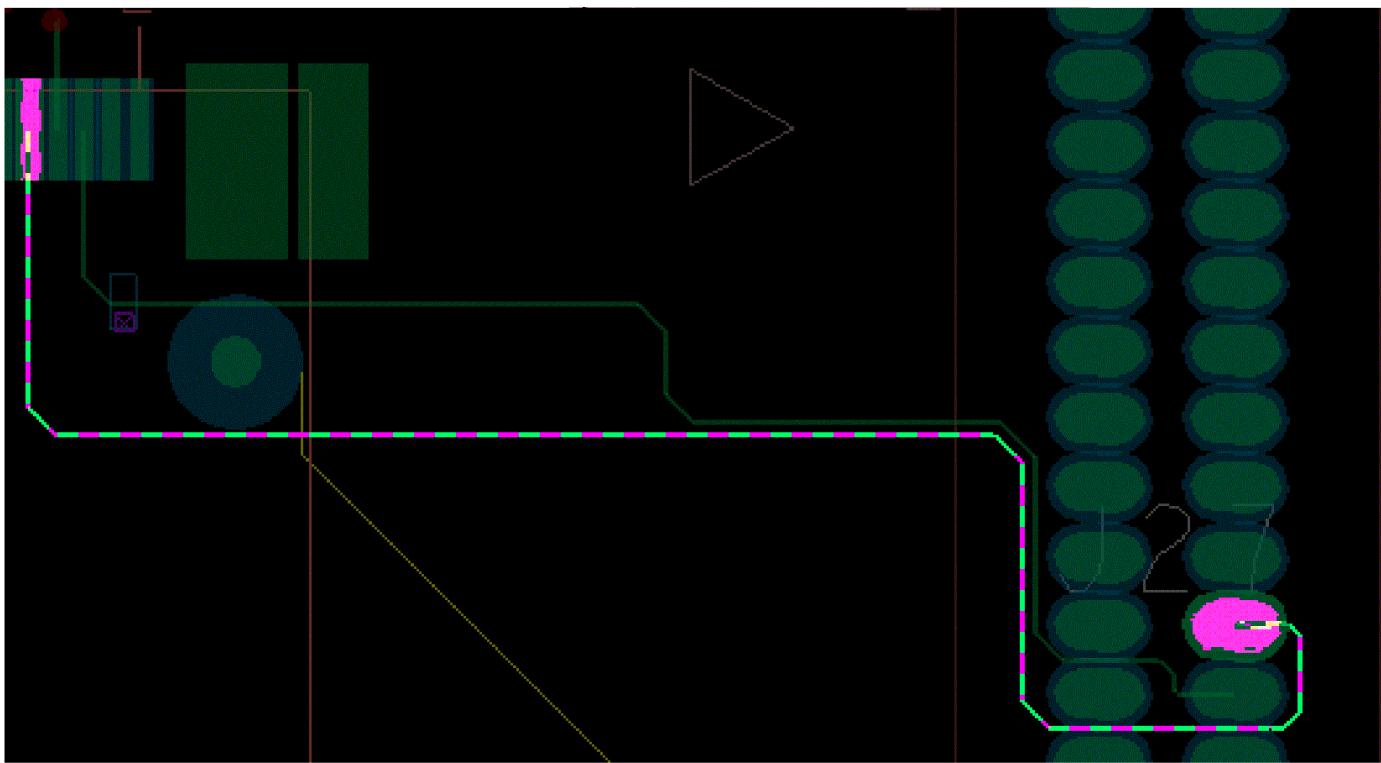
Working with Highlighting and Coloring

Accentuate an element by using the following methods:

- Use a highlight pattern comprising the element's base subclass color and the temporary highlight color defined in the *Display* category of the Color dialog box.
- Override the base subclass color with a custom color but without applying a highlight pattern.
- Assign a custom color plus a highlight pattern.

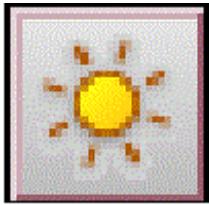
Highlighting Design Elements

Highlighting accentuates certain elements, often nets, with a pattern—or striping—rather than a color, to locate them more easily during debugging. Once the element becomes highlighted in the design canvas, its name also displays in a bold font in the *Nets* section of the Color dialog box.



⚠ Striping is only visible when the `display_nohilitefont` variable is disabled.

- Menu: *Display – Highlight*.



- Icon:
- Color Dialog: Select a cell (net only) in the *Nets* grid, then right-click and choose *Set Highlight State*.
- Pre-selection mode: Hover over an element, then right-click and choose *Highlight*.

Assigning Colors to Design Elements

Colors can be assigned to Nets, Symbols, Pins, DRCs, Groups, and Functions using the following methods:

- Menu: *Display – Assign Color* ([assign color](#) command), set the Find window pane, and

choose a color from the *Options* window pane.

- **Icon:** Placeholder for new Icon
- **Color Dialog:** Use the *Nets* grid to color Net elements.
- **Pre-selection mode:** Hover over an element, right-click and choose *Assign Color*, then choose a color from the palette that displays.

When you choose *Display – Assign Color*, the following displays in the Options window pane, where you assign a custom color and also choose a highlight pattern.



When you right-click and choose *Assign Color* from the pop-up menu, the following palette displays, from which you can assign a color as well as highlight an element:



The element's color changes in the design canvas and in the *Nets* section of the Color dialog box.

Unassigning Colors

Color overrides can be removed by using the dehighlight command.

- **Menu:** *Display – Dehighlight*; set the Find window pane, or work with elements the Options window pane.



- **Icon:** [Dehighlight icon]
- **Color Dialog:** Select a cell in the *Nets* grid, then right-click and choose *Clear Custom Color*.
- **Pre-selection mode:** Hover over an element, then right-click and choose *Dehighlight*.

The Color Dialog Box

The Allegro layout editors support a palette of 192 modifiable colors, 96 of which appear at one time in a primary color palette, which is the Cadence default, and another 96 which appear in secondary

palette, used for customization. The first 24 positions are reserved for colors used in pre-Release 16.0 databases. Choose *Display – Color/Visibility* ([color192](#) command) to display the *Color* dialog box, which comprises the *Layers* and *Nets* grids.

The *Layers* grid primarily controls the color and visibility settings of classes and subclasses, along with levels of transparency for the design and shapes. Use the *Layers* grid to also control shadow dimming, highlighting, ratsnest display, waived DRCs, and drill holes. You can create your own unique colors or palettes that may be saved to external .col files and then applied to other designs. You can also search for and display subclass layers by entering text in the *Filter layers* field.

The *Nets* grid allows each element of a net, including clines, pins, vias, shapes, and rats, to be uniquely color coded to differentiate them from other nets or net elements on a layer.

The *Favorites* tab allows you to maintain and use your favorite layers.

The *Visibility Pane* tab lets you customize the *Visibility* window pane.

You can add subclass layers to the *Favorites* or *Visibility Pane* tabs from the pop-up menu. In each of the tab, hovering over a color swatch under *Available Colors* displays the class and subclass where the color is used.

You can re-use customized layer or net colors defined in one design in other designs by creating a database parameter (.prm) file with *File – Export – Parameters* ([param out](#) command) and choosing to include the *Color Layer* and *Color Net* parameters.

Using the Layers Grid

Use the *Layers* section of the *Color* dialog box for the following tasks:

- [Assigning Subclass Colors and Enabling Visibility](#)
- [Controlling Ratsnest Colors](#)
- [Setting Graphics Transparency](#)
- [Creating My Favorites’ Folder](#)
- [Customizing Design Colors](#)

Figure 3.21: Layers Grid of the Color Dialog Box



Assigning Subclass Colors and Enabling Visibility

The *Layers* grid lets you assign color and visibility to individual subclasses or to quickly enable or disable color and visibility settings for entire subclasses. The color boxes allocate color across a column or row, as Figure 3-2 shows. The white boxes control visibility.

Figure 3.22: Assigning Colors and Enabling Visibility

The diagram illustrates the 'Layers' grid interface. It features a header row with columns labeled 'Subclasses', 'All', 'Pin', 'Via', 'Etch', 'Drc', 'Anti Et', and 'Bndry'. Below this is a row for 'All' subclasses. The main body contains rows for various subclasses: Top, Gnd, Sig1, Sig2, Vcc, Vcc1, Sig3, Sig4, Gnd1, and Bottom. Each row has a color box in the first column and visibility checkboxes for each subsequent column. Annotations with arrows point to specific elements: 'Choose this box to enable visibility globally' points to the top-left 'All' checkbox; 'Choose this box to apply color across entire row' points to the color box for the 'All' row; 'Choose this box to enable visibility of entire column' points to the 'Etch' column's visibility checkbox; and 'Choose this box to enable visibility of entire row' points to the 'Sig3' row's visibility checkboxes.

Subclasses	All	Pin	Via	Etch	Drc	Anti Et	Bndry
All	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Top	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Gnd	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sig1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sig2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Vcc	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Vcc1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sig3	<input checked="" type="checkbox"/>						
Sig4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Gnd1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Bottom	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

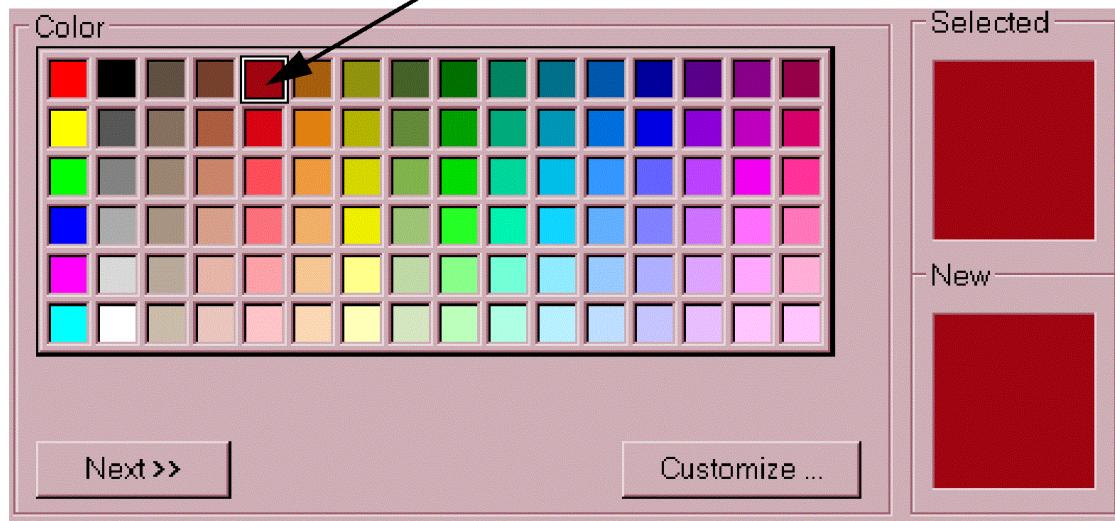
There may be colors assigned to subclasses suitable for re-use on other subclasses. Similar versions of the color may exist in the color palette, so to source the exact color, hover over the color assigned to a subclass, then right-click and choose *Select Color*. This outlines the color used in the palette, even toggling between the primary and secondary palettes if necessary.

Figure 3.23: Sourcing Colors for Reuse

Hover over this color box, right-click and choose *Select Color*

Subclasses	All	Pin	Via	Etch	Drc	Plan	Anti
All							
Top							
Gnd							
In1							
In2							
Gnd1							
Bottom							
Soldermask_Top							

This color box becomes active



Controlling Ratsnest Colors

To differentiate the display of ratsnests, a side-centric coloring scheme is available using the following options in the *Display* folder.

- *Rats top-top*: Specifies the color of ratsnest lines that connect top-side only components (start-end pin on top).
- *Rats top-bottom*: Specifies the color of ratsnest lines (one pin on top, other on bottom).
- *Rats bottom-bottom*: Specifies the color of ratsnest lines that connect bottom-side only components (start-end pin on bottom).

Controlling the Visibility of Individual Elements with Shadow Mode

To highlight specific elements in a design without affecting the visibility settings of that object's entire subclass, use the Shadow mode feature in the *Display* folder. Shadow Mode is used with the `hilight` and `dehilight` commands, as well as various interactive commands. When you enable Shadow Mode, the following occurs:

- The *Brightness* setting slide bar moves to its last applied percentage of brightness. The initial default percentage setting is 50%.
- The colors in the *Color* section dim to the chosen percentage of brightness in the slide bar. This allows you to "preview" how the colors in the design display if you click *Apply* or *OK*.
- *Dim active layer* lets you dim the active layer of a design. Dimming the active layer if it contains a large number of elements displayed normally (non-highlighted) can increase the effectiveness of Shadow Mode. You can dim the active layer with the check box in the *Color* dialog box or in the *Options* tab when shadow mode is turned on.
- The design elements of the current active drawing dim to the percentage of brightness set in the slide bar.

Shadow Mode Display Options

With Shadow Mode active, design elements display in the following ways:

- Normal. elements on the active layer of a design remain unaffected by Shadow Mode unless you select the *Dim active layer* in the *Options* tab.
- Highlighted, either permanently by way of the `hilight` command, or temporarily when you run an interactive command. In this state, elements are unaffected by Shadow Mode. Elements affected or added by a current interactive command are temporarily highlighted while the command is active. For example, if you run `add connect` with Shadow Mode on, the elements highlighted include:
 - Interconnecting pins
 - Existing etch/conductor being tied into
 - Connect lines, vias, and DRCs

When you complete the command, the added/affected elements are dimmed.

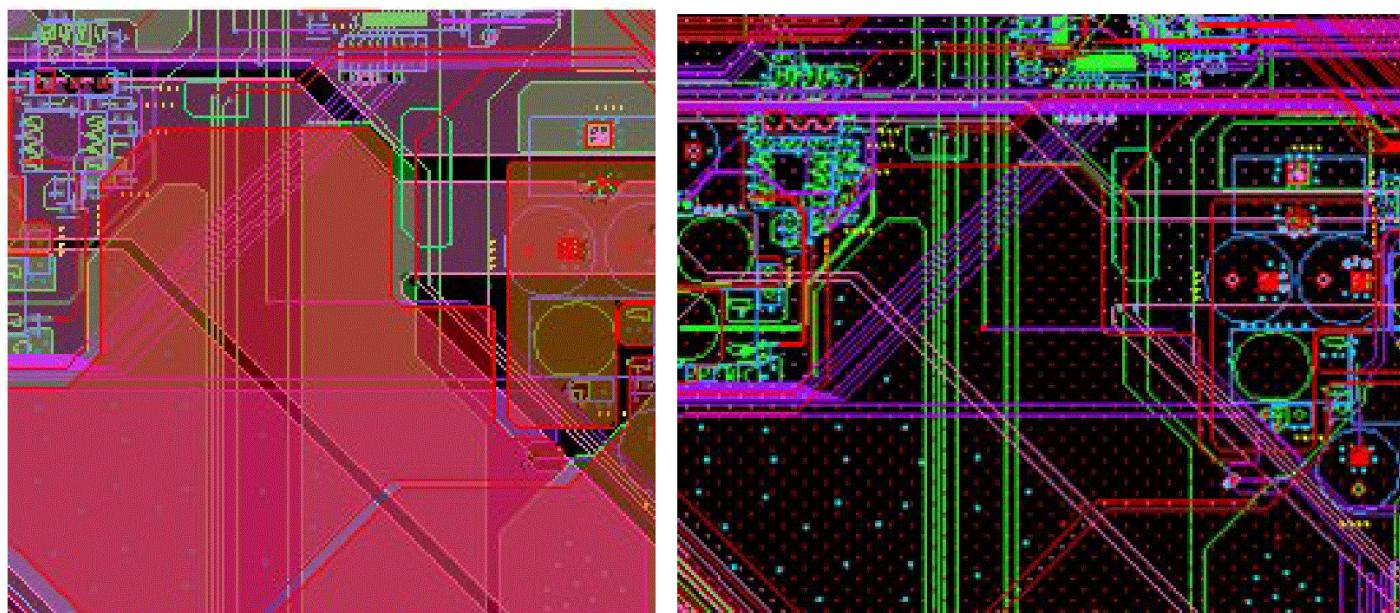
- Dim. The elements are unaffected by the conditions described above. The degree of dimming depends on the percentage of brightness set in the Color dialog box.

You can set global Shadow Mode parameters through the use of keyboard commands entered at the command window prompt, allowing you to assign function keys or toolbars to the dimming controls. For information on the syntax for setting Shadow Mode at the command prompt, see the [shadow](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Setting Graphics Transparency

OpenGL affords graphics transparency, which can be controlled at the global or shape level. A slide bar tailors the display from traditional solid to hollow fill, as Figure 3-4 shows. Transparent graphics allow more layers to display, including plane layers, that often block the graphics on other layers.

Figure 3.24: OpenGL Enabled (left) and OpenGL Disabled (right)

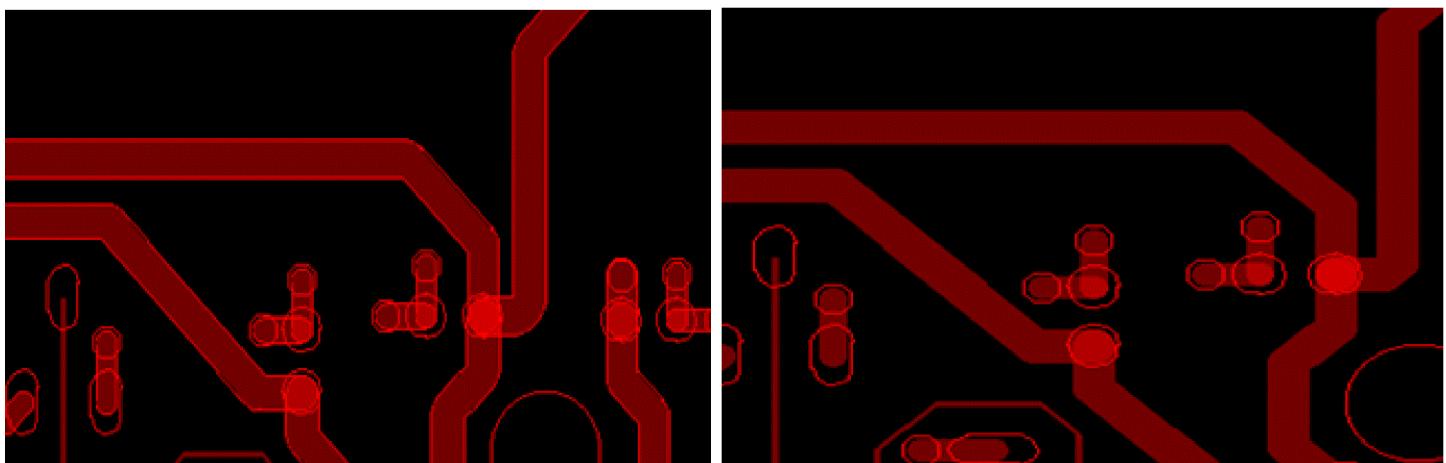


OpenGL is enabled by default. You can disable it using the environment variable *disable_opengl* in the User Preferences Editor dialog box.

Running Allegro with OpenGL requires a workstation with CPU board with at least 128 MB of memory and 128-bit bus interface. Only the 2D mode is supported. OpenGL requires higher-level graphics cards for best performance.

To display polyoutlines as Figure 3-5 shows, set the environment variable *draw_etch_outline* in the User Preferences Editor dialog box, available by choosing *Setup – User Preferences* ([enved](#) command).

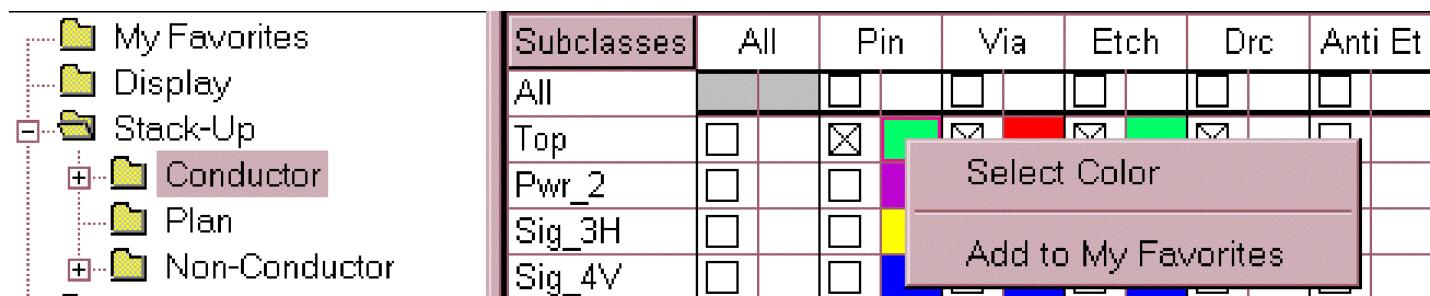
Figure 3.25: Polyoutline (l) and No Polyoutline (r)



Creating My Favorites' Folder

Use the *My Favorites*' folder to store frequently accessed subclasses where either the visibility or color changes often. Hover your cursor over the color box associated with a subclass, right-click and choose *Add to My Favorites*. The subclasses are copied, rather than moved, to the *My Favorites*' folder.

Figure 3.26: My Favorites



Remove a subclass from the *My Favorites*' folder by hovering your cursor over the color box, right-clicking, and choosing *Remove from My Favorites*.

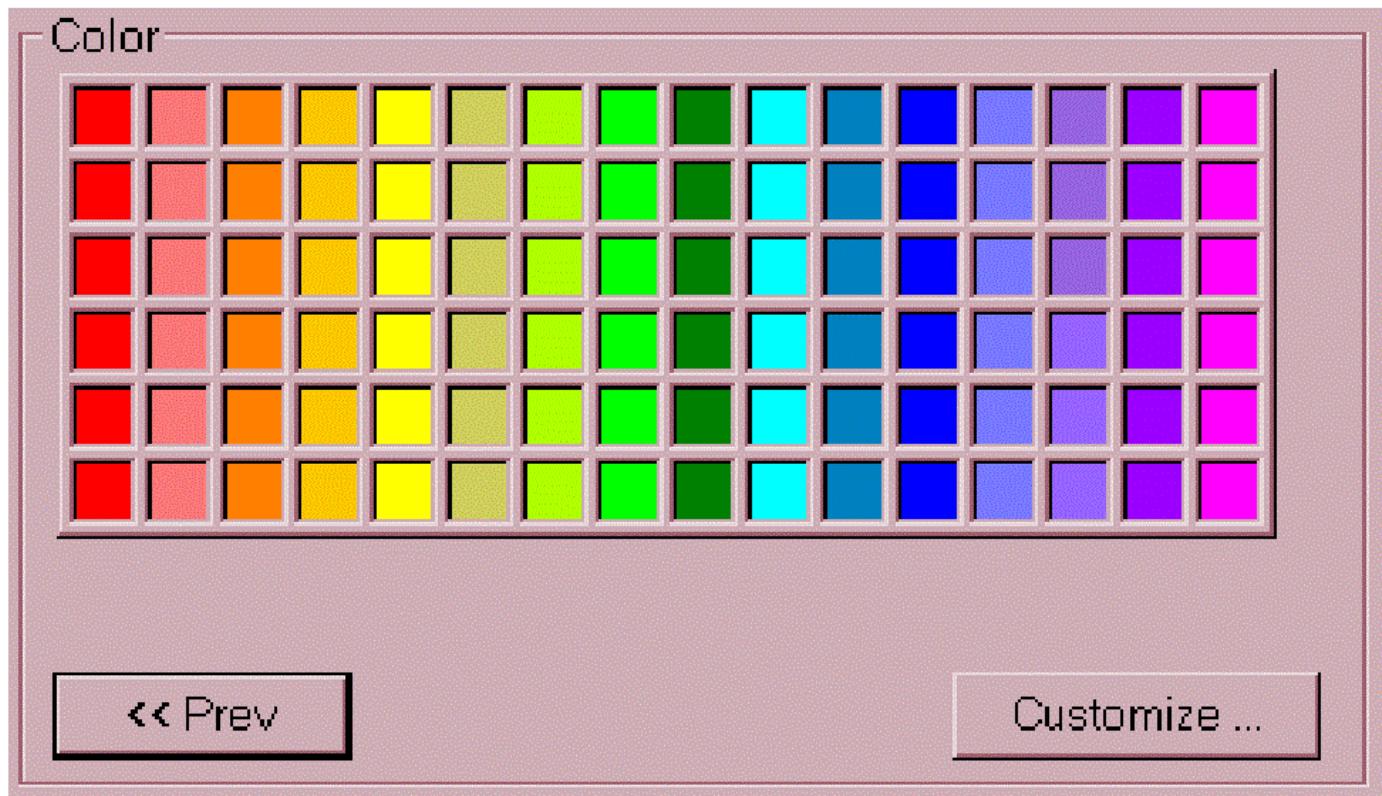
Saving and Reusing Color Palettes

When a design initially opens, the default color palette displays (Figure 3-7), which comprises an array of 16 x 6 colors. The first column comprises popular colors typically used in designs. This palette can always be reloaded using *File – Load Default Cadence Color Palette*.

Figure 3.27: Default Cadence Color Palette



Clicking *Next* displays the secondary palette, used for customization of colors, in Figure 3-8 :
Figure 3.28: Secondary Color Palette

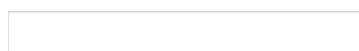


A color palette may be customized and saved to an external .col file using *File – Save Color Palette*. You can then apply a unique color palette to other designs using *File – Load Color Palette*.

Customizing Design Colors

You can customize shades and hues of any color with the *Customize* button, which displays a Color dialog box shown below in Figure 3-9. After moving the control on the vertical sliding bar for luminosity away from the extremes of white or black, you can move the crosshair around the spectrum. All the fields in the dialog box reflect the correct number for the color in the crosshair. You can also type values in the fields to choose a color, click *Add to Custom Colors*, and then *OK*.

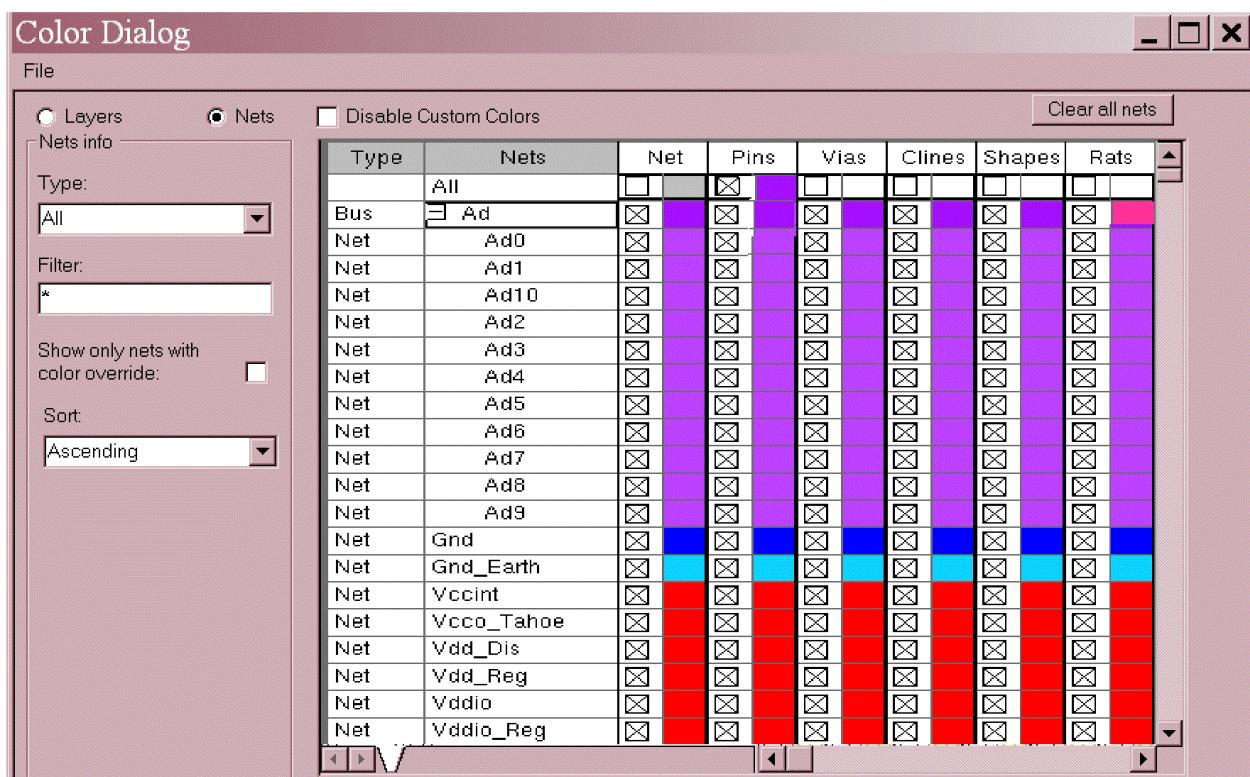
Figure 3.29: Color Customization dialog box



Using the Nets Grid

Database elements may be displayed using either the class/subclass color or a single color assigned to an element, also known as a custom color. To assign a custom color to an entire net or to its pins, vias, clines, shapes, or rats, you use the Nets grid. Assigning a custom color automatically enables the custom color state for that element as well, meaning that the custom color displays in the design canvas. You can also define how the custom color displays the element using a combination of states—none, highlight or custom color state, or highlight plus custom color state—all of which may be set independently.

Figure 3.21: Nets Grid



Highlighting is set or unset by right clicking and choosing *Set Highlight State* or *Clear Highlight State*, respectively.

To prevent custom color from displaying in both the Nets tree and the design, right click and choose *Clear Custom Color*. (A color box without a color assigned to it has no custom color state.) These custom color and highlighting states affect the display of the element as follows:

Element has custom color?	Highlight state?	Custom color state?	Element displays using...
Yes	No	No	Class/subclass color
Yes	Yes	No	Highlighted
Yes	Yes	Yes	Highlighted
Yes	Yes	No	Custom color, no highlighting
No	Yes	No	Highlighted with temp highlight color
No	No	No	Class/subclass color

Net Color Inheritance

Net elements can inherit highlighting and custom color from their parent. Inheritance can apply to specific elements as well. When a custom color is assigned to pins of an Xnet, for instance, all the pins of nets that belong to that Xnet inherit the custom color. A custom color or state of specific objects may also be overridden. For example, pins of a net can be assigned a color that differs from that of the net.

Saving Visibility Assigned to Classes and Subclasses

A color visibility view saves the visibility assigned to classes and subclasses as a collection of layer visibility settings that you can apply to subsequent designs using the *Views* field on the *Visibility* window pane. You save the settings in a file with a *.color* extension, stored in the directory specified by the *viewpath* variable, accessible in the *Paths-Config* folder in the *Categories* section of the *User Preferences* dialog box. A color view also displays film record visibility settings stored in the current design.

Color views (*.color* files) display in the *Views* field as *File: <name>*. Film record names display there as *Film: <name>*, unless you suppress the film record names from the list of color views in the *Visibility* window of the control panel. Suppress these names by selecting the *color_nofilmrecord* environment variable in the [The User Preferences Editor](#). Restart the layout editor for changes to the variable's value to take effect.

You can do the following tasks, all of which are described in the *Allegro PCB and Package Physical Layout Command Reference*:

- Create or change a color visibility view, using *View – Color View Save* ([colorview create](#) command)
- Delete a color visibility view, described under *View – Color View Save* ([colorview create](#) command)
- Load a color visibility view, using the [colorview load](#) command
- Apply the previous color visibility view, using *View – Color View Restore Last* ([colorview restore](#) command)

APD: Highlighting Sets of Pins

Highlighting objects in a package under design allows for quicker recognition of objects and, as a result, easier, more efficient design. The current IC Packaging tools provide the capability to permanently highlight nets, functions, symbols, or pins. Because functions are not widely used inside package designs, they rarely provide additional capability over symbol-based highlighting. Also, the current pin highlighting must be done based on window picks or by using the *Find By Name* advanced filtering techniques.

In Release 15.7, this base highlighting was extended with the `rats by layer` command, which allowed you to change the permanent highlight color of nets, and change the visibility of ratsnests for nets, based on the primary routing layer assignments.

With this release, you can supplement the current capabilities with enhanced pin highlighting, making such operations as pin swapping to optimize routability easier. With this feature, you can quickly highlight sets of pins in the design based on characteristics which otherwise would require you to make individual selections in the design. These include pin use, swap code, or padstack

For additional information on highlighting sets of pins, see the [advanced highlight](#) command.

Plotting a Design

The method by which the layout editor plots a design to a plotter or printer differs according to which platform you are on (UNIX or Windows) and which tools you run.

- The layout editors on Windows use Windows Print Manager for controlling printing operations. For information on installing a driver that supports a printer or plotter, consult the Microsoft Windows documentation.

- The layout editor on UNIX uses the `allegro_plot` program, which is based upon the Cadence corporate plotting package, *plotServ*.

Windows does not support `allegro_plot`. If you create an Intermediate Plot (IPF) file, which is a representation of a the tool database, you can copy it to a UNIX workstation that runs `allegro_plot` or to third-party plotting software.

- On either platform, the tool lets you import IPF files or create them for export using the [load plot](#) and [create plot](#) commands, which are detailed in the *Allegro PCB and Package Physical Layout Command Reference*.

See the *Preparing Manufacturing Data* user guide in your documentation set.

Working with Text

You can add, edit, and delete text in a drawing. Text can provide additional information about the design or it can be included as labels that are attached to graphic elements. This section describes:

- Defining text characteristics
- Adding text to drawings
- Editing text in drawings

Defining Text Characteristics

You can define the size and spacing characteristics of text that appears in the drawing. You can assign text parameters to up to 16 text blocks, which makes it easy to specify the appearance of text that you subsequently add to a design. You specify the text parameters as you add the text or label.

For procedures on defining text parameters, see the *Text* tab of the Design Parameter Editor. Use *Setup – Design Parameters* ([prmed](#) command) to access the Design Parameter Editor or right-click in the preselect use model and choose Design Parameters from the pop- up menu.

Adding Text to Drawings

You can use text in drawings as simple notes and as logical labels of elements. Labels include reference designators, device type, value, tolerance, and user part number.

 **Add – Text** ([add text command](#)) does not let you enter an exclamation point (!) in a design database, since `extracta` uses that character as a field delimiter. Be aware of the possible consequences of this condition if you read into the database a file that contains an exclamation point.

Some label commands require not only the data for text location and content, but also the identity of the element to be labeled, such as labeling placement room areas in the layout.

Use the `add text` command to annotate design elements. Use *Layout – Labels* menu selection (in Symbol mode) to add text labels (Ref Des, Device, Value, Tolerance, User Part Number) to symbols.

For procedures on adding text to a design, see *Add – Text* ([add text command](#)) in the *Allegro PCB and Package Physical Layout Command Reference*.

Editing Existing Text or Labels

You can edit text in a drawing. If the text is a reference designator label, editing the text changes the reference designator in the database. This can have other side effects, as explained in this section. You cannot edit a device type label in a drawing, because it redefines the logical structure of the component.

In general, when you edit text, the tool:

- Highlights the text and displays the text cursor on the first character location of the text string.
- Replaces the existing text.
- Lets you select another text string for editing.

 You cannot enter an exclamation point (!) in the layout database, since `a_extract` uses that character as a field delimiter. If an exclamation point is part of existing text that you are editing from an older version of the tool, be aware that `edit text` cannot replace that character if removed.

For procedures on editing text, see *Edit – Text* ([text edit command](#)) in the *Allegro PCB and Package Physical Layout Command Reference*.

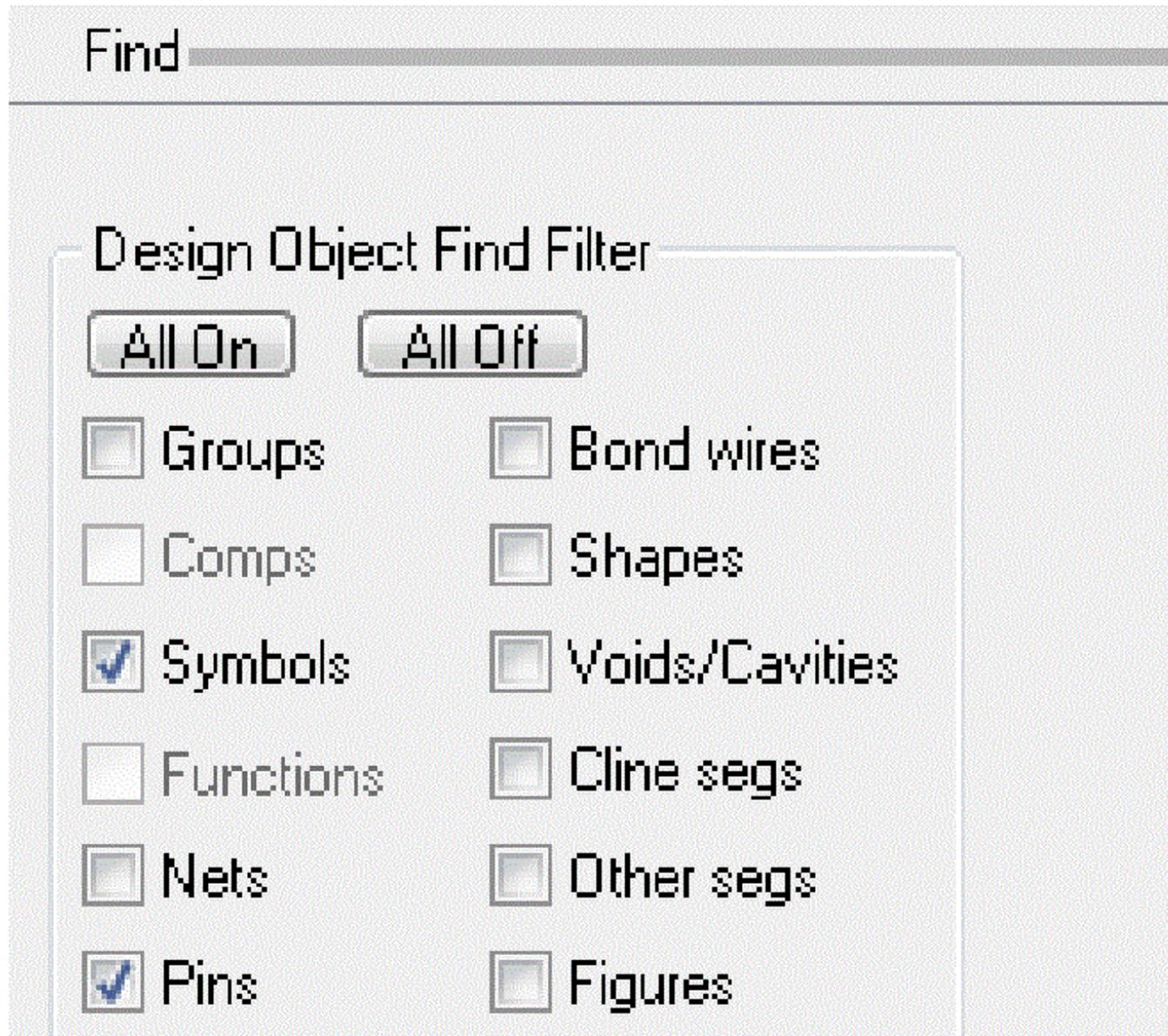
Finding Design Elements

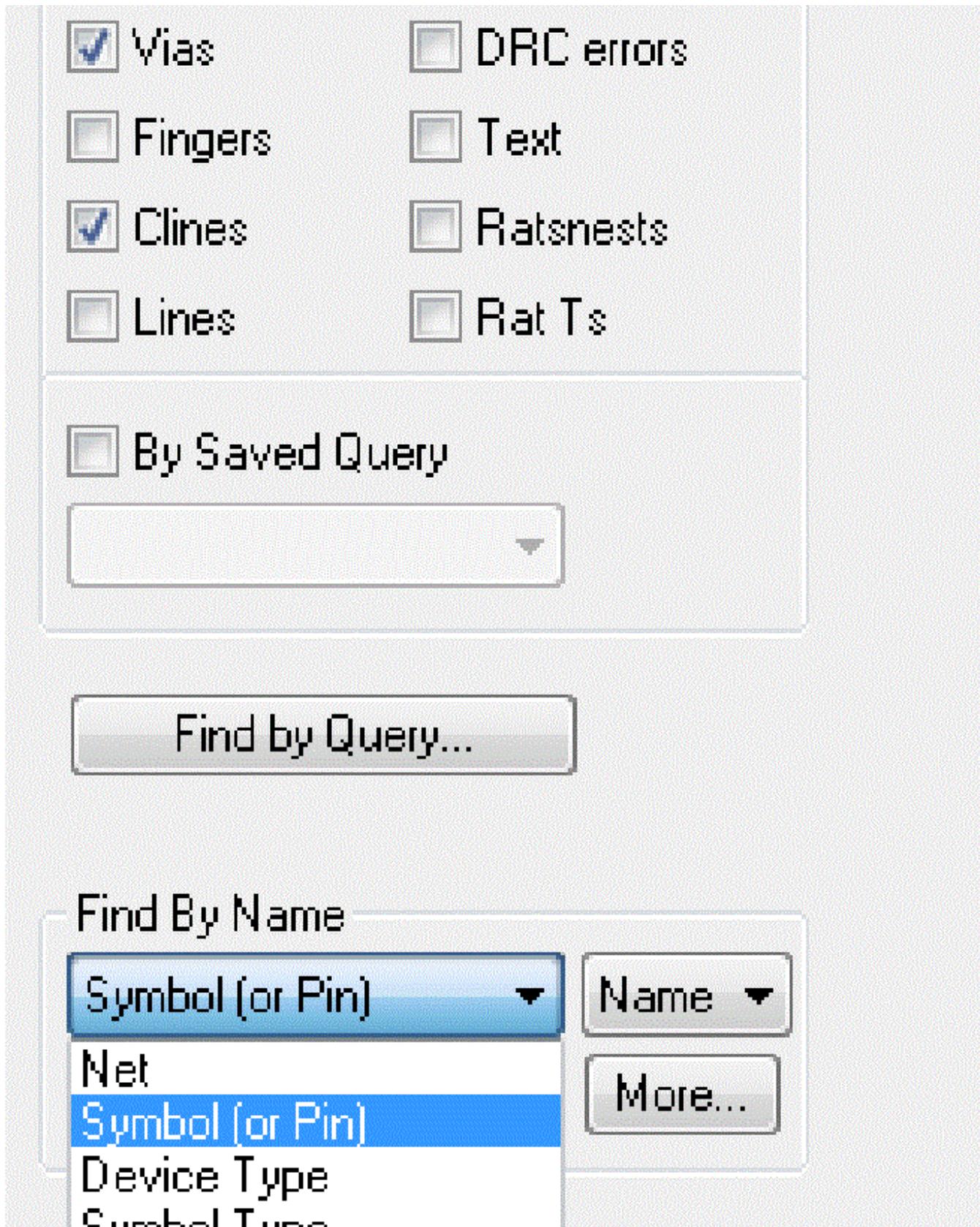
The Find Filter Window Pane

The *Find Filter* lets you specify design elements the active command affects. When you run an interactive command, such as *Edit – Move* ([move](#) command), the *Find Filter* displays the elements the command requires.

In pre-select use model, to refine your selection set and confine your work to a particular element type, such as all nets, you can also right-click and choose the *Superfilter* temporarily to disable the *Find Filter*. When you are using Superfilter, an icon appears in the lower right corner of the status bar.

Figure 3.21: Find Filter Window Pane





Symptom Type
Property
Bus
Diff Pair
Match Group
Module
Net Class
Net Group
Pin Pair
Ratbundle
Region
Xnet
Generic Group

In menu-driven editing mode, the elements in the Find filter available for the active command are in bold text and have their check boxes chosen. The elements available for selection depend on the active command.

You can select or deselect any elements by clicking the check box on or off, or you can select/deselect all the elements with the *All On/All Off* buttons.

If you try to find an invalid element type, the layout editor displays the following message:

<element types> are not selectable at this time.

Name Function Failed.

Determining the Element Selection Hierarchy

The layout database maintains a hierarchy of elements to simplify the selection process. When you choose an element, the tool chooses the highest level element that is associated with that selection. If you disable the higher level elements, such as connect lines or nets, the tool chooses lower level elements, such as line segments.

For example, a pin can be part of a function, net, symbol, component, or group. When determining the proper element to highlight, the tool uses the following hierarchy:

- Groups
- Components
- Symbols
- Functions
- Nets
- Pins

Two primary methods allow you to locate design elements in the tool: *Display – Element* ([show element](#) command) and *Display – Property* ([show property](#) command). Both let you find elements by name or property, but do so in different ways.

Using Show Element

You can use *Display – Element* ([show element](#) command) with the Find filter to locate and identify design elements by property, name or in a list file. You can further refine a find operation by entering a value for the element you want to find. You perform these operations using the Find By Name or Property dialog box.

Finding Elements by Name or Property

With *Display – Element* ([show element](#) command) active, click More in the Find filter to display the Find By Name or Property dialog box, which lists all available object types for chosen elements.

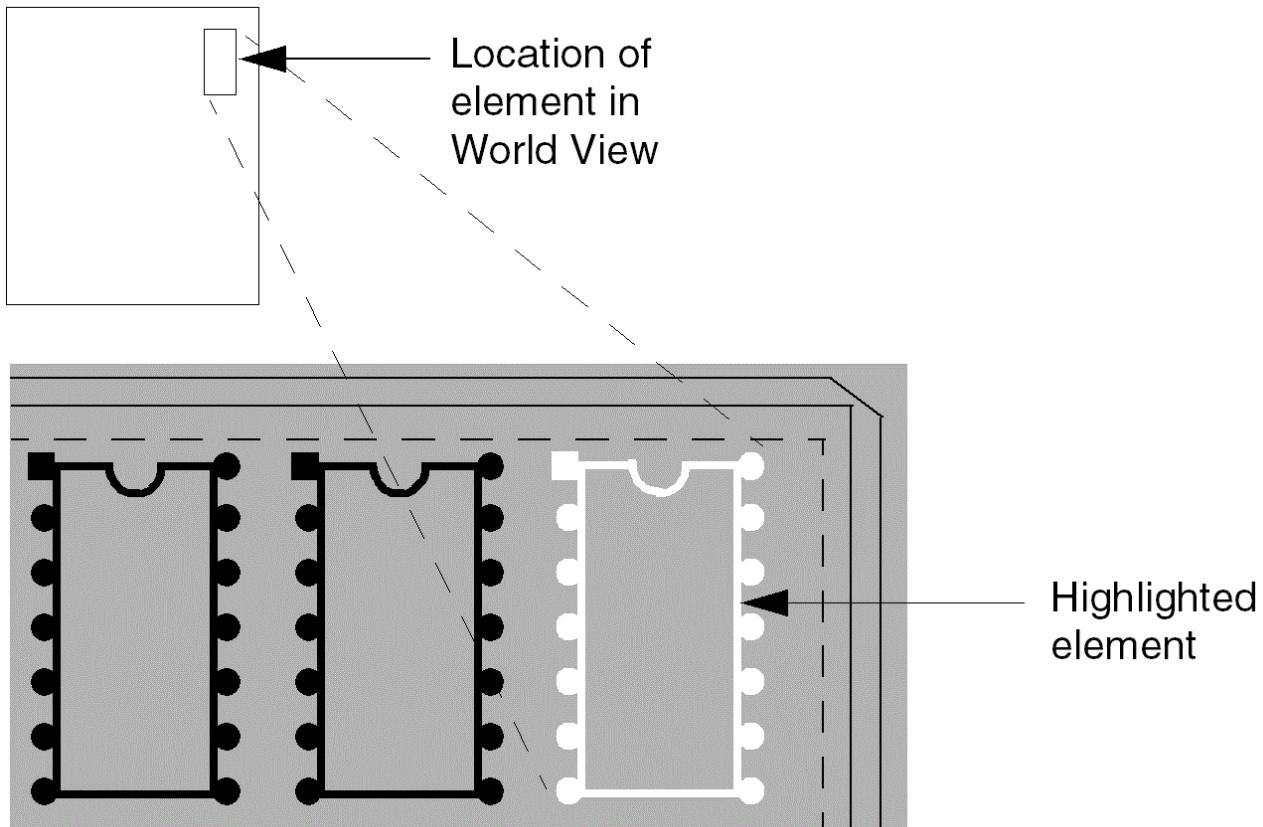
Depending on the object type you select, the Find By Name or Property dialog box allows you to identify an element that you want to find by listing those elements by object type. You can then choose individual items and then by clicking Apply:

- Display the Show Element dialog box on the element(s).
- Display the location of the element(s) in the World View area of the UI.

Highlight the chosen element(s) in the design area

If you know the name of the element that you want to locate (such as U13), you can find it by entering its designation in the Name Filter field and selecting the appropriate object type from the menu.

Figure 3.22: : Example Result of Find by Name



Using Show Property

Unlike *Display – Element* ([show element](#) command), *Display – Property* ([show property](#) command) is not used with the Find Filter, though it can help you locate elements in a design. When you run the command, the Show Property dialog box appears.

By selecting a property (sorted by property or element) and pressing the appropriate Show button, you can display a definition of the property or its value relative to the object to which it is attached. The Name and Value fields let you qualify an element further. When you enter a name or value, the tool searches only for those elements that match both the Name and Value that you entered, and that are valid for the active command.

Using Find by Property from the Console Window Prompt

You can also use the console window prompt to find elements by property. The Find Filter must be activated with elements that allow property assignments.

To use *Find by Property* from the console window prompt:

- At the console window prompt, type

```
find property name <property value>
```

All elements are chosen for the active command that have the defined property name and value.

You can use wildcard characters for both the property name and value. The property name field is not case-sensitive.

Finding by Name from the Command Window Prompt

You can also use the prompt in the command window to find elements by name. You must activate the Find Filter with elements that appear in the design.

When you use the command line at the console window prompt, you can enter character strings, including the element type plus a name or list file, and wildcard characters. Character strings are not case-sensitive.

Table 3-3 lists keywords, the way in which the tool matches that keyword, and an example of each keyword type.

Find by Name Commands		
Keyword	Match	Sample Value
Net	Net that matches name	data1
Symbol(or Pin)	Symbol instance that matches refdes	U34
	Symbol pin that matches refdes.pin	U34.1

Device Type	Component or symbol instances that match device type—components are chosen if the command allows; otherwise, symbols are chosen	74LS74
Symbol Type	Symbol instances that match symbol name	dip14
Property	Property that matches name	NET_SHORT
Bus	Bus that matches name	CON_HDD_BUS
Diff Pair	Diff Pair that matches name	DP1
Match Group	Match Group that matches name	MG1
Module	Module that matches name	MD1
Net Class	Net Class that matches name	CLS_1
Net Group	Net Group that matches name	NG1
Pin Pair	Pin Pair that matches name	U40.3 :U49.4
Ratbundle	Ratbundle that matches name	RAT1
Region	Region that matches name	REGION_1
Xnet	Xnet that matches name	XNET_L1
Generic Group	Generic Group that matches name	GR1

You must enter the keyword exactly as it appears in the drop-down list in the Find Filter. In other words, type `comp` or `symp` instead of component or symbol. If you enter multiple names, put a space between the element names. If the element name contains a space, put quotation marks around it.

For example, the following command selects the nets MEM17, DATA4, and CLOCK for processing.

```
net mem17 data4 clock
```

Likewise, when you enter multiple lists, you must put a space between each list file. For example, the following command selects all components in the files `U.1st` and `R.1st` for processing.

list comp U(.lst) R(.lst)

Using Wild Cards

The tool lets you use wild card characters when you try to find elements by name or by list. Table 3-4 lists the valid wild card characters.

Table 3.4: Valid Wild Card Characters

Wildcard	Match Description	Example
*	Any number of characters	name* = name1, name12, name ANY
?	Any single character	name? = name1, nameA name? ≠ name12, name ANY

Highlighting Chosen Elements

When you select elements by group or window, the tool lets you specify the temporary highlight color. Table 3-5 summarizes the way in which the tool highlights element types:

How the Layout Editor Highlights Element Types	
Element Type	Highlight
Net	All vias, connect lines, shapes, frects, ratsnests, and pins on a chosen net
Component Instance	Symbol instance linked to chosen component instance. Only placed components can be highlighted. Symbols highlighted by symbol and component instance appear the same on the display.
Ratsnest	Chosen ratsnest line
Function Instance	All pins for the chosen function instance

Finding Elements by Using the pick Commands

In addition to using the mouse to highlight elements in a drawing, you can use [pick](#) or [ipick](#) commands to enter x, y coordinates for the elements as described in the *Allegro PCB and Package Physical Layout Command Reference*.

Using Temporary Group Mode

When you run an interactive command in temporary group mode, you can identify elements by name, list, pick, window, or any combination of these until you click right and choose *Complete* from the pop-up menu. Temporary group mode is available only in menu-driven editing mode.

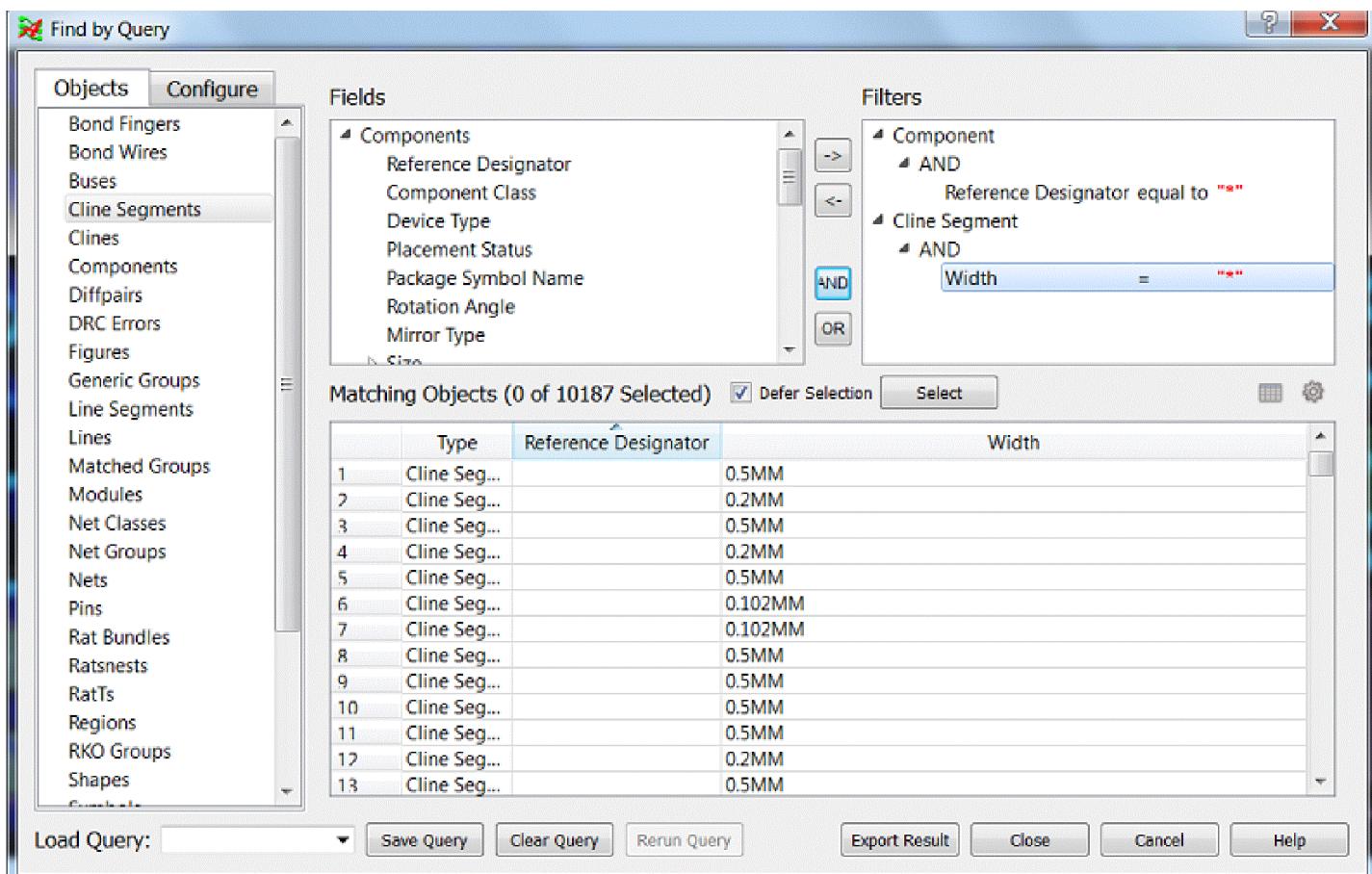
To deselect elements that you select in temporary group mode:

- Press *Ctrl* and click the mouse button.
If you are working with a congested board and multiple elements are chosen by a pick, the elements that you do not deselect go into the reject buffer.

Finding Objects by Query

You can use this method to issue a query to find the specific objects (nets, clines, shapes, voids, and so on) based on their attribute values in the database. The query returns all the objects that match a specified filter or criteria. You can define the criteria in the *Find By Query* dialog box available in the *Find* filter window pane.

Figure 3.23: Finding Objects by Query



The *Objects* tab displays list of different types of objects that can be modified in the *Configure* tab. You can add required objects and their attributes in the *Fields* for creating a query. The *Filters* section defines the query that will be used to extract the information.

The search results are displayed on the fly in the *Matching Objects* section. The *Configure Search Table* icon lets you identify the fields you want to see in the result table.

You have options to save query in a file (.qfnd), clear query, and load query. If inactive for some time, the *Rerun Query* button becomes active to refresh the results.

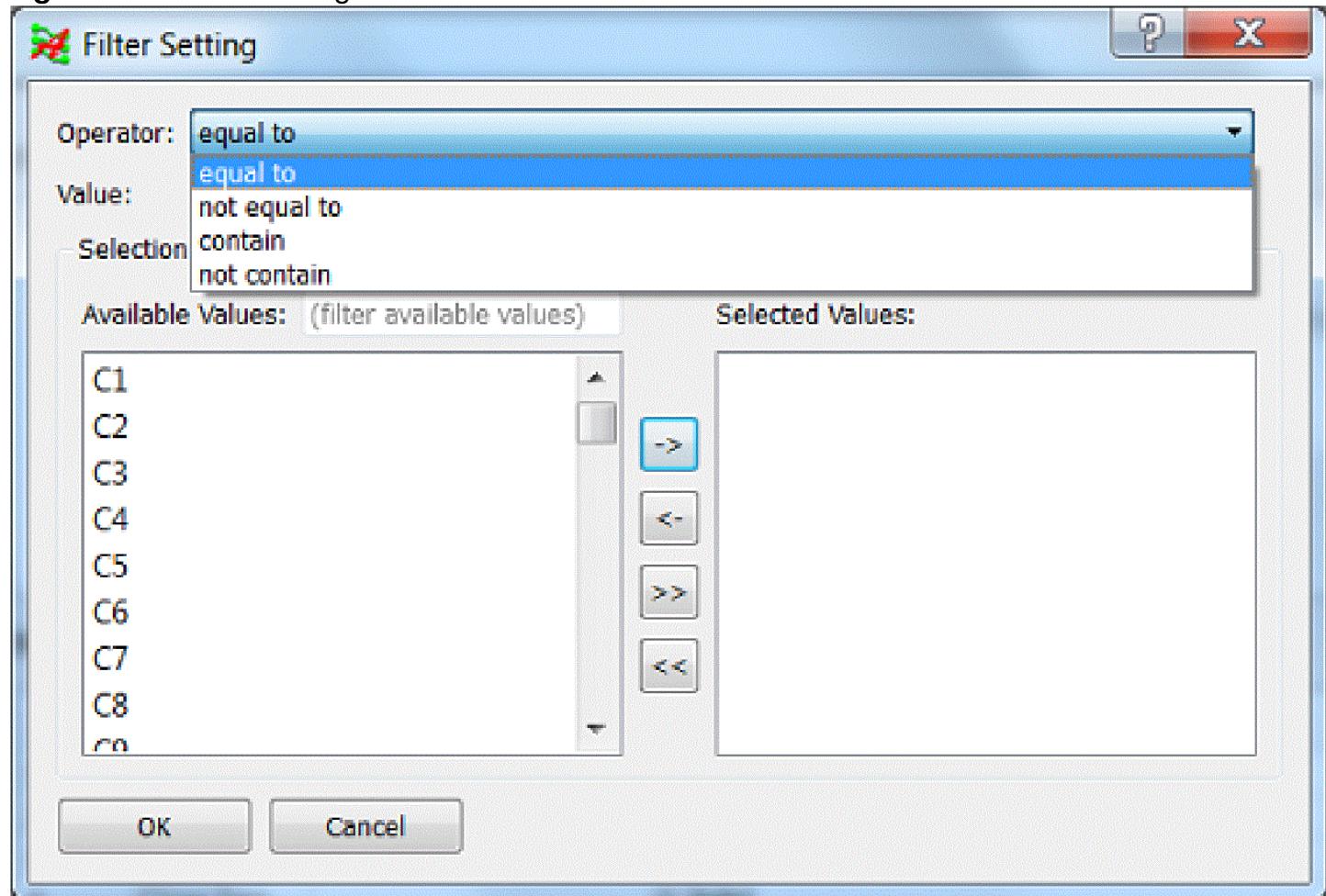
You can export the query in standard XML or CSV formats.

Defining a Query

Choose objects from the *Objects* tab and add them to the *Fields* section by double-clicking or by using drag and drop method. Move the required attributes to the *Filters* section. You can create query based on AND and OR logical conditions. You can specify multiple objects and their attributes for filtering.

In the *Filters* section, the attributes are added with null values representing by "*". Selecting an attribute opens *Filter Setting* dialog box where you can specify its value. You can either directly set values or choose from the list of available values that are present in the database. You can also specify a logical expression for determining the attribute value.

Figure 3.24: Filter Settings

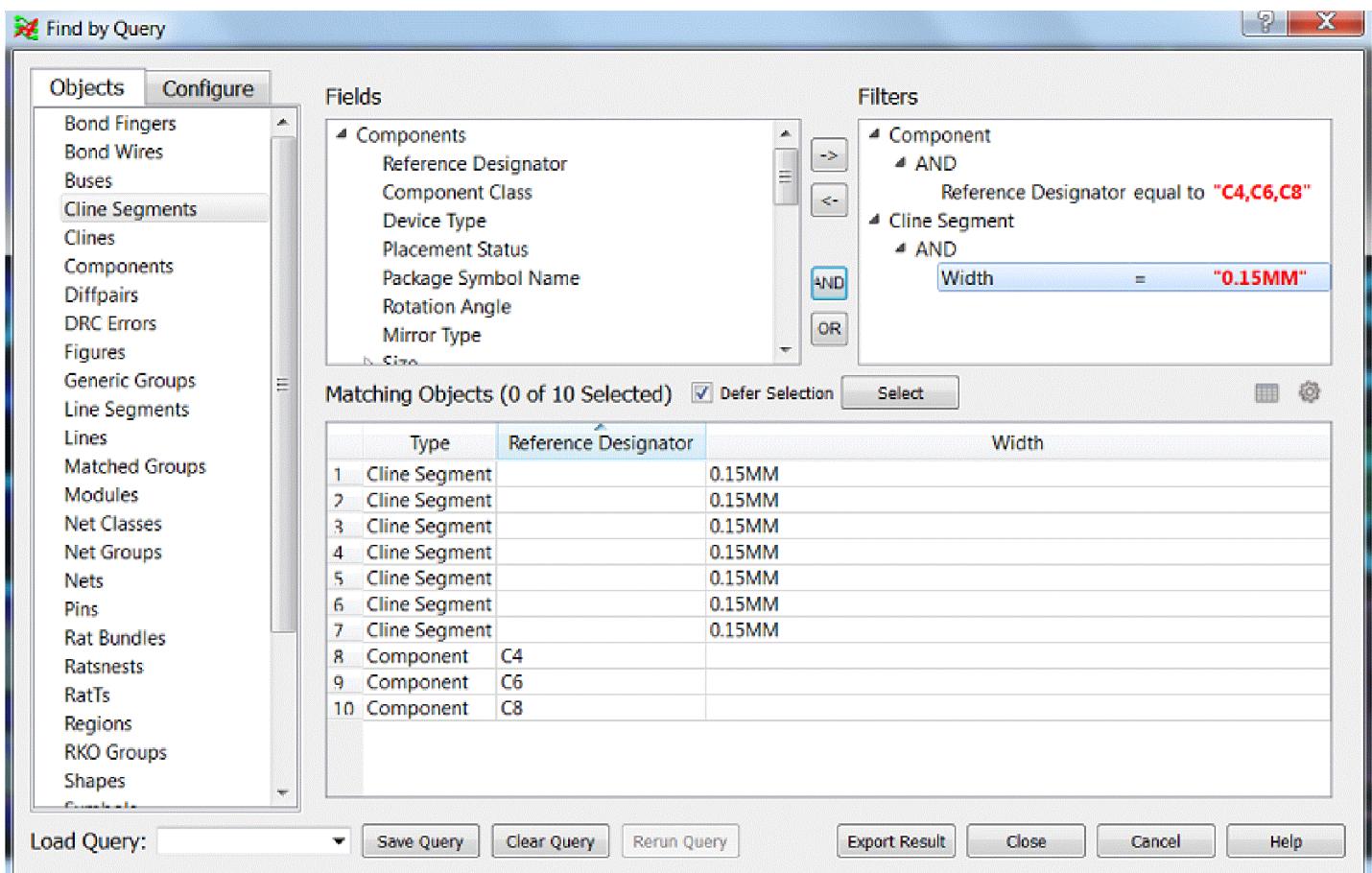


Viewing Matching Objects List

The search results are displayed on the fly in the *Matching Objects* table. The total number of matching results updates based on the changes you made in the *Filters* section.

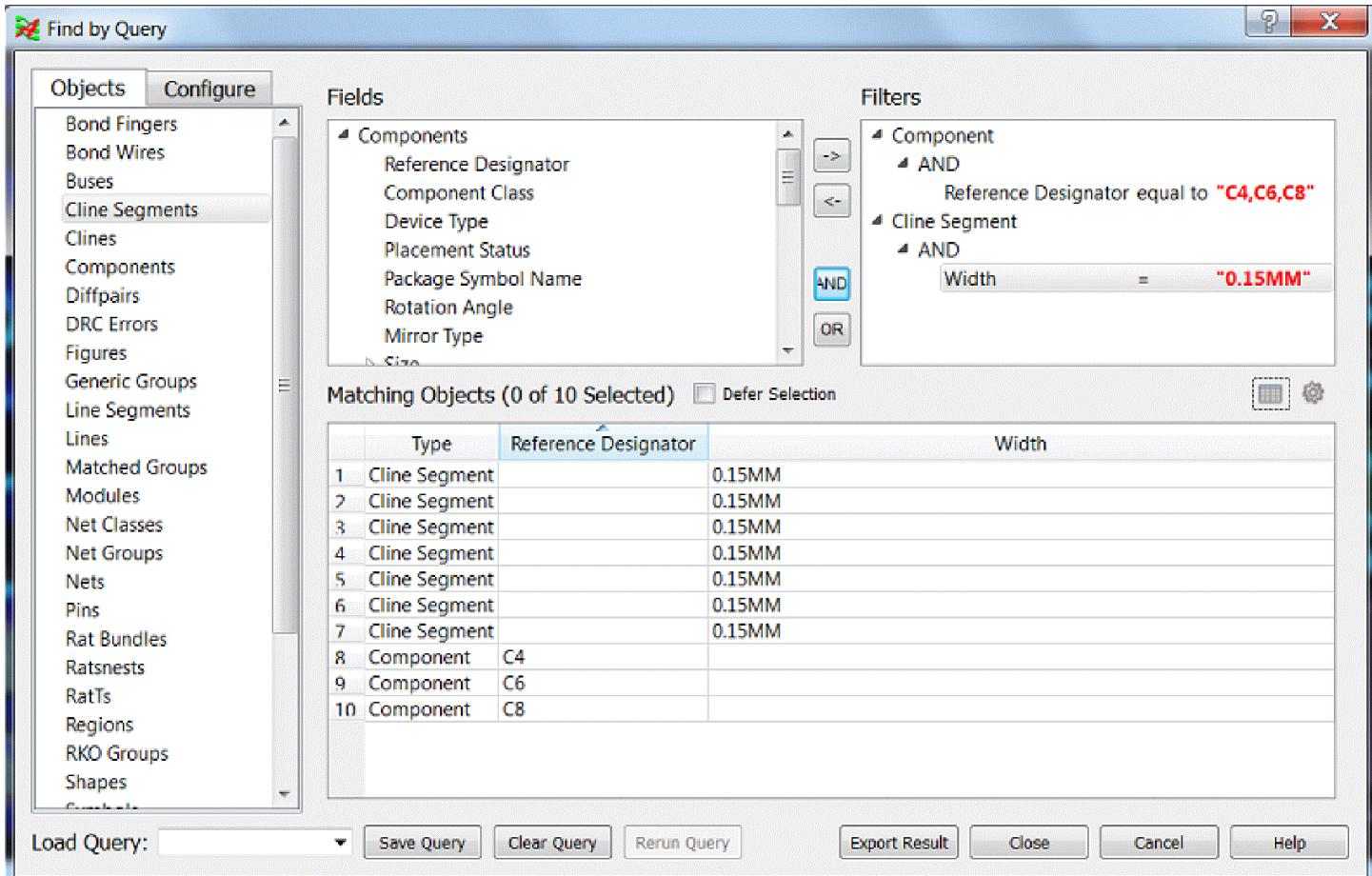
Figure 3.25: Matching Objects List

Getting Started with Physical Design
Using the Layout Editor--Finding Design Elements



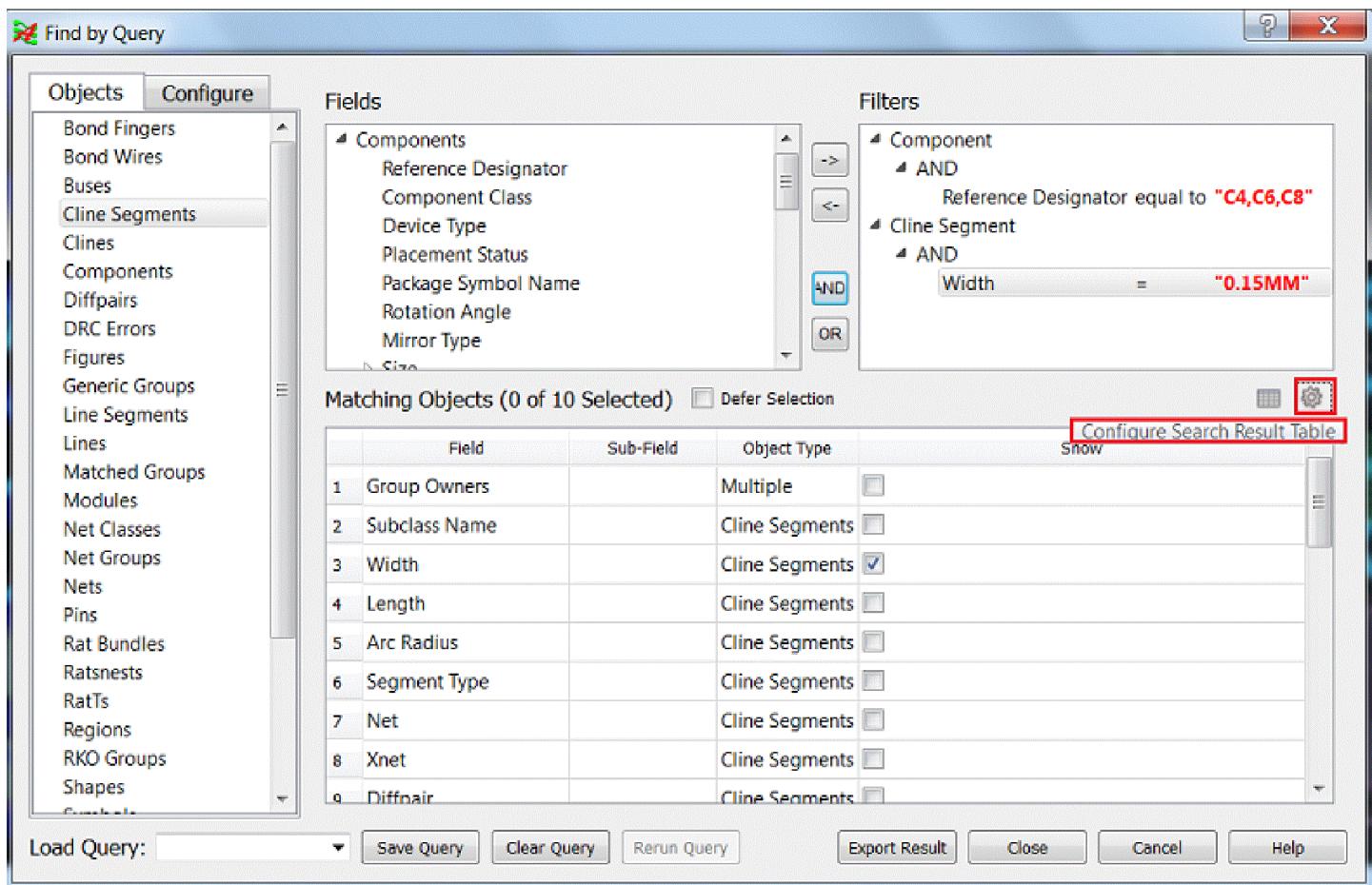
A default set of attributes are always displayed for every object type in the *Matching Objects* table.

Figure 3.26: Default Display of Matching Objects



The *Configure Search Table* icon lets you identify the fields you want to see in the result table. You can enable or disable the corresponding check boxes to show/hide them from the *Matching Objects* table.

Figure 3.27: Configuring Display of Matching Objects

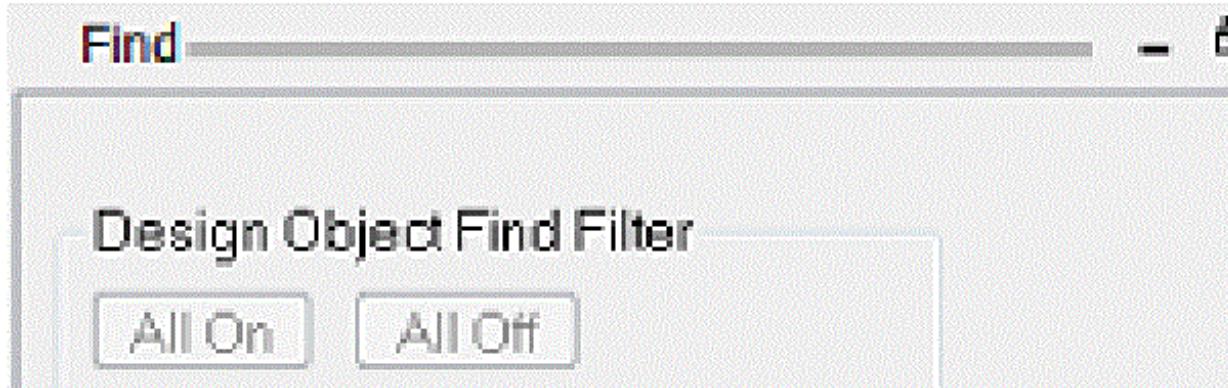


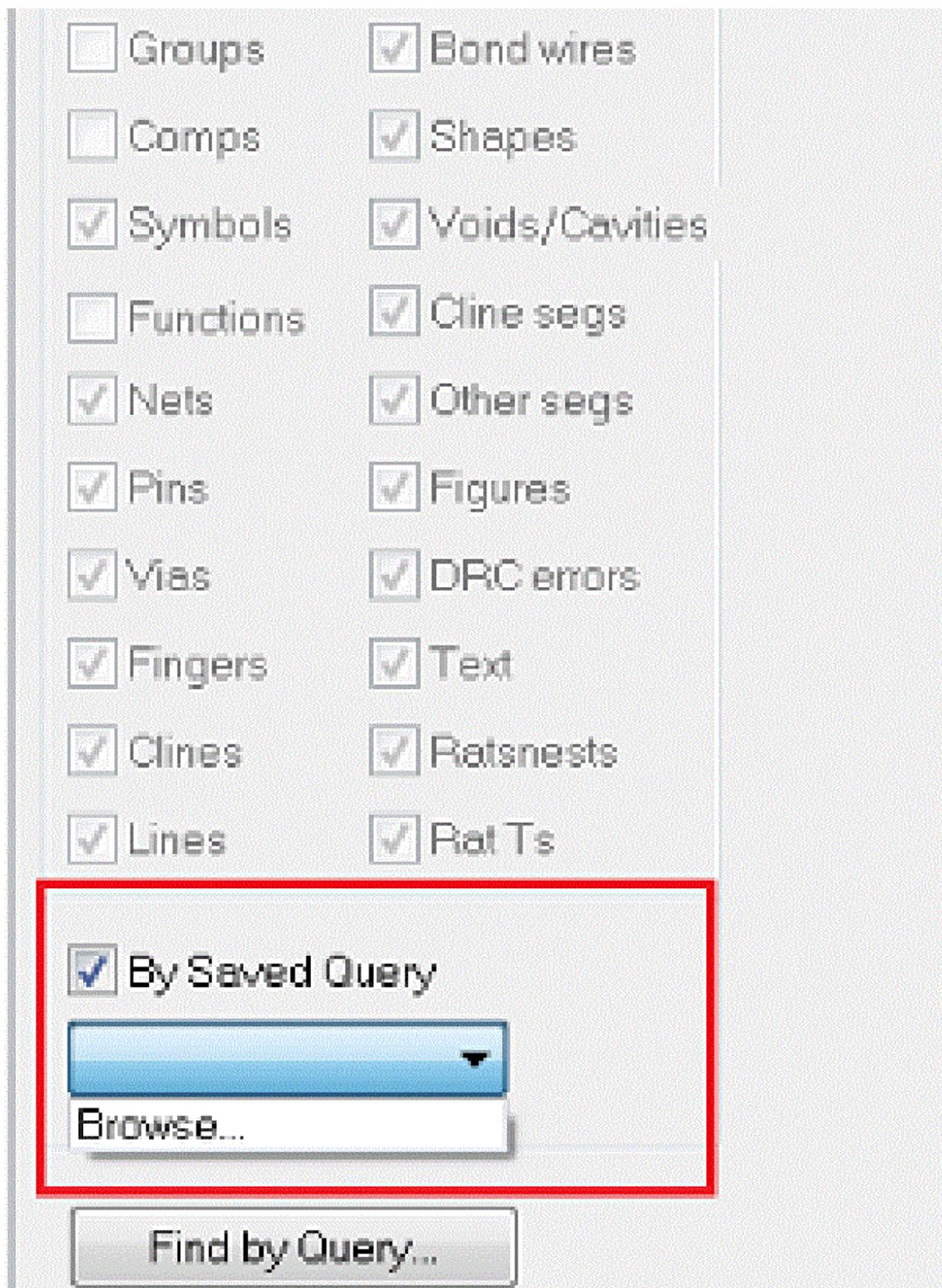
Saving Query

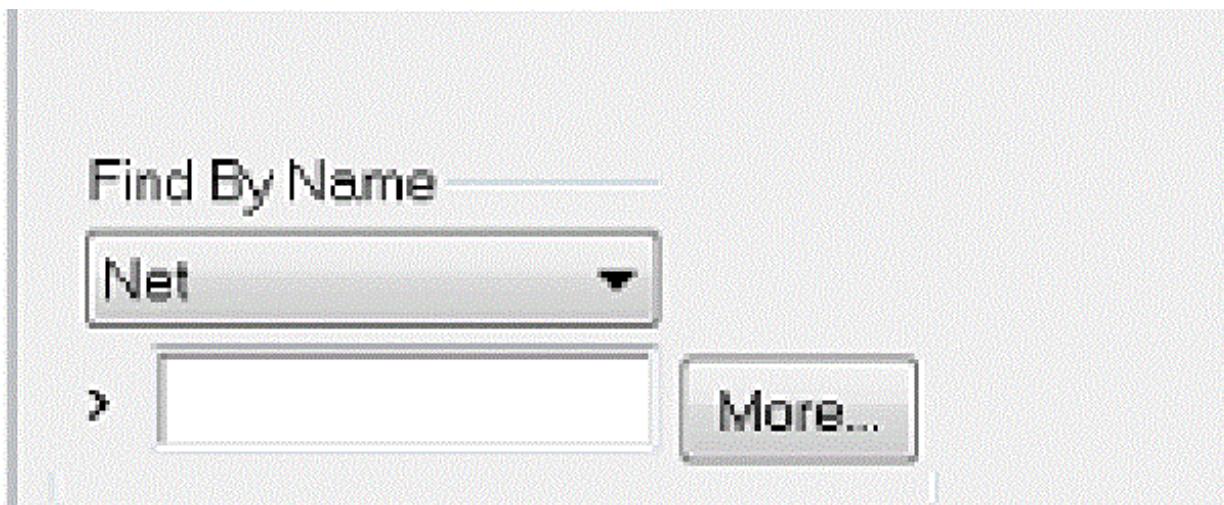
You have options to save query, clear query, and load query. If inactive for some time, the *Rerun Query* button becomes active to refresh the results. You can also export the query in an XML or CSV format.

The current query settings are saved as FNDQ(.qfnd) file. Once saved the query can be executed again by browsing it through the *By Saved Query* option in the *Find Filter*.

Figure 3.28: Running Saved Query







Displaying Matching Objects in the PCB Editor

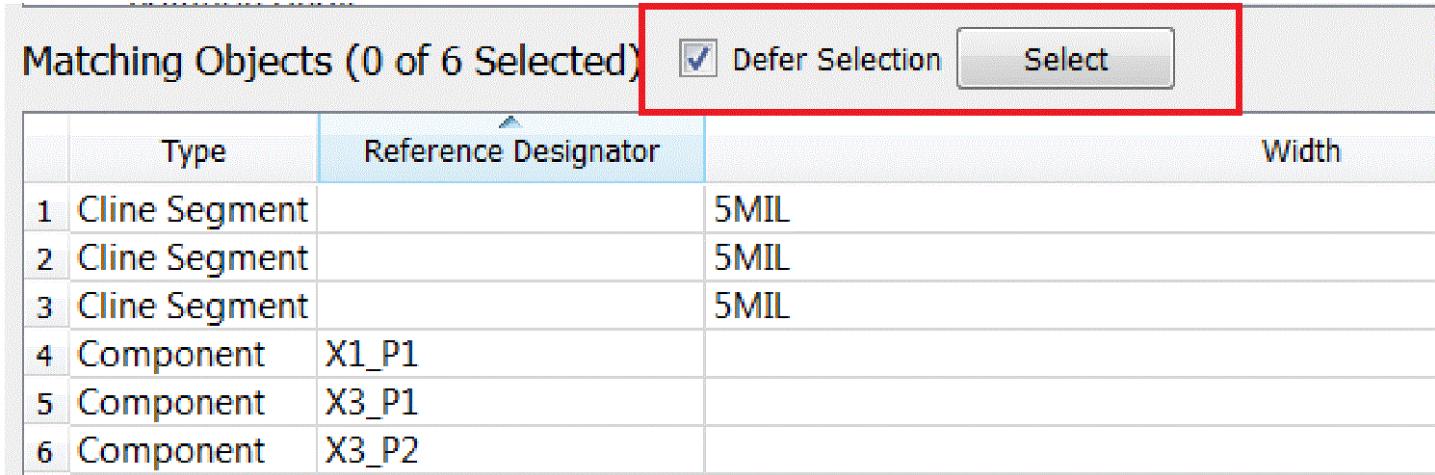
When you click the search result in the *Matching Objects* table, the command:

- Selects the objects to be acted upon by the active command; for example, `property edit`.

⚠ Different commands result in specific behaviors. For example, if you are running `property edit`, the *Edit Properties* and *Show Properties* dialog boxes are displayed; if you are running `move`, the selected objects are selected for editing.
- Displays the location of the objects in the *WorldView* area of the UI.
- Highlights the selected objects in the design area of the UI.

In the result table, select any row right-click and choose *Select All* to highlight all the objects at a time.

If required, you can delay the selection of objects in the design canvas by enabling *Defer Selection* checkbox. A *Select* button becomes available that holds the selection. The selected object is not processed until the button is clicked.



The screenshot shows a table titled "Matching Objects (0 of 6 Selected)". The table has three columns: "Type", "Reference Designator", and "Width". There are six rows of data. The first three rows are "Cline Segment" objects, each with a width of "5MIL". The last three rows are "Component" objects, with reference designators X1_P1, X3_P1, and X3_P2 respectively. At the top right of the table, there are two buttons: "Defer Selection" with a checked checkbox and "Select". A red box highlights these two buttons.

Matching Objects (0 of 6 Selected)		
Type	Reference Designator	Width
1 Cline Segment		5MIL
2 Cline Segment		5MIL
3 Cline Segment		5MIL
4 Component	X1_P1	
5 Component	X3_P1	
6 Component	X3_P2	

Using this functionality you can choose multiple objects from the *Matching Objects* table and view them together or send them as an input to a command in a single step.

Finding Buses in Composer/Allegro Design Entry HDL or System Connectivity Manager and Allegro PCB Editor

When you draw a schematic in Composer, Allegro Design Entry HDL or System Connectivity Manager, you can identify groups of nets as buses. The Find Filter lets you use this bus identification to process nets that are members of the buses. In Composer and Allegro Design Entry HDL or System Connectivity Manager, each net in a bus has a bus name, followed by a number that is enclosed in angle brackets. This number specifies the bit position in the bus. For example, a four-bit data bus can consist of the nets DATA<0>, DATA<1>, DATA<2>, and DATA<3>.

Identifying Buses

When you choose *File – Import Logic* ([netin](#) command) and choose *Design entry HDL* from the Import Logic dialog box, each bus is assigned a **BUS_NAME** property and value that matches its net name. For example, in the bus described in the preceding section, each net receives a **BUS_NAME** property with DATA as the assigned value.

The net name assigned is the original bus name plus the associated number without the angle brackets. For example, the corresponding tool net names for the four-bit data bus are DATA0, DATA1, DATA2, and DATA3.

This association between the net name and the bus name lets you use the Find by Name function to identify the net and by using *Edit – Properties* ([property edit](#) command) to add the **BUS_NAME** property interactively.

Bus Selection Syntax

You can specify designated bus nets on the command line in the command console window or, if you choose *Nets* in the Find Filter, in the Name field.

To specify a group of nets in a bus:

- Enter the bus name and a bit subscript field using the following formats:

<bit>	Specifies a single bit of the bus. For example, DATA<3> defines net DATA3.
<bit1:bit2>	Specifies a subrange of bits. For example, DATA<3:1> defines nets DATA3, DATA2, and DATA1. (The order of this subrange does not matter; DATA<3:1> is the same as DATA<1:3>.)
<bit_list>	Specifies a list of bit subscript fields that can have either of the preceding formats. Separate each list with a comma. For example, DATA<1:3,7,10:12> defines bits 1, 2, 3, 7, 10, 11, and 12.

In each of these formats, angle brackets delimit the bit subscript field; the *bit* variable specifies a bit number and must be an integer greater than or equal to zero. If you leave the angle brackets empty, the tool chooses all nets of the bus. To choose bus members, the bus name must match the net name and bit number exactly.

The following command chooses the DATA1, DATA3, DATA4, DATA5, DATA6, and DATA7 nets for processing.

```
net data<1,3:7>
```

In addition, you can assign a BUS_NAME to nets that do not have a bit number in the name or that match the bus name, but that can be found by using the busname<> syntax. For example, if you assign the BUS_NAME property DATA to the DATA0, DATA1, DATA2, and DATA3 nets and enter the following command in the Name field, you select all the nets.

```
net data< >
```

Using Buses

The following menu selections/commands accept bus names:

- *Display – Highlight* ([hilight](#) command)
- *Display – Dehighlight* ([dehilight](#) command)

- *Display – Element* ([show element](#) command)
- *Display – Property* ([show property](#) command)
- *Edit – Change* ([change](#) command)
- *Edit – Delete* ([delete](#) command)

You can also use the select by bus name option to expedite the following operation:

- Highlighting the bus nets
- Assigning placement weights to a bus by defining the WEIGHT property on bus nets
- Routing buses before the other nets by setting the ROUTE_PRIORITY property on bus nets

Highlighting and Dehighlighting Design Elements

The layout editor lets you highlight and dehighlight eligible elements to accentuate and easily locate them in the design canvas with *Display – Highlight* ([highlight](#) command) and *Display – Dehighlight* ([dehighlight](#) command).

Another means of highlighting elements is *Display – Assign Color* ([assign color](#) command). You can quickly assign both custom color and highlighting to an element without requiring the use of the *Color* dialog box and *Display – Highlight*. Changing the color or highlighting with this command automatically updates the color and highlighting information in the *Nets* section of the *Color* dialog box as well.

These three commands also function in the pre-selection use model, in which you choose an eligible element first, then right-click and execute the command.

Automation of Design Tasks Using Scripts and Macros

If you find yourself repeating certain design tasks on a regular basis, you can create scripts and macros to automatically perform those tasks.

While you can use both scripts and macros across multiple drawings, scripts always start and end at the same coordinate, whereas a macro lets you start at a different coordinate each time you use the macro. Every action included in the macro takes place relative to the starting point.

Scripts are useful when performing repetitive tasks such as setting up fields in dialog boxes, adding elements to multiple databases at the same location, and duplicating drawings.

For information about procedures for using scripts, see *File – Script* ([script command](#)) in the *Allegro X PCB and Package Physical Layout Command Reference*.

Using Environment Commands with Scripts

You can modify the behavior of script recording and replaying through the use of environment commands entered at the console window prompt.

For information on using environment commands in scripts, see the [ifvar](#) and [ifnvar](#) commands in the *Allegro X PCB and Package Physical Layout Command Reference*.

Using Data Browsers

Data browsers are dialog boxes that present elements of the type required by the current command. You can select elements listed in a data browser, but you cannot delete, rename, or otherwise control the type of data displayed. Data browsers list all named elements in a design or within libraries outside the design, based on parameters that you set in the dialog box.

Displaying Quickview Information

Data browsers support quick views of the database elements that you select from the list in the dialog box. Quickview lets you see a graphic preview of a database or a selection of the properties that make up the database.

 Graphics are not available for padstacks; only text regarding the padstack name, type, units, accuracy, and geometry is available.

Supported databases include the following file types:

.brd	.bsm
.mcm	.dra
.osm	.mdd
.ssm	.psm
.fsm	.dfa

.pdf

File browsers that open scripts, logs, and other text files do not support quickviews.

 Older databases must be upreved to version 14.0 (or subsequent versions) with Qvupdate before you can display them in quickview.

By selecting one of the two quickview buttons, you can view different data associated with the selection:

- *Text*

The Text button displays text information, such as the information for a package symbol.

Name: SSOP28

Type: Allegro Symbol

Units: MILS

Accuracy: 2

Pins: 28

- *Preview*

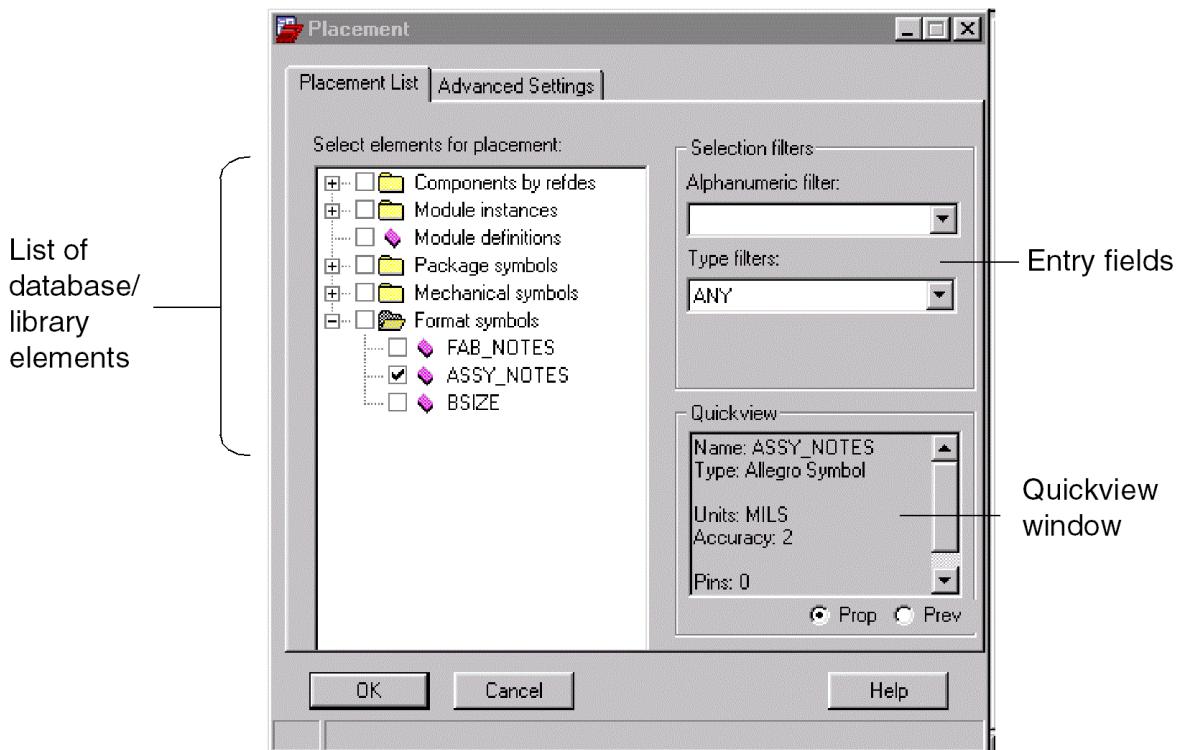
The preview button displays a simple graphic of the database, the image of which depends on the type of database you are viewing.

- Quickviews of .brd and .mdd databases display a board outline, package keepin, or a rectangle of the drawing extents, and a chosen set of the largest pin-count components in the database.
- Quickviews of symbols display a symbol outline and the number of pins on the symbol. If the symbol contains a large number of pins, the quickview does not display all of them. (But that information can be derived from the text view.)

 Try opening and saving symbols that do not show quickviews.

Figure 3-19 shows the data browser that opens when you choose *Place – Manually* ([place manual](#) command) and a quickview of the properties of the chosen object. The title bar reflects the object type you are browsing.

Figure 3.21: Data Browser for Manual Placement



If Quickview cannot display the preview or the properties of the element, a "Not Available" message appears in the quickview window.

Using Qvupdate to Display Quickview Information

This stand-alone program lets you update footprint information in design (.brd), drawing (.dra), padstack (.pad), or module (.mdl) databases that were created prior to release 14.0 so that text and graphics associated with them can be displayed in the Quickview window of file/library browsers. Without running Qvupdate, such information can be displayed in Quickview only by opening the pre-14.0 database in the editor's graphic environment and replacing the database using *File – Save (save* command). Qvupdate lets you update the footprint information for all pre-14.0 libraries in one operation through the use of the * wildcard character.

⚠ Qvupdate does not update symbols; you must update corresponding .dra files. Qvupdate automatically generates symbols from the .dra file.

Note the following conditions:

- Saving pre-14.0 databases in batch mode does not update the footprint information.
- Running uprev does not add the Quickview data to a layout database.

- Databases that were created prior to release 13.0 may have to be upreved before running Qvupdate.

For procedural information on using qvupdate see the [qvupdate command](#) in the *Allegro PCB and Package Physical Layout Command Reference*.

Database and Library Selections

In default mode (Database), data browsers list all the elements in the design's database. You can also view all named elements in the editor libraries when you check *Library*. The elements listed in Library mode may sometimes include items already in the design. This is because database items remain displayed in the list box when the library option is checked.

If an object in the database has the same name as an object in the library but contains different content, the database object takes precedence in the data browser; that is, the database object is chosen.

When you check the *Library* option, it reopens in Library mode for the duration of the design session, or until you de-select the library option.

To choose a database object:

1. Choose an application that prompts you for data by opening a data browser. (Specific instances are covered in the appropriate sections of this user guide.)
2. If the object you are looking for is not listed in the design's database, click *Library* to get a listing of all elements in the library.
You can filter the elements displayed in the list box by typing a string (partial object name) and a "wildcard" character in the field. For example:

- Type FLAT* to display all object names that begin with FLAT.
- Type FLAT*x to display all object names that begin with FLAT and end with x.
- Type FLAT ?, where ? represents any single character.

Data browsers remember filters that you enter in the field. They can be reviewed by clicking the arrow button to the right of the field.

- Highlight a filter by clicking on it or by using the up-arrow/down-arrow keys on the keyboard.
 - Close the filter history menu by clicking the arrow button.
3. Select the object name you want to place in the design using one of these methods:

- Choose the object name.
The object name is highlighted and appears in the field.
 - Type the object name in the field.
The data browser searches the design database, then the library files for the object. If the name you are looking for is in the library, the *Library* check box turns on to indicate the object's location.
 - Double-click on the object name.
The object is chosen and the data browser closes.
4. Do one of the following:
- Choose *OK*.

The data browser closes and the chosen valid object is ready to be placed in the design. (OK does not close the browser until a valid object name is chosen.)
or

- Choose *Cancel* to close the data browser without placing an object.

Using Strokes and Associated Commands

You can run certain commands using predefined patterns of mouse strokes that you draw in in the Design window. The layout editor interprets the pattern as a command and executes the command when you complete the stroke.

You can use the layout editor default `allegro.strokes` file located in the `$cdsroot\share\pcb\text` directory or you can create your own file using the Stroke Editor.

The layout editor looks for `.strokes` files in this order:

1. Current working directory
2. `\pcbenv` directory
3. `$cdsroot\share\pcb\text` directory

If you create a new `.strokes` file, store it in your current working directory or in the `\pcbenv` directory. If you do not create a new strokes file, the Stroke Editor places a copy of the default `allegro.strokes` file in your `pcbenv` directory.

To create a `.strokes` file, or edit an existing `.strokes` file, see the [stroke editor](#) command in the

Allegro PCB and Package Physical Layout Command Reference.

Default .strokes File

The following table shows the strokes and associated commands in the default `allegro.strokes` file.

Using the default `allegro.strokes` file, you can:

- Execute the `zoom world` command by drawing a W stroke anywhere on the design.
- Zoom into an area of a design by drawing a Z stroke in the specified area of the design.
- Move, copy, and delete by drawing the M, C, and D strokes respectively. The stroke selects the object under the first point of the stroke, shown here as circles in the patterns.



For more information on the commands listed in this section, see the appropriate sections of the *Allegro PCB and Package Physical Layout Command Reference*.

Running Commands Using Strokes

To run commands using strokes:

1. In the Design Window, place the cursor over the object you want to move, copy, or delete, or over the area you want to zoom into. (You can draw the `world` command anywhere in the Design window.)
2. Press and hold down the `Control` key and the right mouse button *at the same time* to make a stroke.
As you move the mouse, you see the pattern being drawn.
3. When the stroke is complete, release the right mouse button.
If the layout editor recognizes the stroke, the associated command runs. If it does not recognize the stroke, the layout editor displays the following message:

Stroke not recognized.

You must enter strokes in the same direction in which they were created either in the default `.strokes` file or a customized file. This means that if you are creating your own `.strokes` file,

you can have two strokes that look the same but issue different commands.

For example, if two strokes appear as diagonal lines, one can represent the `vertex` command, and the other the `delete vertex` command. The difference is that one stroke is drawn from upper left to lower right and the other from lower left to upper right.

- ⚠** You can set the `no_dragpopup` environment variable by choosing *Setup – User Preferences*. By default, you must hold down the `Ctrl` key and depress the right mouse button at the same time when using strokes. Setting this environment variable lets you depress the right mouse button and drag the mouse when using strokes. With this option, however, you lose the ability to choose popup menu items by pressing the right mouse button and dragging the mouse. Instead, you have to click twice with the right mouse button: once to see the popup and a second time to select a popup item.

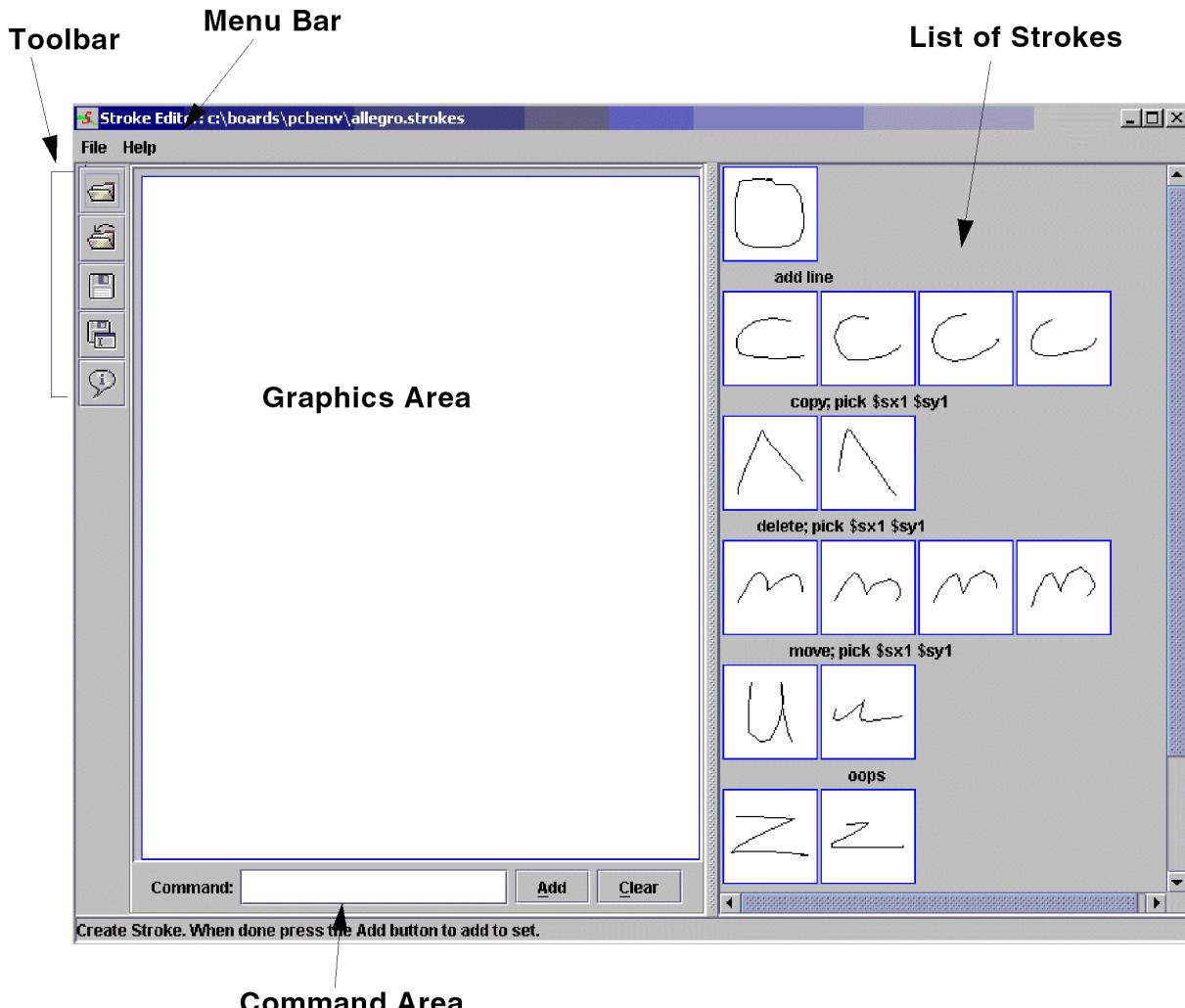
To specify a file containing your own strokes instead of using the default strokes file, see the [strokefile](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

The Stroke Editor

In addition to using the default `.strokes` file that shipped with the software, you can create stroke definitions with associated commands using the Stroke Editor. You can store these files on your system.

Figure 3-20 shows the Stroke Editor.

Figure 3.21: Stroke Editor



The Stroke Editor has these features:

- **Menu Bar** – Located below the title bar, the menu bar provides options for opening, closing, and saving files, saving a file with another name, and getting help for the layout editor and the Stroke Editor.
- **Toolbar** – Located on the left side of the Stroke Editor window, the icons provide the same options as the menu bar items.
- **Graphics Area** – Located on the left side of the window, the Graphics Area is the white portion of the window where you can draw a stroke.
- **Command Area** – Located below the Graphics Area, the Command Area lets you enter a command and associate it with the stroke shown in the Graphics Area. You can also clear existing strokes in the Graphics Area.

- **List of Strokes** – Located at the right side of the Stroke Editor window, the List of Strokes includes all the strokes and associated commands in the file.

 For the `move`, `copy`, and `delete` commands, a notation states *pick \$xs1 \$sy1*. This means that the stroke selects the object under the first point of the stroke.

To create a stroke file, or edit an existing stroke file, see the [stroke editor](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Defining Aliases

The alias feature lets you define a command vocabulary and create shorthand for commands you use most often. You can also program function keys (on most keyboards) to execute commands to increase speed and ease of work.

The alias is an alternative way of entering the command, but it does not disable the full commands. You can still use the standard form of the command.

This section describes how to establish an alias for typed entries and for function keys. Note that aliases work only in the layout editor, not at the operating system level.

A command alias entered at the command prompt is active only for the current work session. When you exit the layout editor and return to the operating system, aliases are lost.

To use command aliases repeatedly:

- Define and save them in a local environment file as described in [Managing Environment Variables](#).

Some default command aliases are provided with the layout editor. The sample global environment file lists the default aliases for the function keys and for the typed commands.

 `a` is used as an alias for `alias`.

You have several options at the keyboard. You can:

- Use standard commands.
- Use the default aliases.
- Define aliases for personal use.
- Define temporary aliases for an individual work session by entering the alias command at the

console window prompt.

- Establish aliases in a local environment file that remain in effect at every login until you change the environment file.

For information on creating aliases, see the [alias](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

For information on deleting aliases, see the [unalias](#) command in the *Allegro PCB and Package Physical Layout Command Reference*.

Managing Environment Variables

This chapter describes how you can use environment variables to set operating conditions at the local and site levels. It includes a description of the global environment. It also describes how to set user-defined variables. Included in this chapter are these topics:

- [The Global Environment File](#)
- [Setting User-Defined Variables](#)
- [Setting Project Level and Site Customization Variables](#)

ⓘ The layout editor uses environment variables specified in the local `env` file or in the interactive User Preferences Editor dialog box to set padstack, footprint, and other search paths. If you define search paths such as `PADPATH` in Allegro Project Manager rather than in the layout editor, you must use Project Manager to launch the layout editor to locate the specified files.

The Global Environment File

The layout editor provides a global environment file during installation. This ASCII text file contains system and configuration information critical to the operation of the software in the form of variables and aliases.

The global environment file (`envenv`) resides in the the layout editor install directory in:

```
$allegro_install_root\share\pcb\text
```

The layout editor looks for the `env` file in this location on startup and, if not found there, generates an error message.

ⓘ Do not move the `env` file nor copy or modify the contents. Changes made to the file will be lost if you reinstall your layout editor or if you upgrade the software. See [Setting User-Defined Variables](#) for details on how to customize local environment variables.

Variables

The layout editor uses other configuration variables to locate system files for menus, forms, and messages. For example:

```
set alibpath = . D:\PCBENV\share\PCB\pcb_lib
```

In addition to these configuration variables, the global `env` file also contains the Cadence default library search path variables that determine how the layout editor searches for various types of files, for example: symbol, device, and help files. Typically, these paths are modified. For information on modifying these variables, see [Setting Project Level and Site Customization Variables](#), and [Setting User-Defined Variables](#).

-  In either case, do not move these directories without making the appropriate changes in the path variables or the layout editor generates errors and will fail to locate information.

Library Path Variables

The global environment file contains the library search paths to all the libraries that are provided with the layout editor. In a local environment file, you can add or modify environment variables that define custom library search paths; for example, to locate component libraries for specific design projects. This procedure is explained in more detail in [Defining Library Path Variables in a Local env File](#).

System Variables

The `env` file controls the appearance and behavior of the layout editor through variables that modify graphics displays, control automatic save functions and plotting, allow file versioning, influence glossing, change the contents of backannotation files, and perform other functions. However, not every variable is included in the installed `env` file. See [Setting User-Defined Variables](#) for a list of variables you add at the local level.

The following variables should not be used in user-defined programs that are not going to be used for the layout editor-specific applications:

- CDS_SITE
- ALLEGRO_SITE
- TELENV
- ALLEGRO_INSTALL_DIR

- ALLEGRO_INSTALL_TOOLS
- ALLEGRO_INSTALL_ROOT
- ALLEGRO_TYPE
- __UNIX (if UNIX)
- _PROGRAM
- HOME
- LOCALPATH
- LOCALENV

Setting User-Defined Variables

User-defined variables let you add or modify certain behaviors to the layout editor. Variables can be set at the local level to provide you with pathways to individual project directories and associated libraries, accommodate individual display preferences, or set certain behaviors.

To set user-defined variables, use one of these methods:

- Modifying the local env file
- Setting variables at the console window prompt
- The User Preferences Editor

Modifying a Local env File

1. Locate your `env` file in the `pcbenv` directory.
2. Open the file using an ASCII text editor.

```
source $TELEENV

### User Preferences section

### This section is computer generated.

### Please do not modify to the end of the file.

### Place your hand edits above this section.

##
```

The pound sign as the first non-whitespace character tells the system to ignore the information on that line. Comment lines can be inserted anywhere in the file.

-  Do not modify line one of the local env file. Line one contains the source command which tells the system to read and execute all the information in the global environment file.

Any data that you enter after line one in the local file becomes part of the "instruction set" of the the layout editor software. Entries before the commented section are permanently saved in the file.

The location of information in the local file is pertinent. Do not insert text below the commented section. This area is reserved for User Preferences Editor insertions. These insertions override values located above the commented section.

3. When you have finished adding or modifying variables in the local env file, save your changes and close the file.

Figure 4-1 illustrates a modified local env file.

Figure 4.1: Modified Local env File

```
source $TELEENV
set alibpath = . D:\PCBENV\share\PCB\pcb_lib
alias F4 cancel
### User Preferences section
### This section is computer generated.
### Please do not modify to the end of the file.
### Place your hand edits above this section.
##
```



Insert lines here to modify
the env file and save the
changes in the file.

⚠ For Windows users only: if a variable path value contains a space, the path strings must be enclosed with quotation marks. For example, `set psmpath= ."/symbols smt"`

Defining Library Path Variables in a Local env File

When you create a new library, you can enter a library path variable in your local environment file that accesses that library instead of the default libraries provided with the layout editor.

The pathname is a directory search list. The layout editor looks for data in the order listed in the path. For example:

```
set PSMPATH = . symbols .. ./symbols $LIBPATH/symbols
```

defines a search path that looks for the required directory (symbols) in the current working directory. If there is a symbols library, the layout editor accesses it for symbols as needed. If there is no symbols library in the current working directory, the layout editor continues to look in the next directories higher up. If no user-created symbols libraries are found, the layout editor uses the installed symbols library.

1. Place project directories containing custom libraries in a location other than the software-installed libraries.
2. In your local environment file, enter the new library search path variable.

Example: You have created a custom symbols library for a project, and have placed that library in a directory called `sym_pro1` in the current working directory. In your local `env` file, under the line that sources the global `env`, add this line:

```
set PSMPATH = ./sym_pro1 .. ./symbols $LIBPATH/symbols
```

The layout editor searches this path for the symbol library instead of the symbols path name in the global `env` file.

Setting Variables at the Console Window Prompt

You can override variables and aliases in your local environment file by entering `set` commands at the console window prompt. Commands that you set here remain in effect for the duration of the current work session.

The `set` Command

The `set` command is one way that you can define or replace an environment variable for the current session. Only environment variables that have immediate effectiveness apply to the current session. For more information about when variables become effective, see the User Preferences Editor dialog box. To set these environment variables permanently, either use the User Preferences Editor dialog box or manually edit the local `env` file.

The syntax for the `set` command is:

```
set variable_name = value(s)
```

 Methods for setting environment variables vary according to the shell you are using. If you are using `csh`, for example, you can set variables using the `setenv` command. If you do not know what shell you are using, refer to your operating-system documentation or see your system administrator.

A simple example is setting the database to save your work automatically every 30 minutes. In your local `env` file:

```
set autosave_time = 30
```

 An autosave only occurs if you make changes to a design; if you open a design without editing it, no autosave occurs. Nor does an autosave occur while a command is active. If a command is active when an autosave is due to occur, the message "Autosave pending" appears on the command line. Once you exit the command, the autosave then proceeds as scheduled.

To disable settings in your local file, you can delete the entry or use the `unset` command.

The `settoggle` Command

Use the `settoggle` command to change the value of an environment variable based on its current value and a list of possible values. The syntax for the `settoggle` command is:

```
settoggle <variable name> [value1] [value2] ... [value n]
```

variable name	required environment variable name	
values [1 - n]	an optional list of possible values for the environment variable	
If you specify no optional values...	...and the variable is unset, the layout editor sets the variable with a value of " ", which is equivalent to: <code>set <variable name></code>	
	...and the variable is currently set, the layout editor unsets the variable, which is equivalent to: <code>unset <variable name></code>	
If you specify one value...	...and the variable is unset, the layout editor sets the variable to that of the specified value , which is equivalent to: <code>set <variable name> value 1</code>	
	...and the variable is currently set, the layout editor unsets the variable, which is equivalent to: <code>unset <variable name></code>	

	If you specify more than one value...	<p>...the layout editor substitutes the value listed immediately after the current environment variable value for the current variable. The comparison is case insensitive. The layout editor sets the environment variable to the first value in the value list when the variable:</p> <ul style="list-style-type: none">• is currently unset• has a value not in the list• has the same value as the last item in the value list <p>This is equivalent to: set <variable name> value 1</p>
--	---------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Command Examples for `settoggle`

Example 1

1. The following unsets the `pcb_cursor` environment variable:

```
unset pcb_cursor
```

2. The following sets the `pcb_cursor` environment variable to `infinite`:

```
settoggle pcb_cursor infinite cross
```

3. The following sets the `pcb_cursor` environment variable to `cross`:

```
settoggle pcb_cursor infinite cross
```

Example 2

1. The following unsets the `display_drcfill` environment variable:

```
unset display_drcfill
```

2. The following sets the `display_drcfill` environment variable:

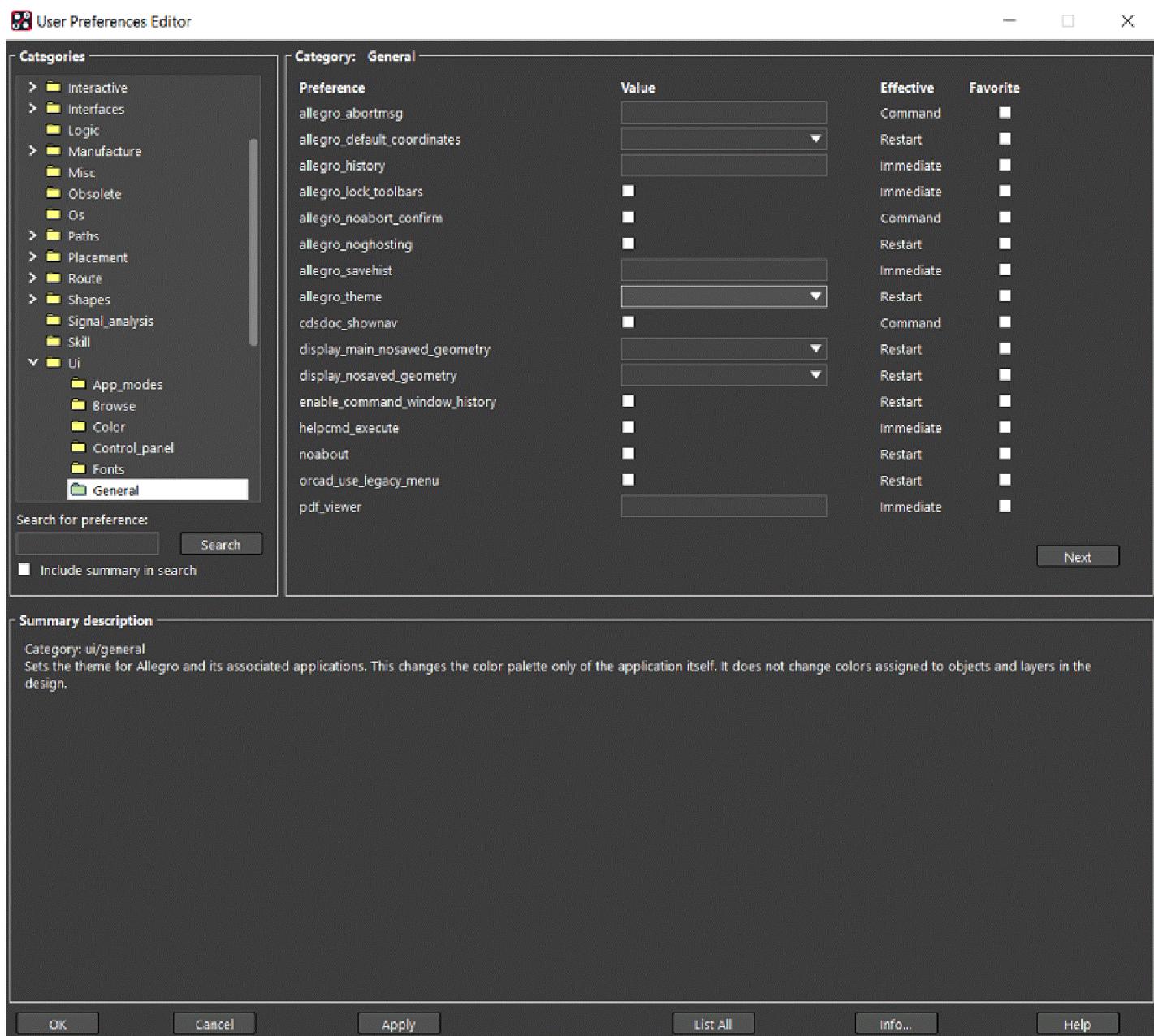
```
settoggle display_drcfill
```

3. The following unsets the `display_drcfill` environment variable:

```
settoggle display_drcfill
```

The User Preferences Editor

You can set or unset environment variables from the User Preferences Editor, a graphical user interface that you open by choosing *Setup – User Preferences* (`enved` command). A list of all user preference variables and a complete description appears when you click *Info* in the User Preferences Editor dialog box. A summary description of each variable also appears at the bottom of the dialog box when you change a value.



Customizing the User Preferences Editor

You can tailor the User Preferences Editor to display variables in groupings that meet your design needs, using either of the following:

- A *My Favorites* category
- Preference (.prf) files

The *My Favorites* category in the tree view in the User Preferences Editor centralizes frequently accessed variables. Selecting the *Favorites* check box next to a variable includes it in *My Favorites*, in addition to its current category; deselecting the check box removes the variable from *My Favorites*. Any change to the *My Favorites* folder updates the `my_favorites.prf` file in the `pcbenv` directory.

The User Preferences Editor dialog box displays groupings of preferences (user-defined environment variables) through a mechanism called a user preference file. Preference (.prf) files specify variables and their associated categories, so they can be used to customize the tree view control in the *Category* section of the User Preferences Editor dialog box.

The tree view contains parent and child categories, as opposed to a single-depth listing of categories. A category named *xyz* may have categories beneath it called *abc*, *def*, and *ghi*, for example. Use the `PARENT_L` entry in the .prf file to assign a parent category to any child category.

Note the following:

- Unless explicitly specified within the file, the name of the preference file determines the name of a category that appears in the dialog box. The name of the category associated with a preference file can be specified within the .prf file via the `PARENT_L` entry.
If no `PARENT_L` is specified at the beginning of the .prf file, then the name of the file determines the category appearing in the dialog. Otherwise, the `PARENT_L` string does. For example, for a file named `xyz.prf`, with the appropriate entry within that file, you can specify that its associated category be *abc* instead of *xyz*.
- Each entry in the file describes an environment variable that corresponds to one displayed in the User Preferences Editor dialog box.
- Entries in the .prf file do not contain or store value settings. They contain only the descriptions of the variables contained in the category. A variable's settings are saved to a user preferences section of your local `env` file

 When you create .prf files, ensure there are no blank lines at the end of the file. Blank lines in a user-created .prf file may cause the layout editor to crash.

Default reference files supplied by Cadence are located in your Allegro software install directory. Searching begins at the local level, so that preference files stored locally or in your home directory take precedence over preference files of the same name located elsewhere; for example, at a customer site location available to a group of users.

Setting Project Level and Site Customization Variables

If you are working within an HDL-based project, you may want to specify design library search paths at the project level to enhance integration of these tools into the design flow. If you are a CAD site administrator, you can customize the the layout editor environment for your work place.

Project File Variables

To better integrate a project into HDL-based design flows, you can base design path variables on the contents of the standard HDL (`.cpm`) file. This file controls variable settings when you work with HDL-based hierarchy; that is, when you open the Allegro software through Program Manager. You can also set environment variables when you run `enved` with a special option (see [Setting .cpm Variables](#)).

If design path variables are not set in the `.cpm` file, your design tool uses the variables defined in the PATH settings of your local environment file. The `.cpm` file supports any of the following design variables:

- PARAMPATH
- PSMPATH
- PADPATH
- TECHPATH
- MODULEPATH
- TOPOLOGY_TEMPLATE_PATH
- SIGNOISEPATH

The `.cpm` design path settings defines the `.cpm` project file at your user-defined location

Setting .cpm Variables

You can set .cpm-based design path variables by:

- Editing the .cpm file
- Accessing the env file
- Running Tool Setup in Project Manager
- Using the `enved` command with the `-proj <.cpm file location>` option. This is the recommended method and the one described in this section.
 1. Run the `enved` command from your operating system prompt with the `-proj` option, as shown in the example:
`enved -proj<.cpm file location>`
 2. The User Preferences Editor opens.
 3. You cannot run `enved` with the `-proj` option from the command prompt in the layout editor.
 4. Choose *Design_paths* from the Categories list.
 5. The design path preferences are listed in the dialog box. The CPM column appears only if you run `enved` with the `-proj` option.
 6. Check the CPM boxes for the design paths you want defined by the .cpm file.
 7. Choose *OK* to save the changes and close the dialog box.
 8. Restart the layout editor to put the changes into effect. (This step is necessary only if you are running `enved` in stand-alone mode, or if you are running setup from Project Manager

Site Customization

Site customization through the operating-system variable CDS_SITE lets you customize the Cadence-supplied environment by overriding the default site location, `<cdsroot>/share/local`. It allows you to create a directory hierarchy in CDS_SITE where you can place personalized files that extend or enhance your site's entire suite of Cadence tools. In addition to the CDS_SITE variable at the operating-system level, you can set a variable, ALLEGRO_SITE, within the layout editor for individual users. ALLEGRO_SITE lets you locate specific configuration files outside the standard default location, \$CDS_SITE/pcb. Site customization does not require any changes to the installation hierarchy or modification to the local environment.

 This feature is designed for use by CAD site administrators.

The layout editor searches for site-specific locations in the following order:

- \$ALLEGRO_SITE (default location: CDS_SITE pcb)
- \$CDS_SITE pcb
- <cds_root> /local pcb

Using CDS_SITE Functionality

The CDS_SITE variable allows you to create directories and files that support the Allegro software functionality. Directories you might choose to create at the CDS_SITE location could include:

- Standard script files in a "scripts" directory
- Locally developed Skill programs—and an *allegro.ilinit* file to load them—in a "skill" directory

The layout editor searches for Skill files in the following order:

- <cdsroot>/share/pcb/etc/skill (or a user-defined location specified by CDS_SITE)
- \$ALLEGRO_SITE/skill
- \$HOME/pcbenv
- . (the program's start directory)

 You can reverse the search order by setting the environment variable **skill_old_ilinit**.

You can also create a *site.env* file containing variable settings that would propagate across an entire design site. For example:

- Infinite cross-hair cursor
- Replacement of default Allegro symbol paths with your own

To effect these settings, your *site.env* file would need to contain the following data:

```
set pcb_cursor = infinite

set psmpath = . $allegro_site/symbol1 $allegro_site/symbol12
```

```
set padpath = . $allegro_site/symbol1 $allegro_site/symbol12
```

To load Skill files, your `allegro.ilinit` file must contain specific data. The following is an example of a `.ilinit` file. (This example file can be found at `<cdsroot>/share/local pcb/skill.`)

```
;
```

```
; This example file shows how to load Skill files (those with the
```

```
; extension ".il" in the current directory.
```

```
; To use, copy to allegro.ilinit if to be used by all Allegro PCB Editor-based programs
```

```
; or <programName>.ilinit if intended for only one program
```

```
;
```

```
; Setting Allegro PCB Editor environment variable, LoadSkillFilesDebug will turn
```

```
; on printing the name of each file as it is loaded.
```

```
unless (boundp ('LoadSkillFilesDebug)
```

```
LoadSkillFilesDebug = axlGetVariable ("LoadSkillFilesDebug"))
```

```
when (LoadSkillFilesDebug printf ("\n"))
```

```
(foreach file (rexMatchList "\\.il$" (getDirFiles "."))
```

```
when (LoadSkillFilesDebug printf ("Loading Skill file: %s\n" file))
```

```
(load strcat ("./" file))
```

)

```
when (LoadSkillFilesDebug printf("\n"))
```

Customizing Default Measurement Units Using CDS_SITE

You may want to customize a default unit value at the site level to accommodate your design requirements. For instance, when you run the `extracta` command to generate reports for a design, the default measurement units in the `$CDS_SITE/pcb/signal` directory are used when simulations have not been run on the design (that is, no `signoise.run` folder exists containing a `cds_signoise.cfg` file). For designs for which simulations have been run, a `cds_signoise.cfg` file exists in the `signoise.run` folder of each work directory, and the units specified in the `cds_signoise.cfg` take precedence.

For site-level customization of a default unit value, Cadence recommends copying the Cadence-supplied, system default `cds_signoise.cfg` file located at:

```
<cdsroot>/share/pcb/signal/cds_signoise.cfg
```

to the `CDS_SITE` directory, the standard location for placing company-wide customization files for Cadence tools:

```
<CDS_SITE>/pcb/signal
```

The `CDS_SITE` default location is:

```
CDS_SITE = <cdsroot>/share/local
```

For example, to change the default value of `EtchThicknessUnits` from mil to mm, edit the *Report Units* section of the `cds_signoise.cfg` file to reflect the new value. If no local version of `cds_signoise.cfg` exists at the design level (`.brd`), then the tool uses the `CDS_SITE` version. Once the company site file exists, you may have to delete the local `cds_signoise.cfg` file to ensure default values refresh.

For designs for which simulations have been run, modify the *Etch Thickness Units* field on the *Units* tab of the *Analysis Preferences* dialog box, available by choosing *Analyze – SI/EMI Sim – Preferences* (`signal prefs` command). Modify the `cds_signoise.cfg` as well to ensure any new designs function properly.

Cadence does not recommend modification of `units.dat`, a private Allegro file in the Cadence

installation hierarchy, as subsequent installation of patches can overwrite modifications.

Environment Compatibility

HDL design path information is ignored when you open designs in pre 14.2 releases. In these instances, traditional environment path variables are used.

Site-based changes that you make through the CDS_SITE variable are ignored in older shell environments unless you use the -q option when you source your environment file. Doing so appends a line to your master env file that reads the *site.env* file, when present. The format for the command is

```
source [-q] <environment_filename>
```

Managing Allegro X Physical Databases

This chapter provides information about database compatibility and analysis, script and SKILL compatibility, and the package integrity tool, including

- Database Compatibility Across Platforms
- Database Compatibility with Previous Software Releases
- Database UPREV (DBDoctor)
- Saving – Partial Versus Full Database Consistency Checks
- Script Compatibility
- SKILL Compatibility
- APD+: Using the Package Design Integrity Tool

Database Compatibility Across Platforms

Databases are compatible across all configurations on all platforms. uses the same database format for all versions of on all platforms, so no conversion is necessary to move between them.

Database Compatibility with Previous Software Releases

Allegro databases are backward-compatible within their major version number (the number to the left of the dot) by using the database downrev found in the tool's *File – Export – Downrev design*. These downrevs produce a log file indicating design data that are removed in order to be compatible with the older Allegro version. It will offer to save the design to a name different than the current design.

For 17.0, you downrev a design across major releases (number to the right of the dot).

Uprevving

Opening a design saved in a previous version of the layout editor in the current version automatically uprevs (updates) the database. For example, you can open a Release 16.5 database in Release 16.6 and choose *File – Save* ([save](#) command) to save it as a Release 16.6 database.

On either Windows or UNIX, you can also use DBDoctor to uprev the database to the current revision of software and move designs forward multiple revs. Use wildcard options to uprev an entire directory of designs.

For example, use DBDoctor for a design that originated in version 11.x, thereby preserving the original version of the design and uprevving it to a new name in the current version.

Database UPREV (DBDoctor)

Databases one major revision back and minor versions, when opened, automatically uprev to the current version. This means you can keep your physical libraries (.pad, .dra, and .sm) at the older major release if you do not need to update them with capabilities in the current release. Windows is unable to uprev databases older than 11.0 while Linux requires databases of at least 14.0.

Databases older than one major release must be run through an external program to update them:

```
dbdoctor_ui  
dbdoctor <file_name>  
uprev  
uprev_overwrite <file_name>
```

 Both `uprev` and `uprev_overwrite` offer a command line option to traverse directories to uprev all Allegro physical designs. This is useful when you uprev physical libraries. To understand how to use these options, in either tool at the command line use the `-help` option; For example:

```
uprev -help
```

DBDoctor can also:

- Analyze and fix database problems.
- Regenerate xnets
- Eliminate duplicate vias
- Perform batch design rule checking (DRC)
- Lock and unlock designs (batch command only)
- Update dynamic shapes (batch command only)
- Purge used vias (batch command only)
- Perform batch design rule checking (DRC)

Saving – Partial Versus Full Database Consistency Checks

When you save a design, the layout editor executes a partial database consistency check by default, in essence, a quick check.

The `dbsave_full_check` environment variable indicates to the database save utility when to do a full check rather than a quick check. A number of 1, or 0 specifies that each time a design is saved, execute a full check. If you set the variable to 100, then every 100 checks, a full check occurs.

For example, to set the `dbsave_full_check` environment variable to do a full check every five saves, at the console window prompt, type:

```
set dbsave_full_check = 5
```

If the layout editor detects errors, it saves the file as `<design_name>.SAV`.

 A full database check may considerably lengthen the time required to save large databases.

Script Compatibility

 Cadence does not guarantee that scripts are 100% upwardly compatible from release to release.

SKILL Compatibility

SKILL programs are fully compatible with the layout environment and should run without modification.

APD+: Using the Package Design Integrity Tool

The Cadence IC Packaging tools are complex, flexible tools that provide many ways to create a package substrate layout. Although the tools have a built-in check for the integrity of the database, the database doctor only validates that the database is architecturally correct, not that it is structured properly for the many commands that may access it. As a result, you might have run a particular feature at a time when the database was not configured to handle the request. For example, if you used the IC Packaging software to interface with the signal integrity field solver, the solver may have returned incorrect results if some information in the database is missing or incorrectly configured properly. The solver would run for hours, even days, before returning bad results.

With the new Package Design Integrity tool, you can:

- Save time by running integrity checks to ensure that the database is configured correctly.
- Diagnose your design problems before calling Cadence Technical Support.
- Customize the tool to look for problems or deviations from your company requirements that the standard tool may not consider errors. See [Adding Checks Using SKILL Functions](#).

See the `package integrity` command for additional information on field descriptions and procedures.

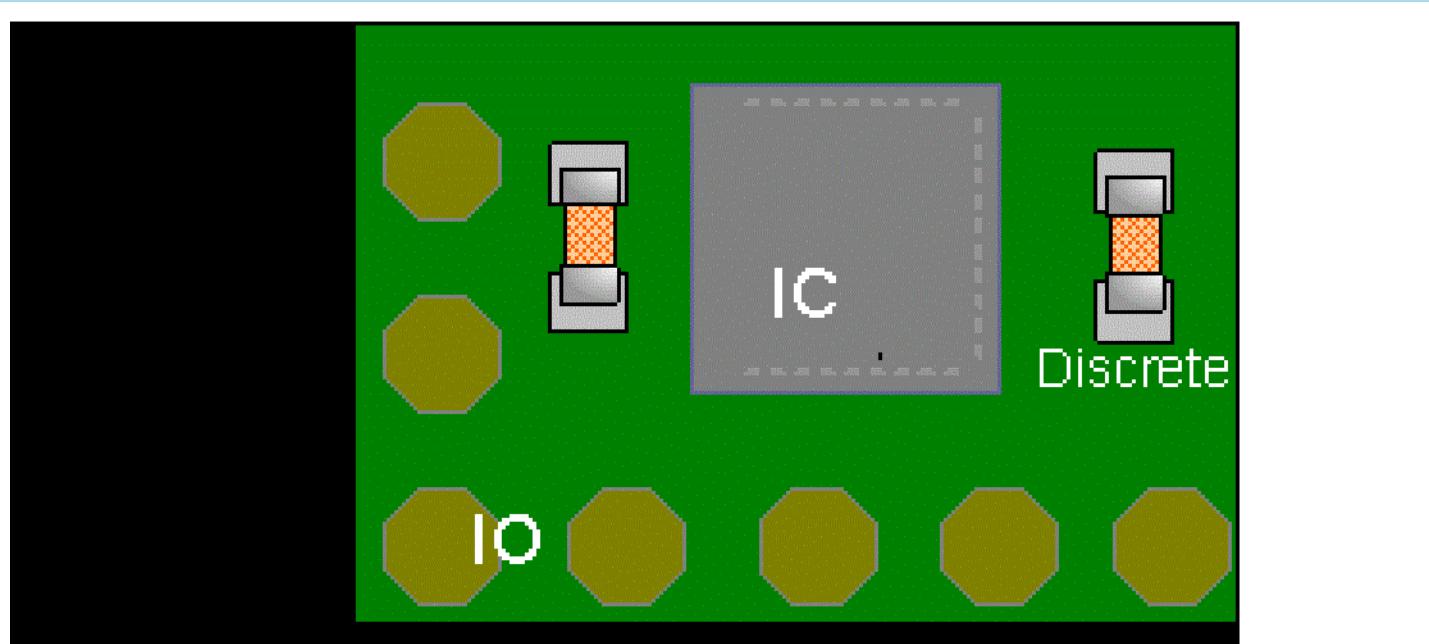
Package Design Integrity Checks

The following table describes the checks that are currently included in the standard tool. Any rule that the tool can automatically fix has an F suffix and, when selected, displays the rule name followed by *Fixable* in the right panel. Rules, that the tool cannot automatically fix, display instructions on fixing the problem and reasons why the resulting database structure is an improvement.

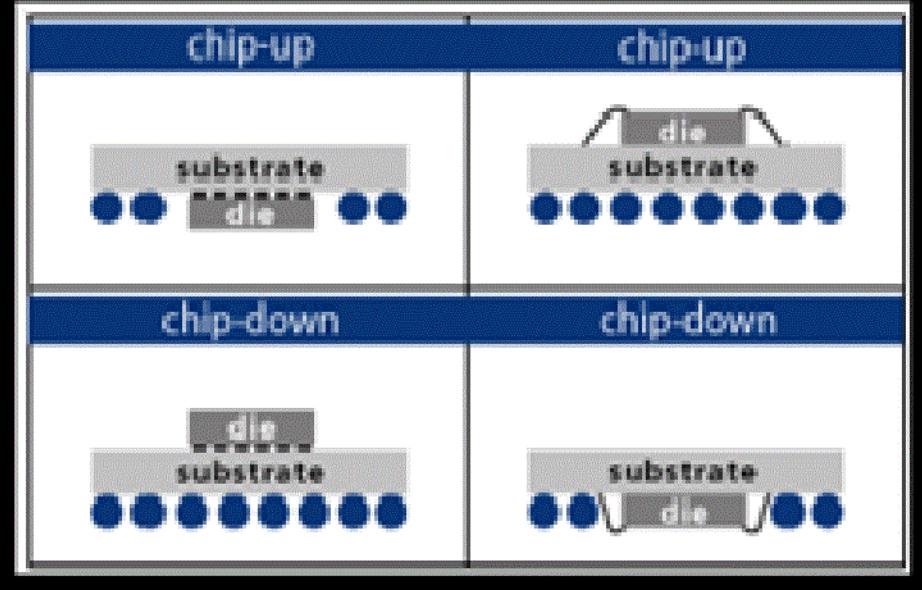
General

General

This category includes checks that may impact a wider range of commands within your package substrate layout editor, and therefore do not lend themselves to inclusion in one of the more specific categories. Correcting issues reported in this category may improve results obtained with multiple commands in the system. If you are running checks from one or more categories, it is recommended that you also consider running these supplementary scans.



Defaulted Component Class (F)	<p>This class plays an important role in the component treatment within the Cadence IC Package design tools. A die component, for example, may be either flip-chip or wire bond. Only a wire bond die may be connected to the substrate with bond wires. The following describes the usage of the component classes:</p> <ul style="list-style-type: none">• IC: Die Components, flip-chip or wire bond• IO: BGA components• DISCRETE: Discrete components such as capacitors and resistors• PLATING_BAR: Reserved for the plating bar in the design• MECHANICAL: Parts with no logic, such as via structures <p>If you assign the wrong component class to a component instance or device, it may not behave in the expected manner. You cannot copy a discrete component with class IC, as the tool views it as a die. A wire bonded die with class IO is drawn with balls on the pads in the Cadence 3D Design Viewer because the tool views it as a BGA component. In many cases, the tool may mistakenly set the component class IC for discrete components, as any component without a class specified in the front-end tools defaults to IC class when added to the physical layout. Most of the time, this tool can correct the issue by changing the class. If the tool cannot deduce the proper class, it flags the component with a DRC so that you may fix it yourself using the <i>Logic – Edit Parts</i> command.</p>
	To prevent future occurrences, be sure to assign a component class when defining components in the front-end tool.

Flip Chip**Wire Bond****Die Attachment Method**

The attachment method of a placed die symbol must be correct in order for many of the commands in the IC Packaging tools to operate on them correctly. If a die is classed as a flip-chip, for example, the tool will not allow you to add bond wires from the pins to fingers of the substrate. Similarly, if the die is classified as a wire bonded die, the pads must be on a DIESTACK/DIE type layer outside the substrate layers. If the die is actually a flip-chip, its pads will most frequently need to be on a CONDUCTOR type layer or inside the substrate.

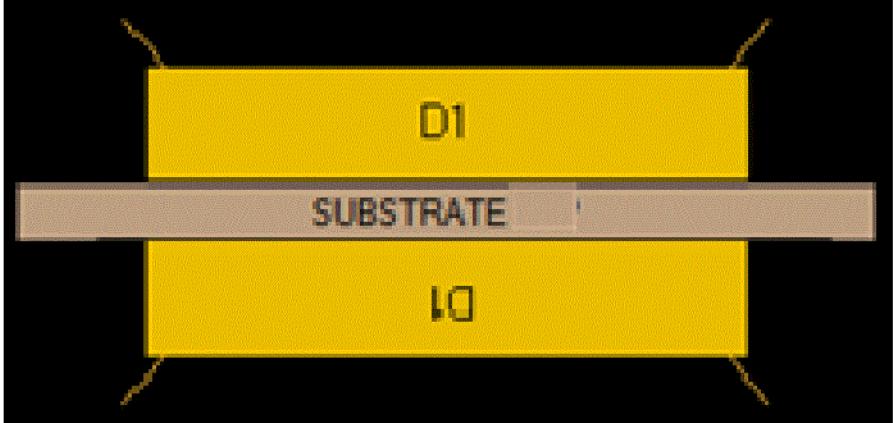
The most common orientations for dies are as follows:

- Wirebond: Chip-up (not mirrored) on top of the substrate surface layer
- Flip-chip: Chip-down on top of the substrate layer

If the component is placed on the bottom (underside) of the substrate, the orientation and mirror setting are reversed, normally. These are not the only possible orientation and chip attachment combinations, however. The tool also supports flip-chips placed chip-up (or down) on a DIESTACK layer, where it connects to an interposer for later connection with the package substrate. To correct a die's attachment method if it is wrong, run the *Edit – Die Properties* command and select the proper attachment method. This will update the die and you will see the changes reflected immediately in everything from Show Elements to the Die Stack Editor.

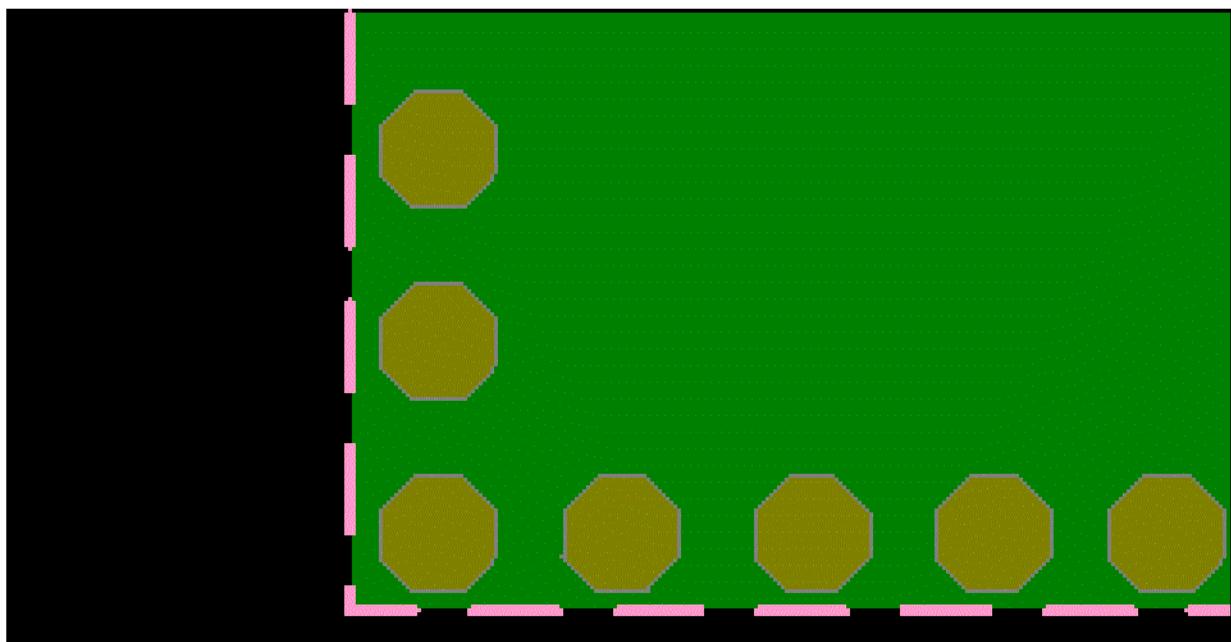
Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Die Group Membership (F)	Die components in IC package designs carry additional information beyond that required for discrete, IO, or plating bar components. Such information includes, but is not limited to, source IC design information and chip attachment type (wire bond versus flip-chip). When a component is changed from IC to another class, this information must be removed. When a component is changed to class IC, default values need to be established. If this information is missing, a die component may not be treated correctly by some parts of the tool. For example, it may not show up in any die stack, or it may be viewed as a die instead of a BGA and not be properly selectable for logic assignment operations. This check looks for incorrect or missing die groups. If desired, it can fix the situation by adding or removing the bad information.
	
Die Symbol Orientation	The orientation of a placed die symbol must be correct so that many of the commands in the IC Packaging tools operate on it correctly. If the die is oriented the wrong way, the wire bond pads will be on the wrong side of the substrate (or bumps for a flip-chip die), which can cause extraction and 3D viewing errors. It can also lead to improper relationships with other elements that are part of the same die-stack. The most common orientations for dies are: <ul style="list-style-type: none">• Wire bond: Chip up (not mirrored) on the top of the substrate• Flip-chip: Chip down (mirror geometry) on the top of the substrate. If the component is placed on the bottom (underneath) of the substrate, the orientation and mirror settings are reversed normally. These are not the only possible orientations, however so this rule does not actually change the database. The easiest way to correct these errors is to export a die text file. Check the mirror pin coordinates in the y axis option on the second page of the wizard. Then read the text file in again to replace the die. Select the same target pad layer but pick the correct chip orientation. This updates the die and reconnects any bond wires back to the pins.
Missing Dielectrics	A dielectric layer is necessary between each pair of adjacent conductor layers. These layers ensure there is no short between overlapping conductive objects on those layers. When dielectric layers are absent from a design, many commands will not be able to provide accurate feedback. This includes any signal integrity or power integrity extraction, most notably, but does not extend to other areas of the tool (including such simple things as calculating a via's true vertical height). You may notice that certain commands take much longer to run, as they are unable to properly resolve electrical characteristics of and between layers. These layers cannot be created for you automatically by this tool. They should be added through the Setup - Cross Section command. They are not added automatically because they must be defined with the correct thickness, material, and other properties in order to ensure the accuracy of signal integrity extraction, 3D display and so on.

Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

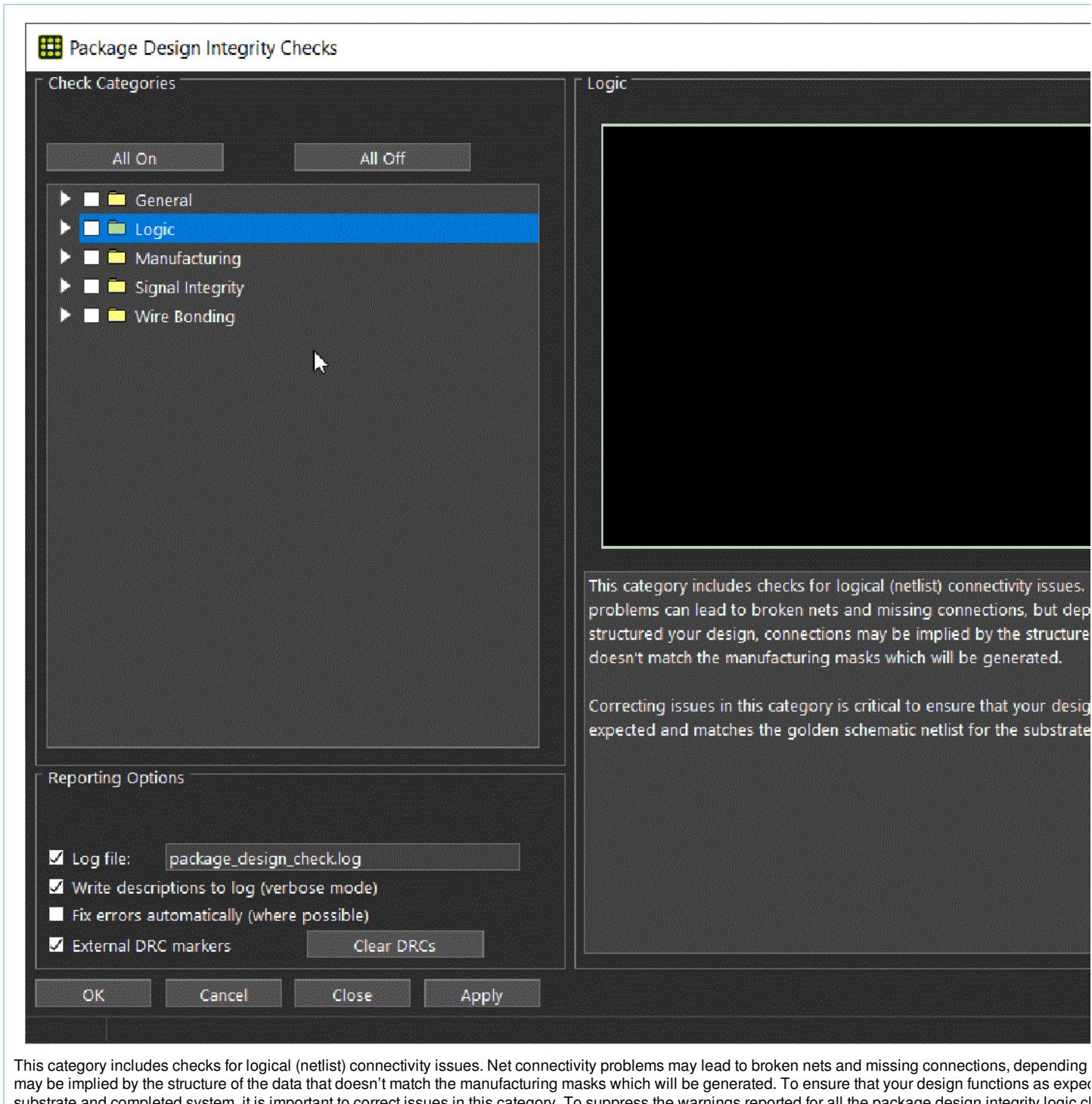


Missing Substrate Outline (F)	A package design requires a substrate geometry outline shape, which defines the overall boundary of the physical design space. In most cases, this is the same size as the main BGA component in the database. For example, this package outline shape would normally include the plating bar region as this is not part of the final physical package. The substrate outline is used in a number of commands within the system, from routing to analysis. As a result, it is important to have this boundary in your design as early as possible in the flow. The BGA generator and the BGA text-in wizards automatically define a substrate outline rectangle the same size as the component. If this size is incorrect or you are reading the BGA from another source, it may be necessary to manually add the outline. This tool can correct this issue by creating an outline which is the same size as the largest of the BGA (IO class) components in the design. However, it is not able to create the rectangle if no BGA exists. In this situation, a DRC is added to the design. To prevent future occurrences of this error, add the substrate geometry outline to your design manually as soon as you know the design's size or use one of the BGA wizard commands to ensure its automatic creation as part of the BGA component.
Unidentified Voltage Nets(F)	Any power and ground nets in a design should be identified as such with the appropriate properties. It is not sufficient just for the pins on that net to have a use of <i>POWER</i> or <i>GROUND</i> . Each net should have the <i>VOLTAGE</i> property (<i>0.0v</i> for ground, positive voltage for power) and in most cases, the <i>RATSNEST_SCHEDULE</i> property should be set to <i>POWER_AND_GROUND</i> . When these properties are missing, it can cause portions of the tool to perform slowly or inaccurately, such as signal integrity extraction and electrical constraint calculations. In other cases, the design ratsnests lines may appear confusing and messy because the voltage nets are being drawn with the default signal net scheduling algorithm. This tool can fix these errors. It will add a <i>VOLTAGE</i> property, with a value of <i>0.0v</i> for nets having names containing the strings <i>GND</i> , <i>GROUND</i> , or <i>VSS</i> . It will use a value of <i>1.5v</i> for other nets that meet the pin number requirements. It will also add a missing <i>RATSNEST_SCHEDULE</i> property with the <i>POWER_AND_GROUND</i> value. The <i>packinteg_voltage_nets_min_pins</i> environment variable can be set in the <i>IC_Packaging/Package_Integrity</i> folder of the User Preferences to control the minimum number of pins before a net is considered as being power or ground.

Logic

Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks



Getting Started with Physical Design

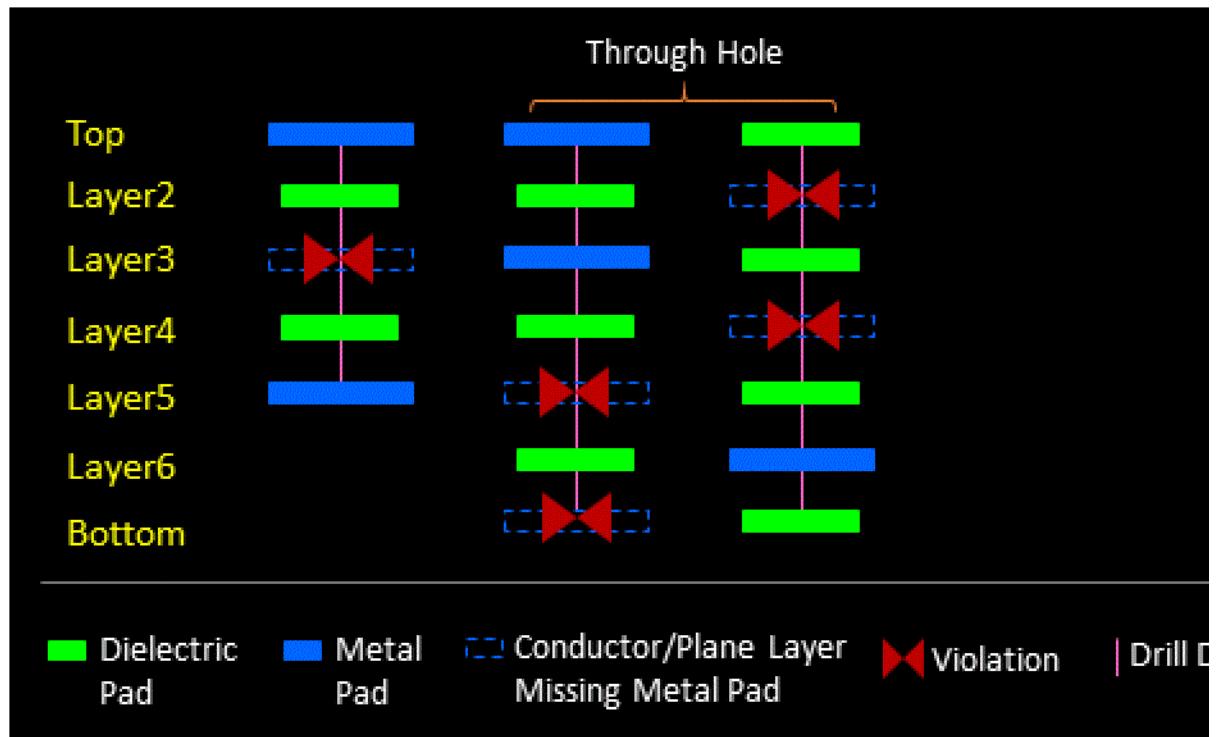
Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Opens from Missing Conductor Pads

If the package design has named dielectric layers and padstacks with a finished diameter of 0.0 for drill, the dielectric pad opening is as immediately above and below this dielectric layer. In such designs, missing dielectric openings may lead to undetected missing connections for the padstack definition, all the pads in the padstack are assumed to be connected vertically together by the drill.

- If a padstack with 0.0 drill diameter is not a through hole and any conductor or plane layers between the first and the last pad of the padstack are missing, such violations are reported in the log file that is generated after running the check.
- If a padstack with 0.0 drill diameter is a through hole and any conductor or plane layers between the first and the last pad of the padstack are missing, a regular pad definition (metal pad) is generated after running the check.

This rule cannot be automatically fixed because the appropriate metal pad size and shape cannot be determined. To add the missing padstacks, When complete, re-run this check to verify that there are no missing metal pads that can cause missing connections.



Getting Started with Physical Design

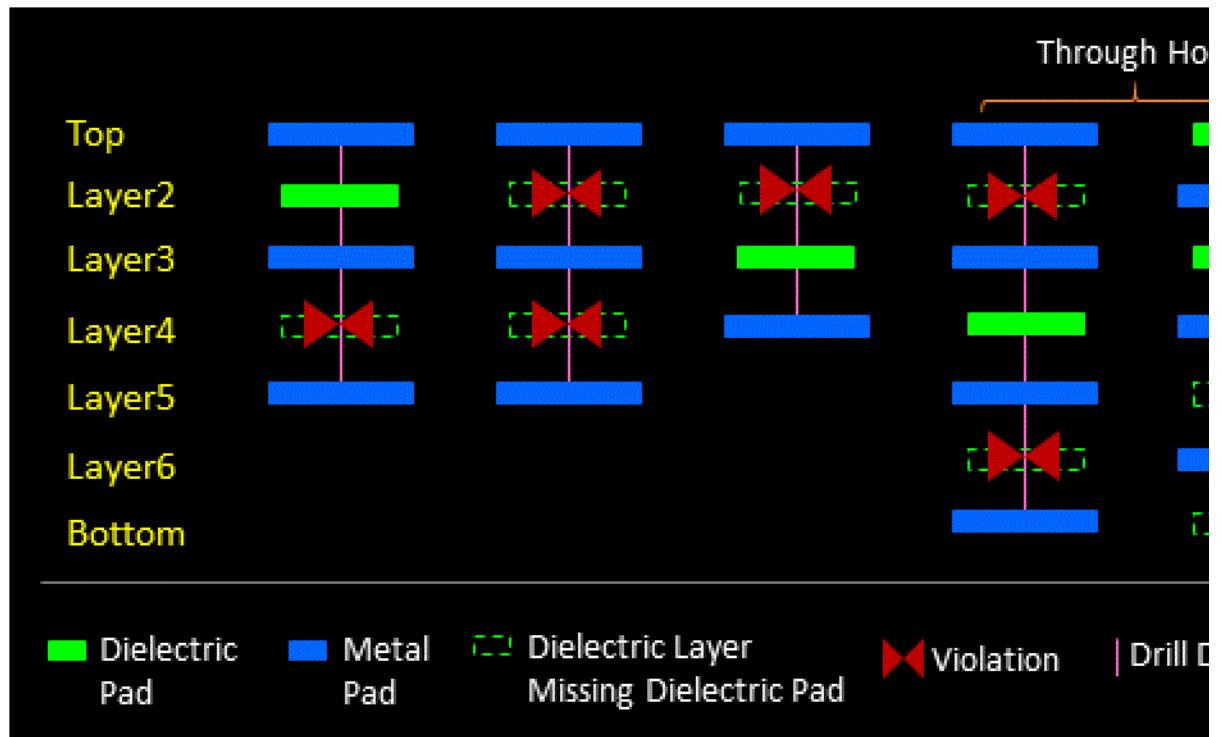
Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Opens from Missing Dielectric Openings

If the package design has named dielectric layers and padstacks with a finished diameter of 0.0 for drill, the dielectric pad opening is as immediately above and below this dielectric layer. In such designs, missing dielectric openings may lead to undetected missing connections the padstack definition, all the pads in the padstack are assumed to be connected vertically together by the drill.

- If a padstack with 0.0 drill diameter is not a through hole and any named dielectric layers between the first and the last pad of the padstack opening), violations are reported in a log file that are generated after running the check.
- If a padstack with 0.0 drill diameter is a through hole and any named dielectric layers is missing a regular pad definition (dielectric generated after running the check)

This rule cannot be automatically fixed because the appropriate dielectric opening size and shape cannot be determined. To add the missing dielectric openings manually. When complete, re-run this check to verify that there are no missing dielectric pads that can cause missing connections.



Getting Started with Physical Design

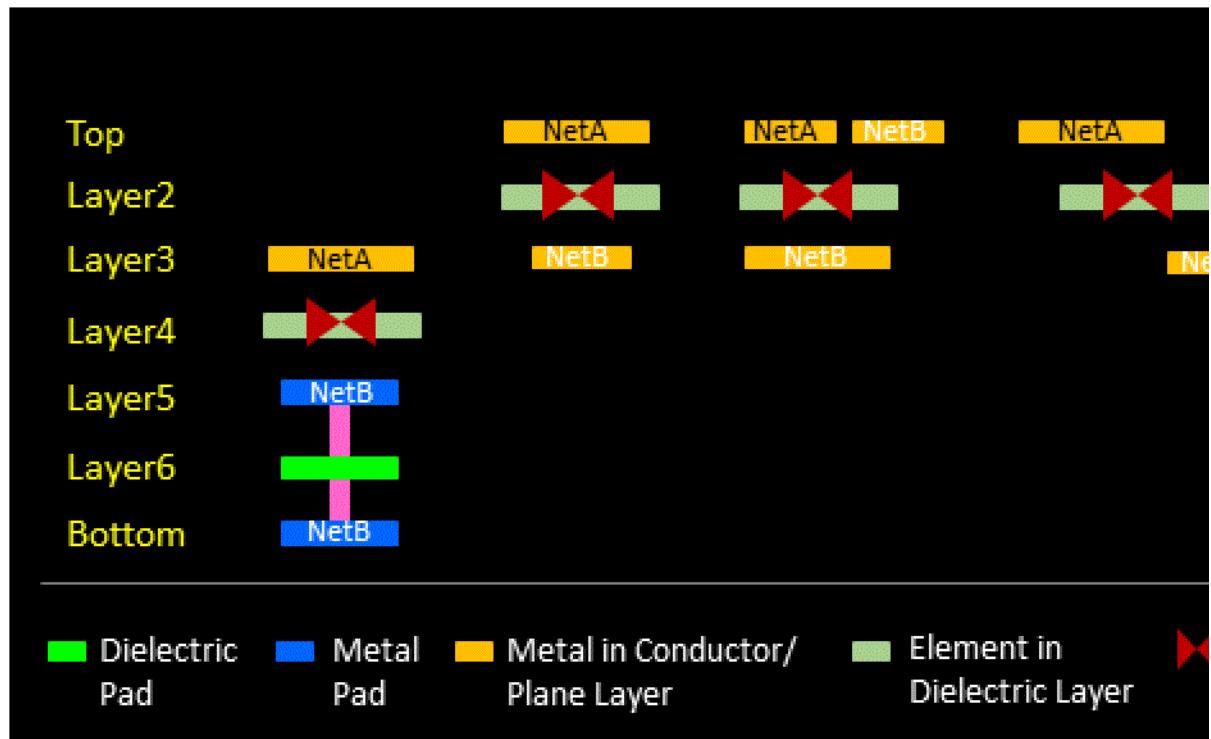
Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Shorts from Dielectric Layer Elements (F)

If the package design has named dielectric layers, a common practice is to define regular pad geometries on these layers to represent them. If this is done, but you also have other elements, such as shapes, clines, and lines on the dielectric layer then inadvertent shorts flagged by the tool. This check searches for elements, other than via and pin pads on these layers, and warns about elements that can be displayed:

- If an object overlaps other objects on adjacent layers, especially, if the two objects are on different nets, an error is displayed.
- If there are no objects connected by this dielectric element, a warning is displayed.

In all cases, it is recommended to disable the conductor elements on these layers using Constraint Manager. Doing so, you will get feedback named dielectric and avoid unintended errors. This rule can be automatically fixed. Doing so, the routing elements on the dielectric layers will investigate these elements to determine if these are added with the intention of creating an inter-layer connection.

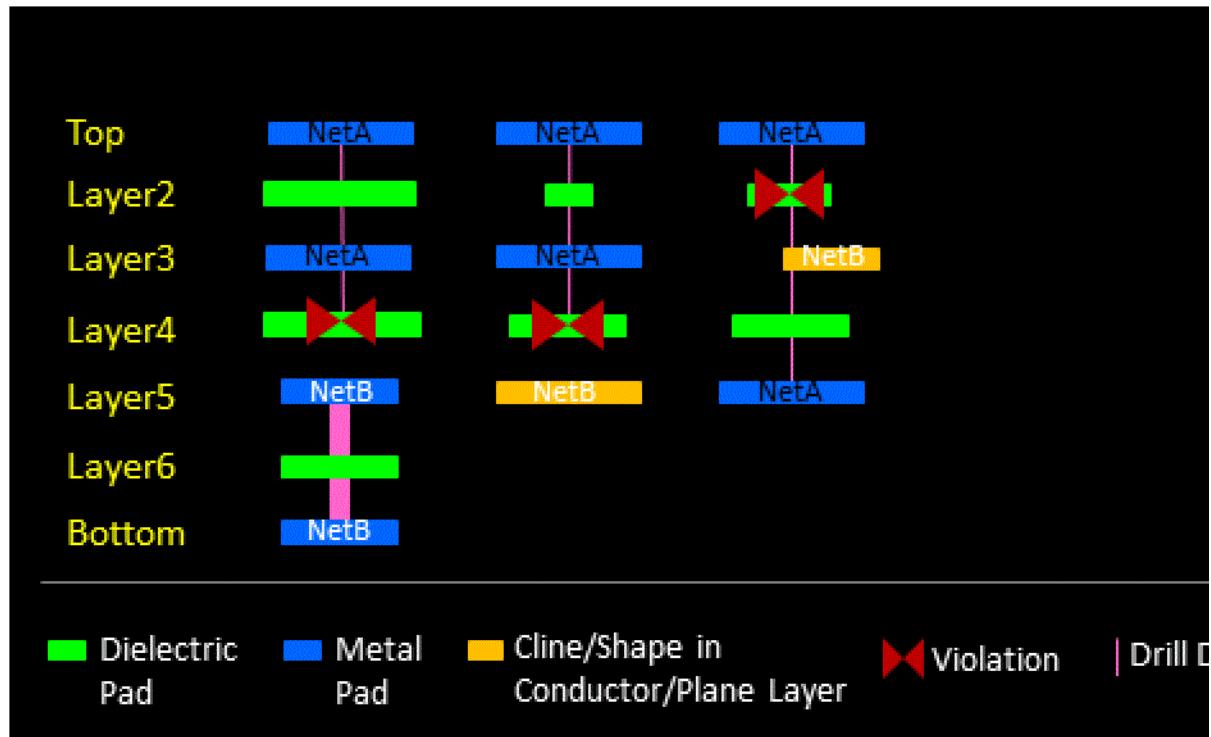


Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Shorts from Dielectric Openings

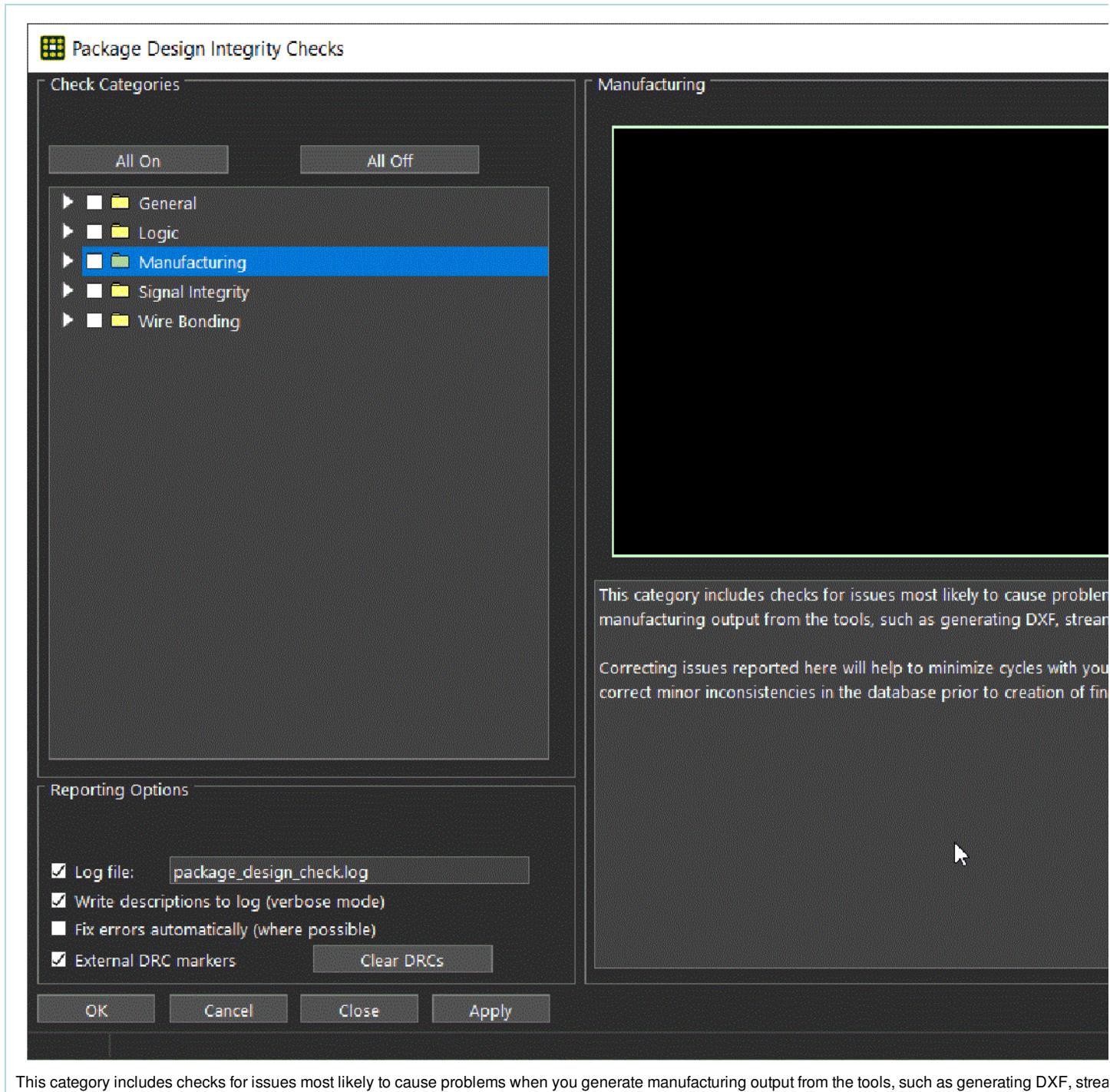
If the package design has named dielectric layers and padstacks with a finished diameter of 0.0 for drill, the dielectric pad opening is as immediately above and below this dielectric layer. All pads in the padstack are assumed to be connected vertically together by the drill. In such designs, any potential shorts arising from the pad geometries in the named dielectric layers are undetected. A violation will be reported for objects or nets that have the NET_SHORT property. This rule cannot be automatically fixed. You must edit the design appropriate metal under a dielectric pad opening, you must create a route keepout on the adjacent conductor or plane layer. To prevent etch obstructions, set Etch Allow to False for the dielectric layers in Constraint Manager. If there is an internal missing metal pad resulting in a short, padstack overlaps with an adjacent dielectric pad, you must edit the shape/cline manually so that it does not overlap with the adjacent conductor to verify that there are no more violations that can cause a short.



Manufacturing

Getting Started with Physical Design

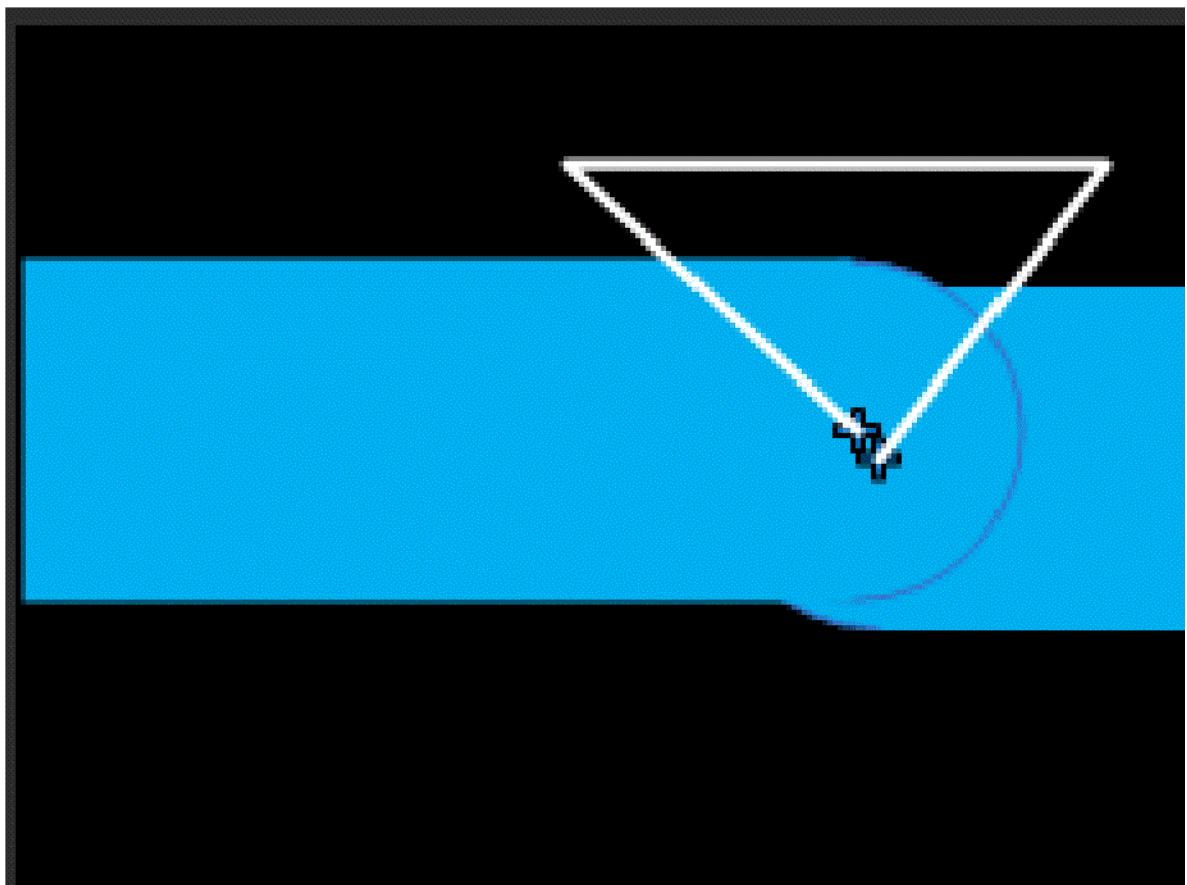
Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks



This category includes checks for issues most likely to cause problems when you generate manufacturing output from the tools, such as generating DXF, stream category helps minimize cycles with your package foundry. You can correct minor inconsistencies in the database prior to the creation of final mask data.

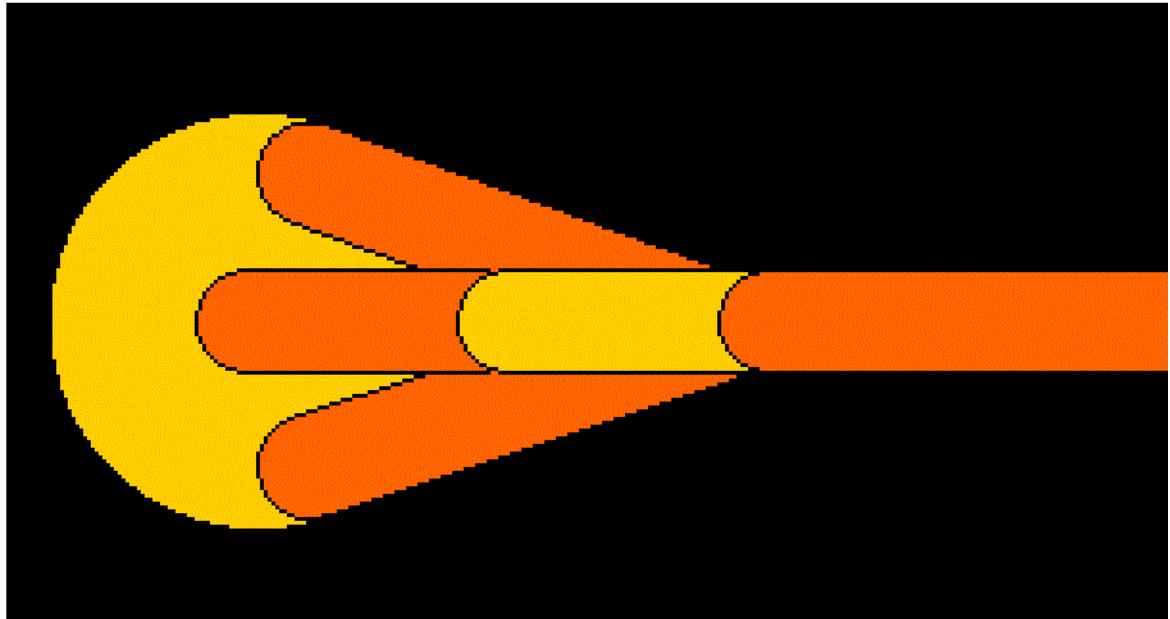
Disconnected Clines

Cadence IC Package databases using an origin-based connectivity model, not a touch or overlap model. This means that routing din They cannot merely overlap each other. Similarly, clines must connect all the way to a via or pin origin point, not merely cross the pac in the package netlist where the gap is extremely small, or where the metal actually overlaps -- just not to the extent that the element c connections can be difficult to locate and correct for this reason. This check, when run in fix mode resolves issues by moving vias is tl any connected dines/vias with it. Cline end points are stretched to connect to the pad origin or, for two overlapping cline end caps, to



Extra Cline Segments (F)

During the course designing a package substrate, most routing clines will be pushed, shoved, smoothed, or otherwise adjusted. This segment is actually multiple segments. This can sometimes cause problems when adding fillets or generating manufacturing data. You can merge these commands. However, these commands can cause additional changes that you may not want in the fully-routed design. Run can be merged. No other changes are performed.

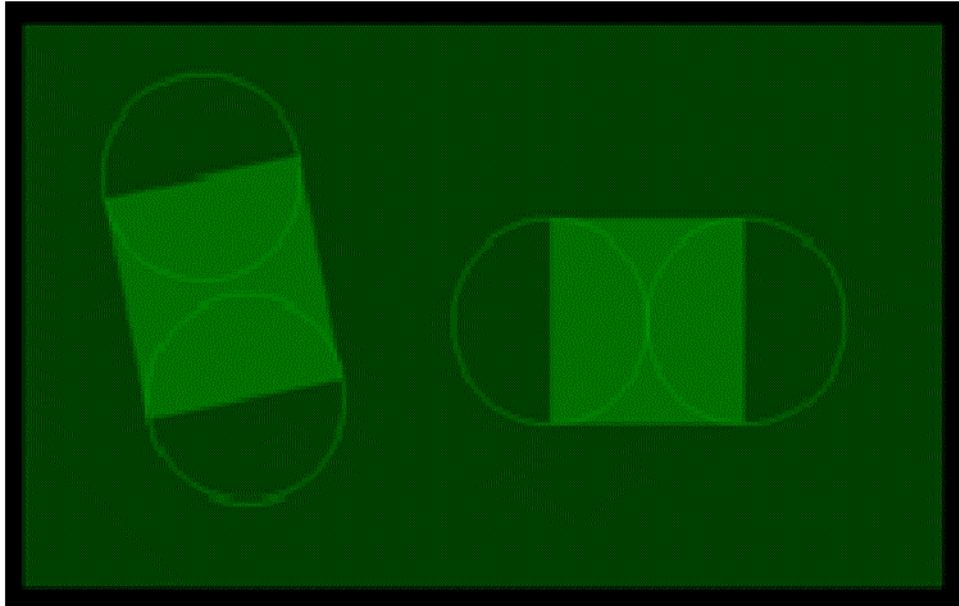


Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

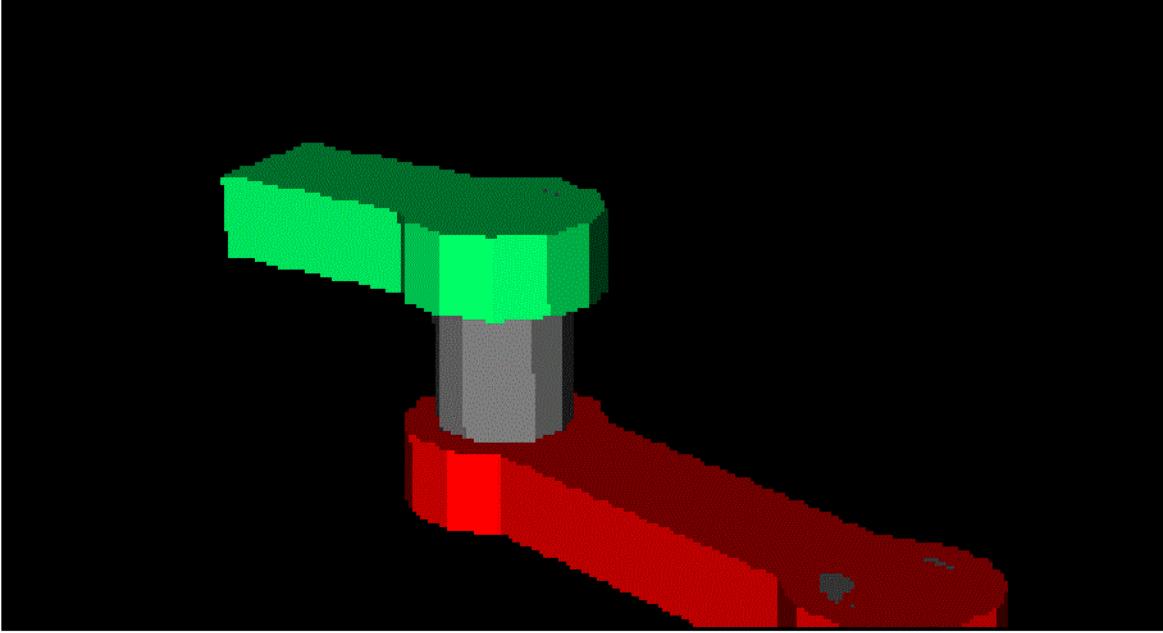
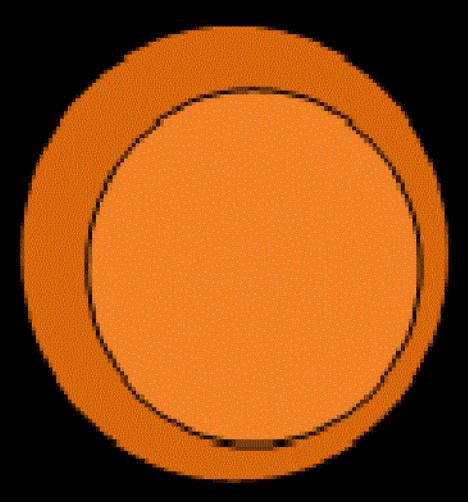
Redundant Clines (F)

During the course of designing a substrate, it is possible to create clines which are completely underneath a shape on the same layer. This can confuse signal integrity calculations, etch length calculations, and more. This can also add geometries in the manufacturing data which are not needed. Extra clines can be safely removed by this tool without changes to the logical connectivity or the integrity of the overall database.



Getting Started with Physical Design

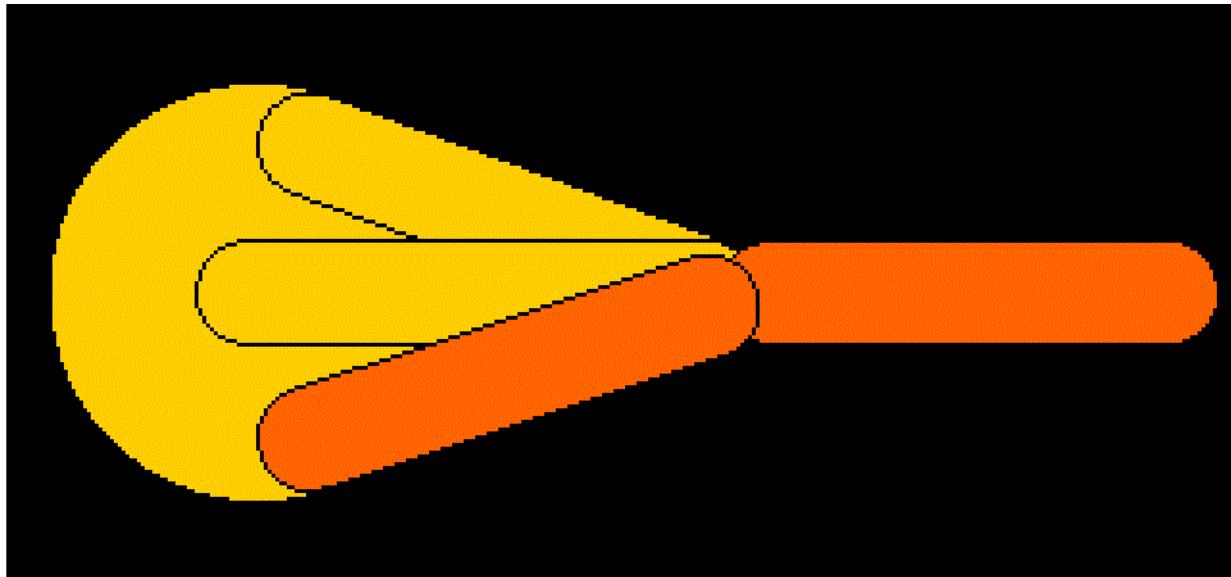
Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Redundant Padstacks (F)	<p>During the course of designing a substrate, it is possible to create multiple identical vias in the same location. These extra vias can increase conductor length for a net, while also adding unnecessary geometries to manufacturing output formats. This tool can safely remove redundant padstacks to ensure the integrity of the database. To minimize the chance of this scenario occurring in future databases, ensure that the appropriate layers are used during operations or when placing via structures.</p> 
Via-Pin alignment (F)	<p>When a component is moved or replaced (ECO process, etc.) during the course of designing a substrate, vias connected to the pins will move to the new location of the pin. While the two items are still connected, they are not considered aligned. This may be undesirable for manufacturing. Where direct connections to pins occur and the location of the via and pin are not same, a DRC error will be created at the via's pin's location and any connected clines will be stretched to the new location to maintain connectivity. Note that if multiple vias (for instance) are connected to a single pin, only the via closest to the pin's original location will be aligned. Only pins with single via connections will be processed.</p> 

Signal Integrity

Signal Integrity

This category includes checks for items that are most likely to cause problems when performing signal integrity analysis using either a 2D or 3D field solver. Correcting issues reported in this category helps minimize problems running the solvers, while also ensuring that the tools give the most accurate results possible.



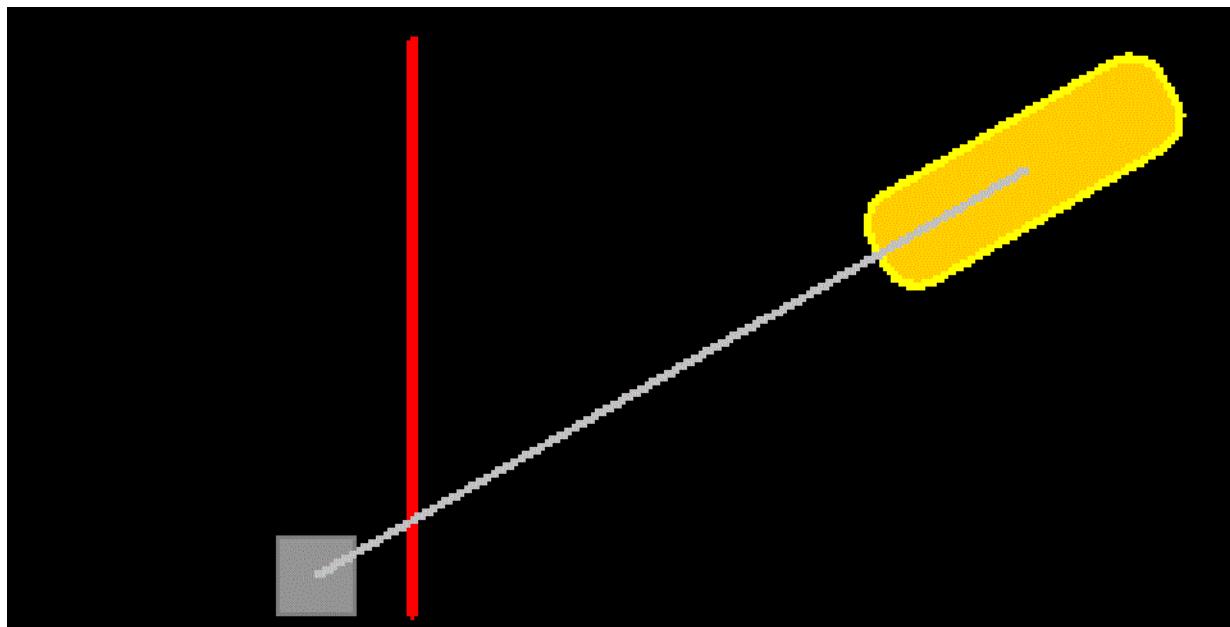
Bad Fillet Properties (F)

Routing clines, that have the FILLET property attached but are not real fillets, can cause the system to generate inaccurate results for signal integrity, routing length, and other calculations. Fillets are ignored in many calculations as they do not impact the electrical path, but rather are present to improve the substrate manufacturability. Therefore clines, which have this property but may cause inaccurate calculation results. This situation can occur when you edit routing after adding fillets to the design. Adjacent cline objects on the same net may merge into a single cline object. If one of the original clines has the FILLET property, this may be mistakenly inherited through this process. Similar problems can occur when pasting from a subdrawing. You can prevent this problem by adding fillets to your design only at the end of the design flow. If you need to edit routing after you add fillets, first remove any fillets from the clines or nets in question, perform your edits, and then re-run the filleting procedures.

Wire Bonding

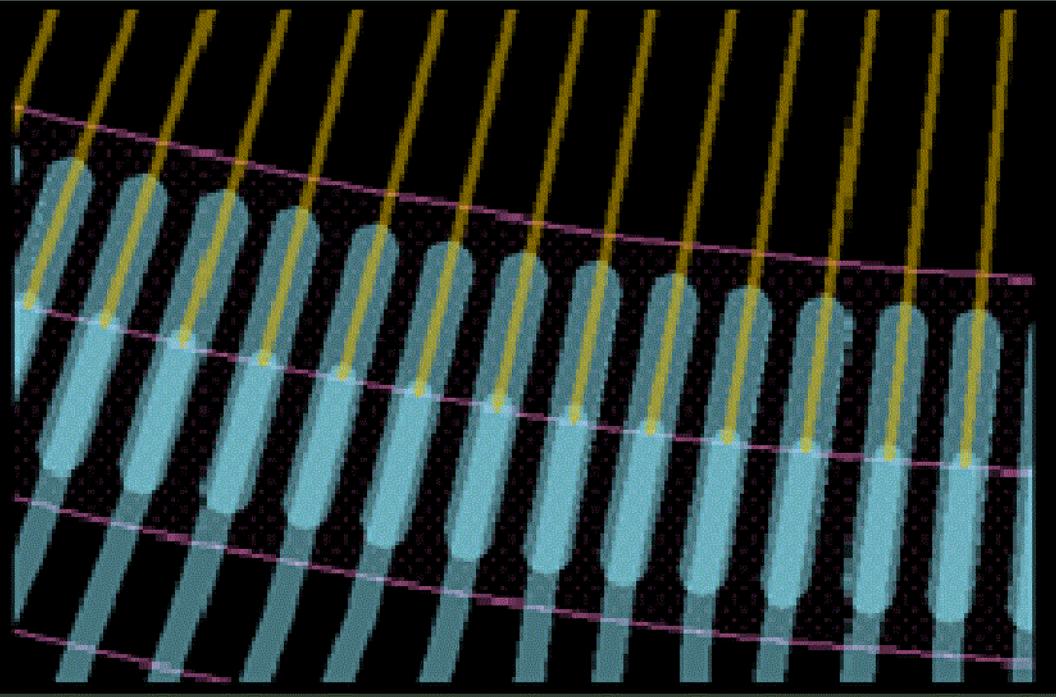
Wire Bonding

This category includes checks for issues related to adding and manipulating the wire bonds in your design. Correcting issues reported in this category helps to ensure that you do not encounter unforeseen problems when trying to adjust your wire bonds later in your design flow. In many cases, these checks may also help if you are experiencing problems with the display of bond wires in the Cadence 3D Design Viewer or extracting them for the 3D field solver.



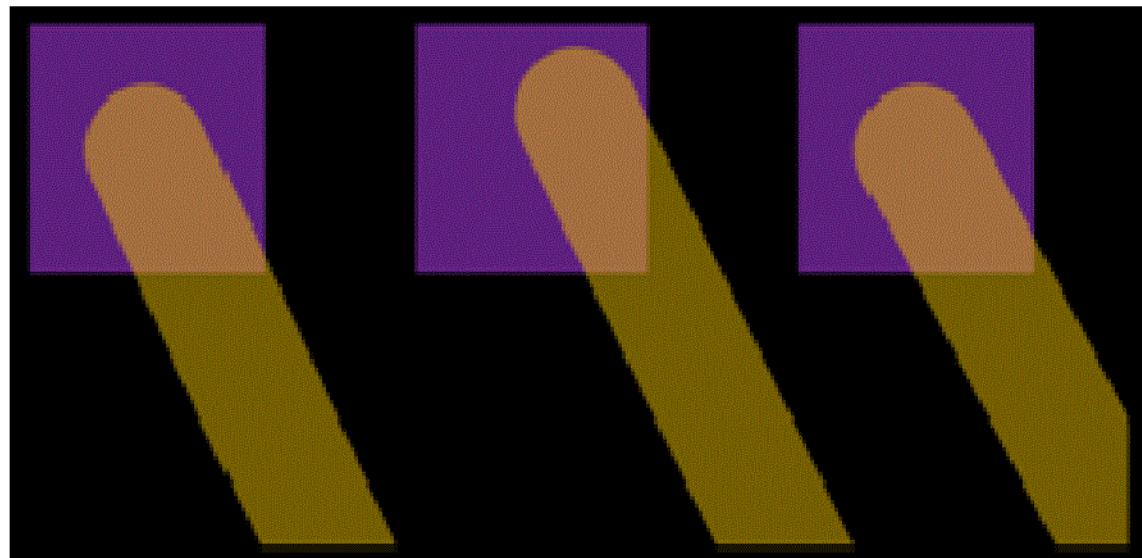
Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Dangling Bond Wires (F)	Bond wires are dangling when they are not connected to valid objects, such as pins, bond fingers, and rings, on both ends. Dangling bond wires do not know the height of the object to which they should connect at the dangling end, which is a requirement to draw the wire accurately in the Cadence 3D Design Viewer. As a result, a wire that is dangling has that end drawn to an undetermined z-axis elevation in the Cadence 3D Design Viewer. This can cause further issues with any 3D-based DRC checks that you want to perform. In most cases, dangling wire bond problems occur if tools other than the <code>wirebond</code> commands are used to modify the wire bonds in the design. Using the <code>edit vertex</code> , <code>move</code> , or <code>add connect</code> commands to modify bond wires is a primary source of this problem. To minimize the odds of reoccurrence in the future, modify the wire bonds only with the commands in the <i>Route – Wire Bond</i> menu of the layout editor.
Duplicate Wire Bond Guide Paths (F)	 On path and orthogonal style bond finger placement requires a guide path object that the finger follows as it moves. It is possible for multiple guide paths to be defined in the database which are either very close together or completely identical. When this situation occurs, the tool cannot tell, for a specific finger, which of the two paths should be used. As a result, if some fingers associate with one overlapping guide path and some with another, the results of some push and shove operations may be undesirable, particularly when in the "shove all" placement style. One manifestation of this problem is that fingers will, apparently at random, stop pushing adjacent bond fingers. This situation can be easily remedied by removing the duplicate guide paths. However, in a complex design, the precise guide path(s) which have more than one instance may be difficult to find. This check will identify the presence of duplicate guide paths. If enabled, it will remove one of the duplicate entries. This rule will only delete exact duplicate paths. If two paths are very close together, it will warn you of this situation, but will not remove either path, to avoid a potential incorrect change to your design that may not be noticed until late in the design flow.

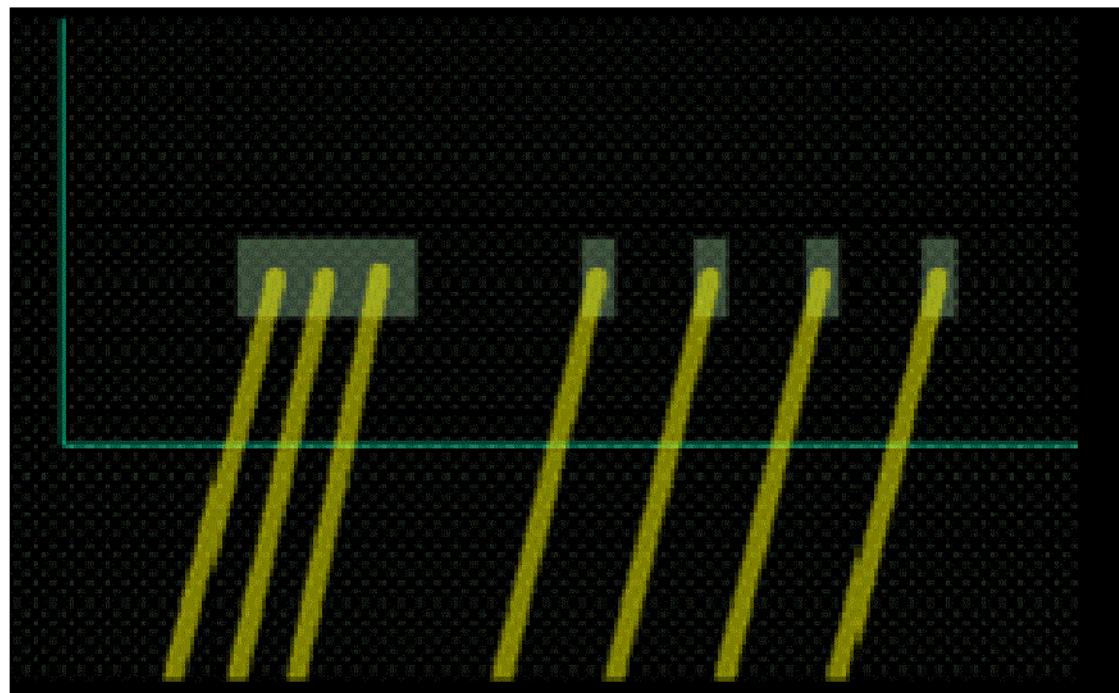
Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks



Pad Bond Wire Offset (F)

In cases where a single bond wire is connected to a die pin or bond finger, that wire should connect at the pad's center to ensure proper connectivity when the design is manufactured. If the wire does not connect at the center, the bond wire may only partially connect to the pad or, in the worst case scenario, may not connect to the pad at all if the alignment of the bonding machine and the offset of the bond wire to the pad center are off by their maximum tolerances. This check will look for bond wires that connect to die pins and fingers at locations other than the pad center. These will generate errors in the log file and external DRC markers in the design (if requested). These wires can be automatically corrected by this check, if desired. In this situation, the wire's start and /or end points will be adjusted to be at the center of the connecting pad. The log file will contain an entry for all bond wires with end points that were adjusted, but no DRC will be created in the design if the tool corrected the error automatically.

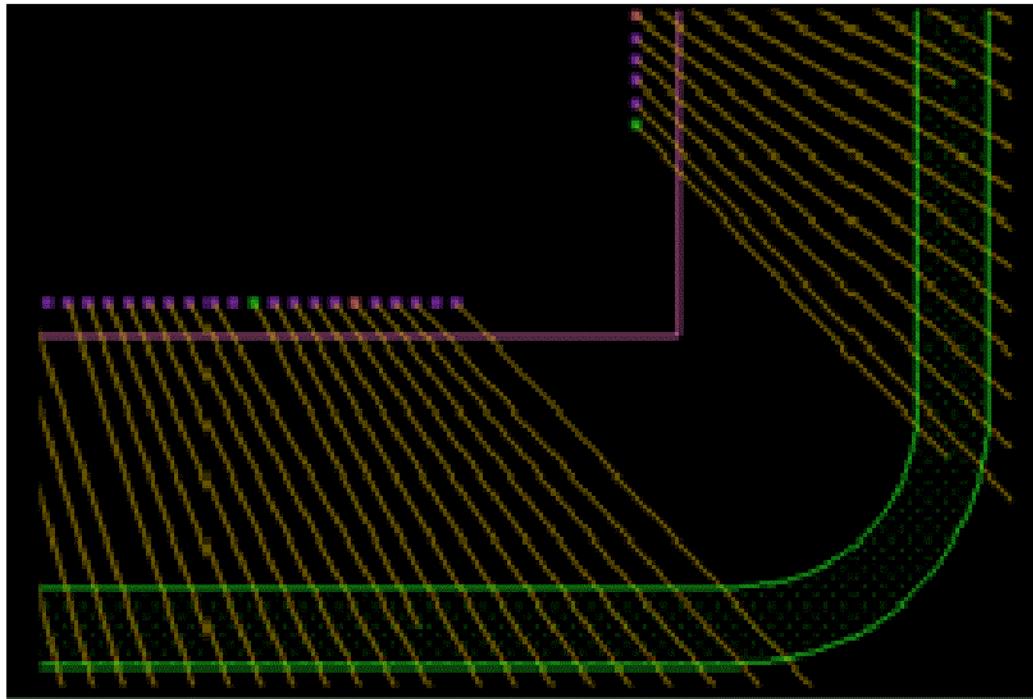


Getting Started with Physical Design

Managing Allegro X Physical Databases--Saving – Partial Versus Full Database Consistency Checks

Pin Bond Wire Count

Most wire bonded dies require a single bond wire connection. In some situations, a different number of bond wires may be required, depending upon the pin's net assignment, bond wire material, or SI requirements. In designs with these wire count requirements, this check will look at all wire bond die pins and validate the actual wire count against the specified required value stored in the `WIRE_COUNT` property on the pin. If the two values do not match, this is flagged an error. Any pin which does not have a `WIRE_COUNT` property attached will not be checked. It will be assumed that the pin requires either zero wires (as with dummy or NC net) or one wire (standard net), and the customer will see a rat line indicating the missing connection in this case. Only bond wires connecting to objects further down the die stack/substrate are counted. A bond wire from the top die in a stack to the bottom die in the stack is counted only for the pin of the top die. The tool cannot correct this for you automatically. You must add the appropriate bond wires to the pin using the wire bond commands. When finished, run this check again to verify the appropriate corrections have been made.



Power/Ground Ring Configuration	For a shape to be allowable as a power or ground ring during wire bond operations, the shape must be on the nearest exposed package substrate layer, and must be assigned to a power or ground net (net with <code>VOLTAGE</code> property). If either of these conditions is not met, then the shape will not be considered as a ring, and will subsequently be ignored during bonding operations. In most cases, this situation occurs if power and ground nets are not properly identified when they are created. This can be remedied by adding the <code>voltage</code> property (with the correct value) using the <code>Edit – Properties</code> (<code>property edit</code>) command or the <code>Logic – Identify DC Nets</code> (<code>identify nets</code>) command. If the ring is not on the correct layer in the substrate, this can be fixed using the <code>Edit – Z-Copy</code> (<code>zcopy shape</code>) tool. Because the tool cannot automatically determine the correct net, layer, or voltage assignment, this error cannot be fixed automatically.
---------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Adding Checks Using SKILL Functions

You can add new categories and rules to this tool and have them operate as if they were part of the standard tool. The four SKILL functions are described below.

 You need a SKILL development license if you plan to use the SKILL functions for debugging purposes.

packageDesignCheckAddCategory

NAME

```
packageDesignCheckAddCategory
```

SYNOPSIS

```
packageDesignCheckAddCategory (t_name t_bitmap t_description)
```

`==> defstruct defining category.`

FUNCTION

This function will register a new category inside the

IC Packaging tools' "package integrity" command check tree.

You must define a category before adding checks to it. So, this

function should always be called prior to `packageDesignCheckAddCheck`.

A newly added category will be inserted into the tree in alphabetically

sorted order. Therefore, you do not need to manage the order categories

are added by yourself.

NOTE: If the category name already exists, it will not be redefined.

NEEDS

`t_name` – Name of the category of rules as it should appear in the

user interface. This name should be used when calling

`packageDesignCheckAddCheck` to add specific rules.

```
t_bitmap - Name of the bitmap file which should be shown when this rule category is active in the user interface. This should be a full path to the bitmap or else the bitmap must be resolvable through BMPPATH.
```



```
t_description - The description to be displayed in the GUI when this category is highlighted.
```

RETURNS

Skill defstruct defining the category.

SEE ALSO

packageDesignCheckAddCheck

NAME

packageDesignCheckAddCheck

SYNOPSIS

```
packageDesignCheckAddCheck (
```



```
    t_category t_name t_bitmap t_description
```



```
    s_runCommand g_fixable
```



```
) ==> defstruct defining check.
```

FUNCTION

This function will register a new check in the specified category of

the IC Packaging tools' "package integrity" command check tree.

You must define a category before adding checks to it. So, this

function should always be called after packageDesignCheckAddCategory.

A newly added check will be inserted into the tree in alphabetically

sorted order. Therefore, you do not need to manage the order checks

are added by yourself.

s_runCommand is the skill function which should be called if this

rule is selected to run. This function MUST adhere to the following

guidelines:

#1: It must take exactly one argument, which is whether to

fix errors it encounters or not.

#2: It must return an integer value of the number of errors that

were found in the database.

#3: It must call the following functions:

packageDesignCheckLogError(<error string>) and

packageDesignCheckDrcError(<error location>)

to report any errors it finds.

These restrictions are imposed to ensure that output is consistent

across all checks run by this command interface.

NEEDS

t_category - Name of the category this check should be placed under

in the user interface tree. This should be the same name as

sent to packageDesignCheckAddCategory.

t_name - Name of the check as it should appear in the

user interface. This will be the name given to the rule in the

resulting log file, and will be the description for any

external DRCs created.

t_bitmap - Name of the bitmap file which should be shown when this

rule check is active in the user interface. This

should be a full path to the bitmap or else the bitmap

must be resolvable through BMPPATH.

t_description - The description to be displayed in the GUI when this

check is highlighted. This description will also be printed to

the log file ahead of any violations found for this check. As a

```
result, the description should be as descriptive as possible in  
  
order to maximize its usefulness.  
  
s_runCommand - A symbol representing the function to be called to  
  
check this rule. See FUNCTION description for details about  
  
the required format and return value of this function.  
  
g_fixable - Boolean flag to tell the user on the interface whether  
  
problems found by this check can be automatically fixed or not.
```

RETURNS

Skill defstruct defining the check.

SEE ALSO

packageDesignCheckAddCategory

packageDesignCheck.LogError

NAME

packageDesignCheck.LogError

SYNOPSIS

```
packageDesignCheck.LogError(t_errorString g_fixed g_location)  
  
==> nil
```

FUNCTION

This function will log an error found by this function to the log
file if the log file is enabled. By using this interface, you
are ensuring that the API will format your message consistently.

NEEDS

```
t_errorString - String to be printed to the log file. This should  
  
have all variable substitutions already done and be a simple  
  
string. This function will take care of any formatting necessary.  
  
g_fixed - Boolean indicating whether the error was fixed by the  
  
tool.
```

g_location - Location where the error occurred, if applicable. This is appended to the log entry for you and is used to let the user zoom to the error location in the design. If the location is unknown or not applicable, pass nil for the location.

RETURNS

nil

SEE ALSO

packageDesignCheckAddCheck
packageDesignCheckDrcError

packageDesignCheckDrcError

NAME

packageDesignCheckDrcError

SYNOPSIS

```
packageDesignCheckDrcError(l_location g_dbids)  
==> nil
```

FUNCTION

This function will create an external DRC marker for an error found by the currently running package integrity check. The tool itself will track the rule being run so that it knows the name to use for the rule violation.

NEEDS

l_location - Location at which to place the DRC marker.
g_dbids - Optional list of database object ids which are associated with this error. Usually 0-2 objects are affected.

RETURNS

nil

SEE ALSO

[packageDesignCheckAddCheck](#)

[packageDesignCheckLogError](#)

Configuration

This appendix explains how to configure the Allegro layout editors after you install the Cadence software. For additional information, see the "What's New" document.

- [UNIX-Based Installation Directory Information and Troubleshooting](#)
 - [Files That Reference the Installation Directory](#)
 - [Checking File References to the Installation Directory](#)
 - [Automatically Correcting Installation Directory References](#)
- [Windows-Based Installation Directory Information](#)
- [Licensing Issues](#)
- [Compatibility for Libraries and Designs](#)
- [IBM DFS](#)

UNIX-Based Installation Directory Information and Troubleshooting

After you install Cadence software, configuration files are modified to reference the installation directories you specified during installation. This section describes:

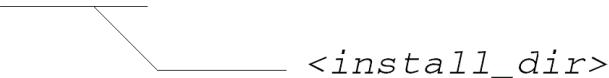
- The files that reference installation directories include
- How to verify the files are edited correctly
- A sample cshrc file
- A sample profile file
- How to automatically edit incorrect references

Files That Reference the Installation Directory

When you install the layout editor using softload , the following files are automatically edited to reference the installation directory you specified during softload :

- <install_dir>/tools/pcb/bin/cshrc
- <install_dir>/tools/pcb/bin/profile

where <install_dir> is the actual installation directory you specified during softload . For example, <install_dir> could be /usr/cds , as in the following example:

/usr/cds/tools/pcb/bin/cshrc

 <install_dir>

Checking File References to the Installation Directory

If you have problems running the layout editor after installation, verify that the following variables correctly reference the installation directory in Linux platforms:

- In the <install_dir>/tools/pcb/bin/cshrc file
set CDS_ROOT = <install_dir>
- In the <install_dir>/tools/pcb/bin/profile file
CDS_ROOT=<install_dir>

If the cshrc and profile files contain incorrect references to the installation directory, you can edit them individually, or run the setup_Allegro/APD script to automatically correct all three files, as described in the following section.

Automatically Correcting Installation Directory References

During installation, the setup_Allegro script saves the original versions of the cshrc , profile , and Allegro/APDHelp files in their respective directories. These original files have the extension .fcs

- <install_dir>/tools/pcb/bin/cshrc.fcs
- <install_dir>/tools/pcb/profile.fcs

The setup_Allegro/APD script edits these original versions to create the following files:

- tools/pcb/bin/cshrc

- tools pcb/bin/profile
- tools pcb/text/Allegro/APDHelp

Do not edit the original .fcs files. The setup_Allegro/APD script does not run correctly if these files have been modified. You can, however, edit the resulting cshrc , profile or Allegro/APDHelp files.

To use setup_Allegro/APD to place the correct install_dir directory references in the cshrc , profile and Allegro/APDHelp files:

1. Change directories to the installation directory:

```
cd <install_dir>
```

The installation directory is the directory immediately above the tools directory.

1. Type

```
tools/pcb/bin/setup_Allegro/APD <platform> [<path>]
```

where platform is one of the following platform specifications:

sun4v Sun Sparc running Solaris

hppa HP Precision Architecture running HP-UX

ibmrs IBM rs6000 running AIX

wint Windows NT (Intel)

path Is the path to the specified install directory, install_dir . This enables you to run the layout editor from a directory other than the one where the layout editor is installed. If not specified, install_dir defaults to the current working directory.

For example, if your administrator installed the layout editor on

AServer:/cds/9504

and you have Network Filesystem (NFS) mounted the installation directory on your workstation as

Yourworkstation:/usr/cds

the paths in the installation directory references the path /cds/ 9402 rather than your local path of / usr / cds . Portions of the Allegro software do not run correctly unless your local path to the software matches the path specified in the configuration files.

To resolve this, do one of the following:

- Mount the server directories on your workstation just as they are installed on the server.
mount AServer:/cds/9504 /cds/9504
- Create a symbolic link on your workstation that points from the installation directory to your local NFS mount point.
ln -s /usr/cds /cds/9504
- Have your administrator modify the configuration files on the server to match the directory paths on your workstation.

On the host where the software is installed (AServer), change directories to / cds / 9504 and enter the following command:

```
tools/pcb/bin/setup_Allegro/APD sun4w /usr/cds/  
                                      <install_dir>
```

The default working directory for the install_dir becomes / usr / cds .

Example cshrc File

The following is a sample of the tools/pcb/bin/cshrc file configured for an install_dir specified as / usr / cds .

```
set CDS_ROOT = /usr/cds  
  
set TOOLSDIR = $CDS_ROOT/tools  
  
setenv ALGROPATH      $TOOLSDIR/pcb/bin  
  
set path = (          $TOOLSDIR/bin \  
                         $ALGROPATH \  
                         $path )
```

Example profile File

To use a sample local profile file (~/.profile)

- At the Korn Shell command line, type

```
CDS_ROOT=/usr/cds  
  
TOOLSDIR=$CDS_ROOT/tools  
  
ALGROPATH=$TOOLSDIR pcb/bin  
  
PATH=$TOOLSDIR/bin:$ALGROPATH:$PATH;  
  
export PATH
```

 Do not set the TELEENV variable if you use the standard Cadence installation hierarchy.

Displaying UI Dialog Boxes Correctly

If the secondary (child) dialog boxes disappear behind the main UI of the layout editor, you need to modify the window manager to keep child windows on top.

- For HP and AIX
The typical window manager default configuration is

```
secondariesOnTop:True
```

- For Solaris
The typical window manager default configuration is

```
secondariesOnTop:False
```

If you run CDE, add the following to your ~/.Xdefaults file

```
DTwm*secondariesOnTop:True
```

If you want to restrict this behavior to certain programs, add the following to your ~/.Xdefaults file

```
DTwm*<program>*secondariesOnTop:True
```

For example:

```
DTwm*Allegro*secondariesOnTop:True
```

Add an entry to the file for each program. When finished, restart the window manager.

Windows-Based Installation Directory Information

The Allegro software directory structure for Windows is identical to that on UNIX platforms. However, Windows does not install chsrc or profile files.

Licensing Issues

For detailed information on licensing Cadence products for Unix and Windows environments, see the installation manual that accompanies the CD-ROM.

Compatibility for Libraries and Designs

Symbol Library and Padstacks

All library symbols created with previous releases of Allegro software are compatible with this release. Before loading the generic library, be sure that customized component symbols are not in the same directory in which you load the layout editor.

- ! Loading this generic the layout editor library overwrites the symbol library. The CAE libraries will not be affected. Before using any the layout editor library symbols, carefully review them to ensure that they meet your physical design criteria.

IBM DFS

Distributed File System (DFS) is an IBM networking protocol. Before Release 10.0, databases could become corrupted if the volume that contained the database became full during the database save. This corruption was due to a deficiency in DFS, not the layout editor.

Starting in Release 10.0, the editor detects that a volume is DFS and takes appropriate action to ensure the database is written correctly. Unfortunately, this means that writing databases across the network takes about twice as long as in pre-10.0 versions because the editor verifies the data as it is written to disk. Database reads are not affected.

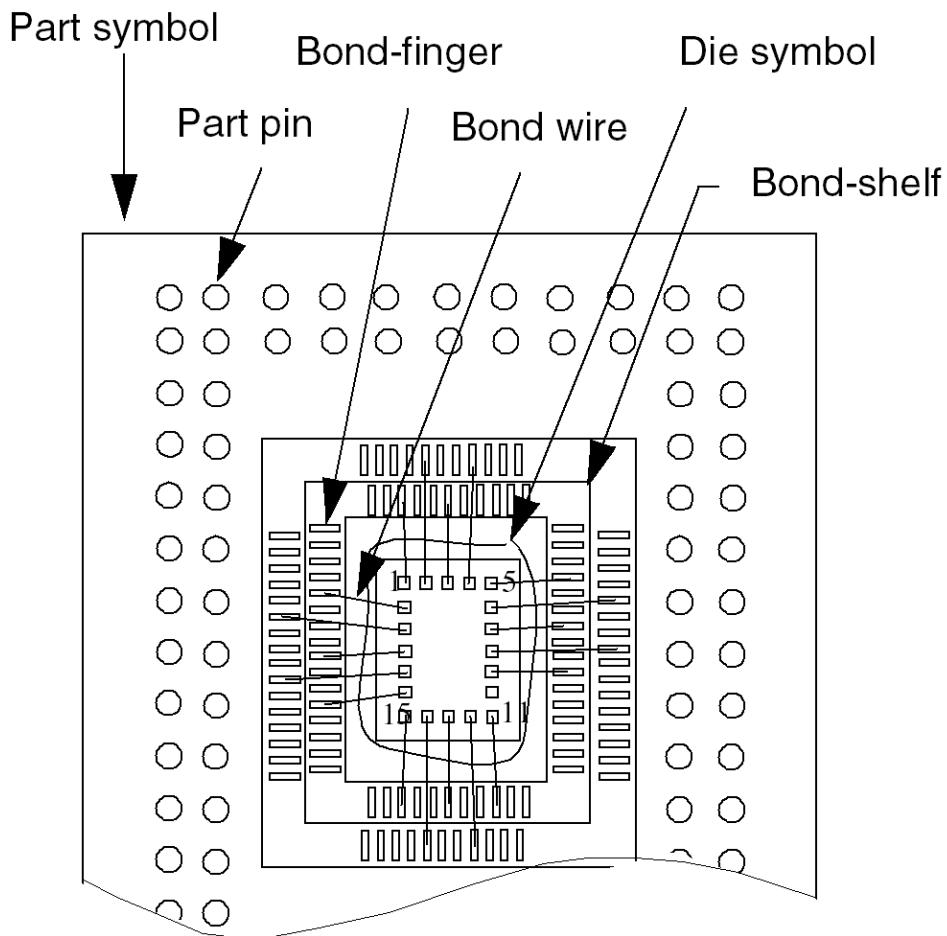
You can disable this DFS safety feature by setting the `afs_nosync` environment variable. With this variable set, databases save as fast as in previous releases, but the file write safety check is not performed.

By default, this additional checking takes place automatically as long as the system uses the standard "/dfs" naming convention.

Component Design Methodology for Allegro X Advanced Package Designer

This appendix presents an overview of the strategies and considerations required to successfully complete a component design using a hybrid design environment consisting of an Electrical Computer Aided Design (ECAD) system coupled with a Mechanical Computer Aided Design (MCAD) system. These strategies and considerations vary as component design requirements vary from company to company. The information contained in this appendix is independent of any specific component designer tool. This appendix focuses on issues of component design and die-to-I/O routing.

A Typical Component



Problem Statement

Design technologies for single/few chip packages vary greatly from company to company and from industry to industry. Depending on certain needs, substrates can be laminate-based (organic), ceramic, thin film (silicon), or a combination of materials. These needs may be based on cost, performance, signal, and thermal considerations.

Although packages have historically been designed using mechanical-based tools, such as AutoCAD, advances in technology have been pushing the limitations of such a system.

Some of these emerging technology issues include:

- Higher operating frequencies
- Larger die area
- Faster rise times

- Lower operating voltages
- Increased I/O points
- Increased I/O density
- Increased bus widths and speed
- Increased power dissipation

These trends directly relate to increased concern over high-speed transmission line effects, signal noise, thermal management, and increased component interconnect densities.

To combat these issues, you need to consider performance characteristics not only for the IC, but also for the component. Thus, packages are now being custom-designed to meet specific performance criteria. Without these new, more complex component solutions, designers are finding it difficult to optimize their system designs.

Acronyms Use

Use this table as a guide to acronyms mentioned throughout this appendix.

Acronyms	Expanded	Defined
Acronym	Expanded	Defined
ASCII	American Standardized Code for Information Interchange	A commonly used text file format
ASIC	Application Specific Integrated Circuit	A custom designed chip
BGA	Ball Grid Array	An array of chip connections, using surface-mount technology, in the form of solder balls
D&D	Delay and Distortion	Undesirable parasitic and coupling effects associated with signals within packages
DIE Format	Die Information Exchange	An ASCII text file containing IC footprint and connectivity data

DRC	Design Rule Check	Continuously checking for electrical and spacing violations based on predefined constraints
DXF	Data Exchange Format	AutoCAD's native exchange format
ECAD	Electronic Computer Aided Design	An environment well-suited for designing electronics
EDA	Electronic Design Automation	Software-driven design of electronic entities
EMI	Electro Magnetic Interference	Undesirable radiation of electrical energy within, or external to, the component device
FPMH	Failures Per Million Hours	A commonly used reliability metric
IBIS	Input/Output Buffer Interconnect System	An ASCII text file containing electrical modeling data
IC	Integrated Circuit	A die of silicon and its enclosure
I/O	Input/Output	A point of transfer commonly associated with die-to-die or die-to-component interconnection
MCAD	Mechanical Computer Aided Design	An environment well-suited for designing mechanical entities
MCM	Multi Chip Module	A packaging entity that encompasses multiple die
MTBF	Mean Time Between Failure	A commonly used reliability metric
PGA	Pin Grid Array	An array of chip connections using through hole technology
PLCC	Plastic Laminate Chip Carrier	A surface-mount packaging enclosure with leads extending out of the sides of the component
QFP	Quad Flat Pack	A surface-mount packaging enclosure with leads extending out of the sides of the component
RLC Model	Resistive, Inductive, Capacitive	A parasitic-matrix modeling technique
RLGC Model	Resistive, Inductive, Conductive, Capacitive	A parasitic-matrix modeling technique

SSN	Simultaneous Switching Noise	Reflection that triggers neighboring signals
-----	------------------------------	----------------------------------------------

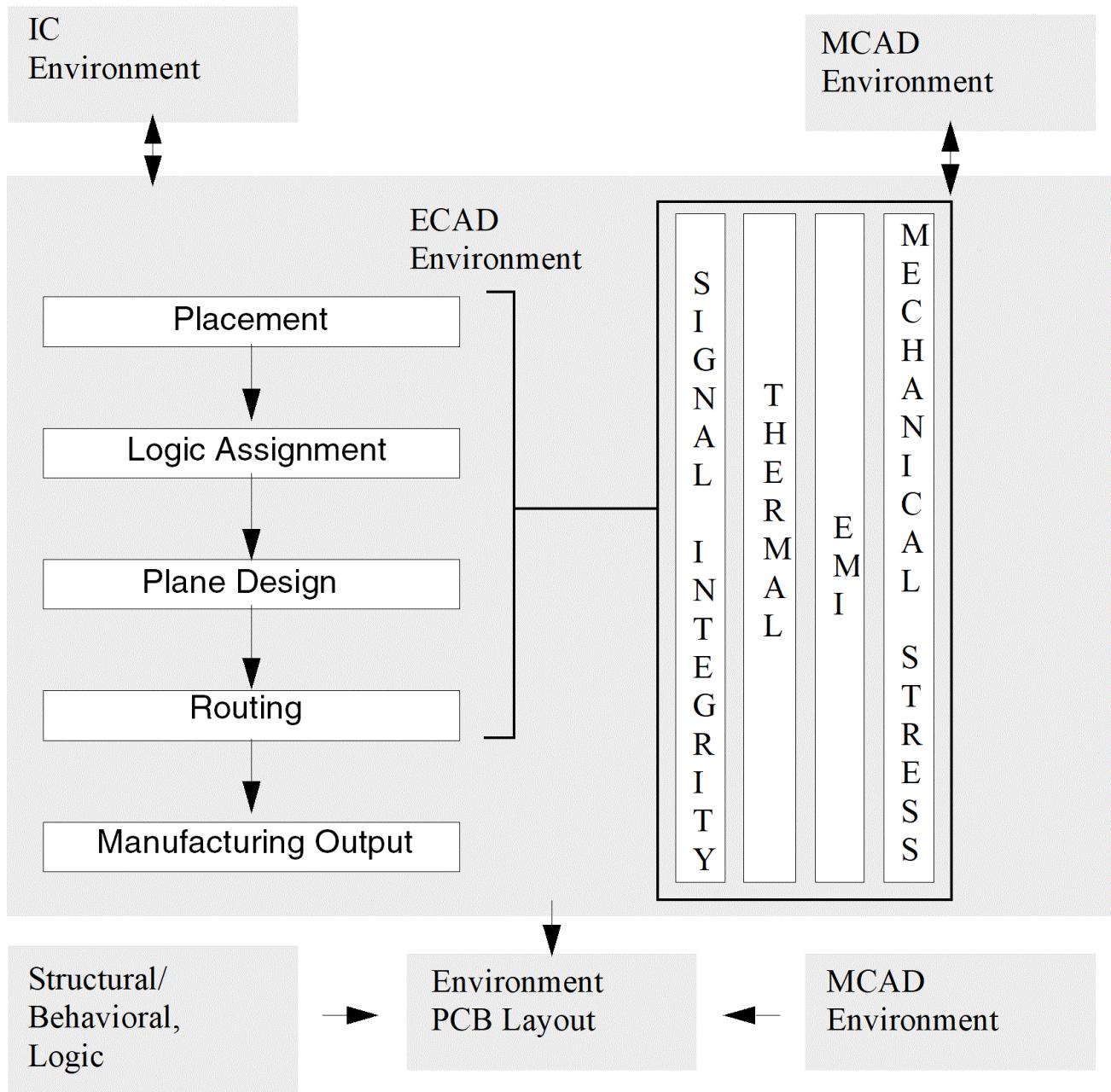
A Structured Approach

The methodology is divided into phases. Depending on the characteristics of your particular design requirements, certain phases may not apply or the order in which you go through the phases may change.

1. Planning and Trade-off Analysis
2. Mechanical Data Transfer
3. IC-to-Component Logic Data Transfer
4. Substrate Definition
5. Constraint Definition
6. Placement
7. Logic Assignment (die-to-pin)
8. Pre-route Analysis
9. Plane Design
10. Routing
11. Post-route analysis
12. Manufacturing Output
13. Component-to-Board Transfer

The "[Typical Design Flow Schema](#)" depicts how each phase of the component design process spans a particular design environment: MCAD, ECAD, PCB layout.

Typical Design Flow Schema



The balance of this appendix discusses the following:

- [Designing the Physical Component](#)
- [MCAD-to-ECAD Data Transfer](#)
- [IC-to-Component Transfer](#)
- [Substrate Definition](#)

- Constraint Definition
- Placement
- Thermal Analysis
- Die-to-Component I/O Net Assignment
- Pre-Route Signal Integrity Analysis
- Power and Ground Plane Definition
- Routing
- Post-route Signal Integrity Analysis

Component Design Considerations and Trade-off Analysis

During the silicon design phase, you must specify component requirements. You should consider items such as size, cost, thermal performance, electrical performance, materials, and mounting technology. Using a three-dimensional MCAD environment, coupled with a robust ECAD environment, you can characterize packaging design alternatives through form-factor and performance trade-off analysis. See the "[Typical Design Flow Schema](#)" for a graphical representation of a hybrid design environment.

The size for the component is a key area to research. This greatly depends on a number of factors. The size of the die and the I/O interconnect density are probably the biggest factors to consider. These determine the minimum size for the substrate and an approximate (depending on the number of power/ground I/Os that are needed) calculation of the number of I/O pins for the component.

With this information, you must decide on the appropriate packaging technology — Pin Grid Array (PGA), Quad Flat Pack (QFP), Plastic-leaded Chip Carrier (PLCC), Ball Grid Array (BGA) — that is optimal for the system that it is being designed.

Of course, you should also consider the type of system for which the component is being designed. For example, if a component is used for a small portable product, then a BGA may be a good choice, whereas if it was being designed for a desktop computer, then a PGA may be more appropriate.

Component Cost Considerations

In considering the total cost involved with choosing a component technology, substrate cost is only one of many factors. Other factors include costs associated with electrical and thermal performance requirements. For example, a more expensive heat sink may be required for a laminate substrate versus a ceramic substrate.

Choosing a component technology involves a cost trade-off analysis. For example, QFP is typically cheaper to manufacture than BGA. However, if the die I/O count is beyond too many pins, then QFP becomes impractical due to mounting problems when the component is mounted on a PCB.

A PGA is another alternative, but as pin counts increase, PGAs become very large, increasing interconnect length and density. This may introduce unwanted electrical noise within the component. A BGA may be a better choice because they are small and reliable for board mounting, however, they may be more expensive to manufacture. However, as manufacturing technology matures, BGA substrates cost is beginning to decline.

Laminate is perceived as the cheapest ceramic in the mid-range, with thin film perceived as the most expensive substrate type. Although this has been true in the past, things are changing. An infrastructure has been built to support large manufacturing for ceramic, and in many instances, ceramic can be cost competitive with laminate technologies.

Laminates are now beginning to support very fine line widths, spacings, and smaller vias. Thin films are still a specialized area for the very performance-driven applications, such as military and microwave applications, since the electrical characteristics for the materials and interconnect rules provide for optimal performance. Again, depending on specific requirements, different component technologies offer different benefits at various costs.

The most effective way to approach your component technology selection is to research design requirements for electrical and thermal performance, contact and investigate foundries specializing in various technologies, and match the information obtained against your specific performance requirements. This should enable you to begin performing cost trade-offs without sacrificing performance.

Electrical and Thermal Considerations

You need to consider electrical and thermal performance needs when planning packaging choices. Ceramics offer the best thermal characteristics but the worst electrical performance. Laminates provide good electrical performance but are poor conductors of heat, and Thin Films offer the best electrical performance but are typically more expensive.

To remedy deficiencies within the various materials, mixed technologies have been introduced to provide a reasonable compromise to both. Mixed thin film/ceramics have been used for very high power/high performance applications, such as mainframe computers. Laminates offer more options for heat sinks for heat dissipation, such as a Eutectic bond from the heat sink directly to the die.

Foundry Constraint Considerations

Prior to actual component design, you must establish design criteria. Many of these design constraints are based on mounting technology, material, size, foundry specifications, and electrical performance. These include line widths and spacings, layer stackup, via type (bbvia or through), sizes and spacings, bond wire length, and electrical thresholds, such as delays, crosstalk, and reflection.

Many times, foundries, once selected, offer design guidelines to produce packages with high yields. These design guidelines are sometimes in the form of Design Kits, which offer you an environment which automatically sets up the appropriate manufacturing rules and guides you through the steps to successfully complete the design task.

Foundries also offer complete design services, often for free depending on the volume of manufacturing. If you decide to design packages in-house, you must still consult with the manufacturing foundry to establish design rules that meet both the electrical, as well as manufacturing requirements. If not, the foundry may decide not to manufacture your design. Electrical threshold rules must be dictated by your design requirements.

Summary

Packaging choices can be complex decisions based on trade-offs among cost, size, electrical and thermal performance, mounting, and materials. The decisions made here, however, drive the design, electrical, and manufacturing constraints for the physical layout. Again, the vendors chosen to manufacture the component substrate establish the boundaries for these constraints and, in most cases, supply detailed specifications or electronic files for use as design constraints.

Designing the Physical Component

Introduction

At the completion (or near completion) of the silicon design phase, you should finalize physical and electrical characteristics of the die to the point where physical component design can begin. At this point, you should also decide upon the component materials, size, mounting technology, vendors, and some constraints before continuing. In some environments, continual trade-off analysis is required to eventually arrive at a few optimized alternatives.

Information critical to begin the physical component layout includes die physical and electrical data, component, plating bar (if used), and stackup information. The pieces of data for each of these areas are derived from various sources. (See "[Typical Design Flow Schema](#)".) Tight integration

between the various sources with the ECAD system is critical for efficient design. Without it, design groups spend hours and sometimes days attempting to pass or rebuild critical information.

This remaining sections focus on building the physical characteristics of the chip carrier component. The component could be a PGA, QFP, BGA, Chip Scale, or a customized enclosure. It is the function of the *mechanical design group* to characterize the detailed physical aspects of the component and provide the detailed drawings to *manufacturing* as part of the product documentation component.

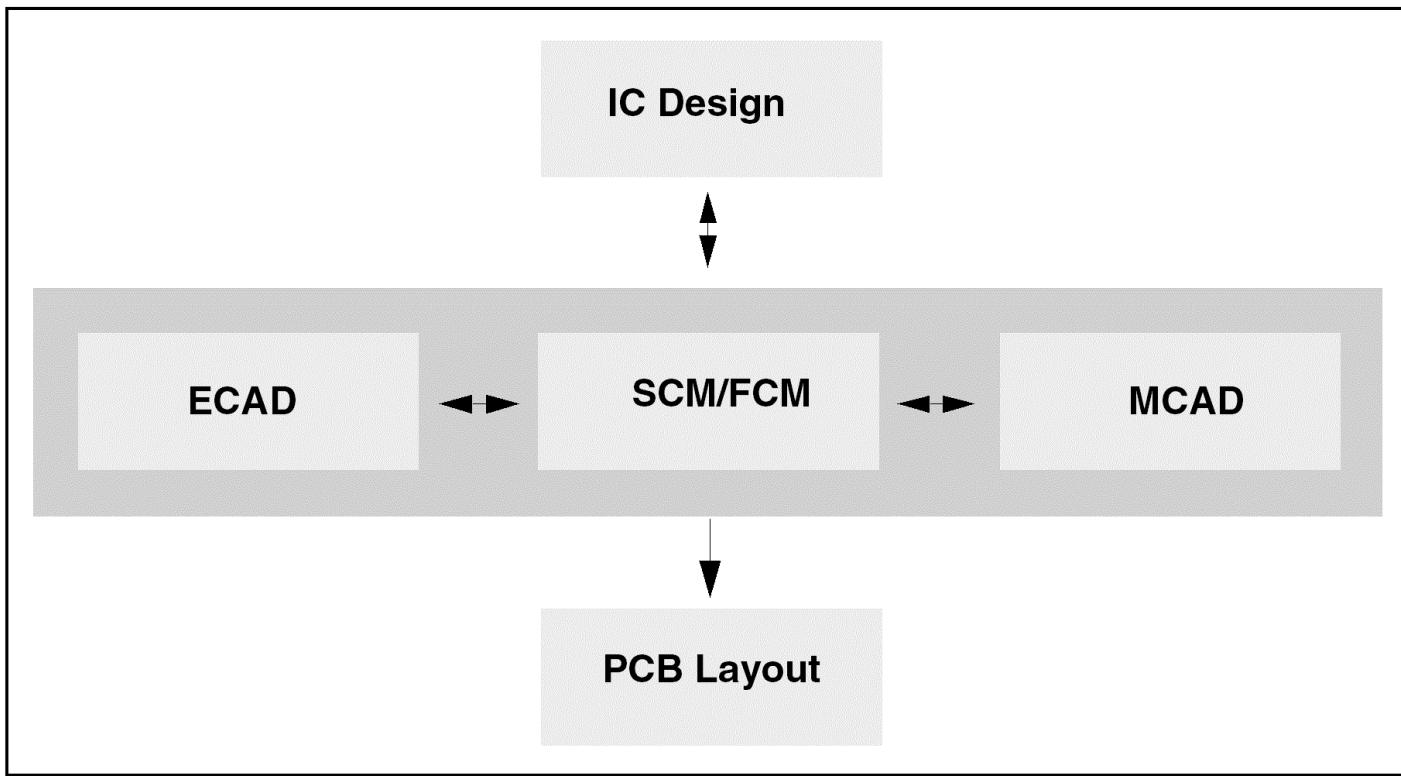
This product documentation component contains component width, height, thickness, I/O pin size and location, seal data, cavity data, and any special notes that may impact the design or manufacture of the finished component.

Problem Statement

Historically, MCAD packages have been used to design the electrical layout of the component. Due to deficiencies in an MCAD environment for electrical design, it is becoming increasingly difficult to meet the higher demands for today's more complex component designs. See "[MCAD-to-ECAD Data Transfer](#)" to learn about the limitations and concerns of component design solely in an MCAD environment.

A Hybrid Solution

To address these shortcomings, the best possible solution is to use a hybrid environment – an ECAD tightly coupled with an MCAD. An ECAD environment is well-suited for electrical design, modeling, and analysis. However, ECAD systems often lack the robust set of mechanical capabilities of an MCAD environment. Since SCM/FCM packaging demands the capabilities of both, it is advantageous to establish an environment where both can coexist.



MCAD-to-ECAD Data Transfer

Mechanical-based applications are mature and capable tools for modeling 3D elements and performing mechanical detailing. However, much of the data elements necessary to meet the demands of new packaging designs are not available in an MCAD database. This makes it increasingly difficult to handle the high logic content, interconnect densities, analysis, and routing.

Component Information

Because packages are three-dimensional elements, a mechanical design environment, such as AutoCAD, is typically used to develop the detailed models and manufacturing detail drawings. AutoCAD's format has been an established standard for many years and has provided for reliable data transfer between mechanical (MCAD) and electrical (ECAD). The DXF file can handle physical geometries of the component, pad information and locations, and mechanical detail information. An ECAD system should be able to read this format and automatically generate the appropriate footprint with all pad information. This again eliminates unnecessary time rebuilding footprint information.

Specific limitations of an MCAD system for component design are the lack of:

- Electrical connectivity information
- Electrical/thermal characterization and modeling of the substrate
- Detailed stackup information
- Online Design Rule Checking (DRC)
- Modeling for electrical elements, such as I/O pins, die, wire bonds, conductor traces, via punches, power and ground planes, and IC device level modeling (driver/load characteristics)
- Interactive or automatic intelligent netlist routing
- Die-to-component connectivity transfer
- Constraint support

To address these limitations, a new methodology for component design must be implemented to address electrical complexities while maintaining a tight link to a mechanical environment, which is more suited to handle form factor issues.

Information Transfer

The first process in this marriage between the MCAD and ECAD environments is to transfer the mechanical design database from the MCAD to the ECAD environment.

The DXF Medium

The DXF file, generated by AutoCAD and most other MCAD tools, is the most universally accepted format for information transfer. The DXF file format was developed specifically to meet the data transfer requirements between the two environments, starting with PCB design tools back in the late 1970s. This now mature data transfer method can be effectively used for the same purpose in a component design environment. Most ECAD environments can import DXF-formatted files.

This transfer of data should result in a physical footprint for use in the ECAD system. This includes the component outline, cavity outlines, and pad data and locations. It may also include other information such as conductor traces, vias, drawing formats, and detail information.

IC-to-Component Transfer

Die Information

The primary goal is to have a resulting CAD footprint that can be used for physical design. In most cases, the die size and outline, die pin description and location, and an electrical netlist for each of the pads are all that are necessary.

DIE Format

A new emerging ASCII file format called DIE (Die Information Exchange) is becoming the industry standard for passing this type of data, as well as electrical and thermal modeling information. DIE files should be supplied by the silicon design group or by the company from which the silicon is being purchased.

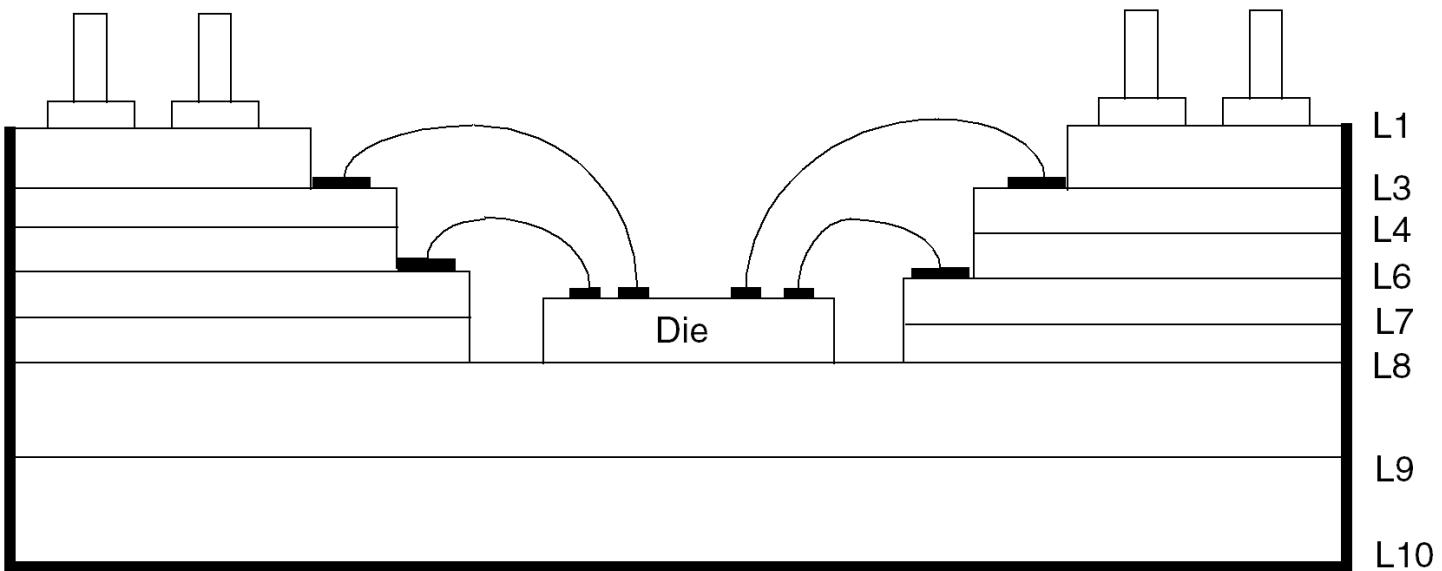
Data can be generated through the silicon design environment or through a commercial or internally developed translator. Once supplied to the component designer, the ECAD design tool should be able to read the DIE format and automatically generate all of the appropriate information for physical layout.

 The automation afforded by a DIE reader significantly reduces the hours spent manually rebuilding and verifying the accuracy of the footprint information.

Substrate Definition

Stackup information

The stackup information defines all the layers in the substrate that will be manufactured. This includes conductive layers, dielectric layers, paste layers, bondwires, pads, and shield planes.



The information necessary to complete an accurate stackup includes material, layer type, name, thickness, dielectric constant, and electrical conductivity.

The stackup of a component is important, not only for the physical characteristics, but also for electrical and thermal characteristics. With proper stackup construction, electrical and thermal simulations can provide a more accurate analysis of behavior. Without it, results become skeptical at best.

The stackup also provides the z-axis aspect of the electrical design, allowing for accurate positioning of conductive trace layers, power/ground layers, wire bond information, I/O pin position, and z-axis connections between trace layers or to a power or ground plane.

Information required for stackup modeling includes:

- Layer
 - Location
 - Material
 - Type (conductive, dielectric, shield)
 - Identifier or name
 - Thickness
- Electrical conductivity
- Dielectric constant

Layer Thickness

The overall thickness of the component is usually known and should be specified within the mechanical detailed information. Thickness is based on the number of layers required for the design. For example, a component may require 8 layers: 2 routing and 6 power/ground. The total thickness is the thickness of the 9 dielectric layers plus 8 layers of conductive material.

Individual layer thicknesses can be derived from the manufacturing data supplied by the foundry. You enter thickness data when you define the layer stackup.

Layer Materials

You should, at this point, know which materials to use for each of the conductive and dielectric layers. Material type is important to define the electrical characteristics of the component. You define these characteristics by specifying the electrical conductivity, and dielectric constant. However, you should obtain the exact specifications for the material through the manufacturing foundry, check them against the ECAD tool-generated values, and tune them to the manufacturing specification.

Layer Type

You specify a layer type to define the purpose the layer serves in the component design. Dielectric, Conductive, Plane, and Bond Wire are the most commonly used layer types, however, your design may require others. The significance of the layer type to the ECAD system is to specify effects (shield planes, design rule checking, and manufacturing output) for signal analysis.

Template Files

Providing stackup information results in more accurate thermal and signal analysis, design rule checking, and manufacturable routing. Once completed, stackup information can often be stored in an ASCII file format (template file) that is recognized by the ECAD system.

Template files store a wealth of information which can then be reused on other similar designs to reduce setup time and effort.

Constraint Definition

Introduction

With the physical modeling of the component substrate complete, the next phase is to set up constraints.

Constraints fall into two categories: physical and electrical. Physical constraints are driven by manufacturing guidelines, though Electrical rules may impact the physical rules. For example, an electrical rule may be a 50-ohm line impedance which translates into a 4-mil trace width.

Electrical rules are engineering-driven and are imposed to ensure signal quality and overall component performance. You can enter constraints through the ECAD user interface or through template files.

Physical Constraints

Physical constraints can be further broken down into two categories: Physical (Line and Via sizes) and Spacing (Line, Via, Pad, and shape spacings). Again, most of the physical rules are driven by manufacturing specifications, though some latitude may be given depending on the foundry.

You can also assign physical constraints to specific nets or groups of nets (net classes). Net classes allow you to specify different line widths, via sizes, and element-to-element spacing to the entire group or to a specific area layer of the component substrate.

Electrical Constraints

Electrical constraints can be divided into two categories that are commonly lumped together: delay and distortion (D&D). Delay refers to the interconnect delays introduced by the physical layout, typically in terms of nanoseconds (ns). Distortion refers to sources of noise caused by the physical layout, such as undershoot or crosstalk. Distortion is measured in millivolts (mV).

You should divide signals in the layout into unique net classes based on performance requirements, for example, clocks and buses. Each constraint set would have its own noise budget and, therefore, corresponding distortion (overshoot, undershoot, crosstalk, and so on) constraints. For each net class you also define timing constraints, such as delay and matched delay. In addition to defining electrical constraints, you should define any thermal constraints.

Template Files

The effort involved in setting up all of the required constraints in a design may seem cumbersome and time consuming. Technology files allow you to dump out an ASCII representation of all defined constraints, which you can then import into other designs.

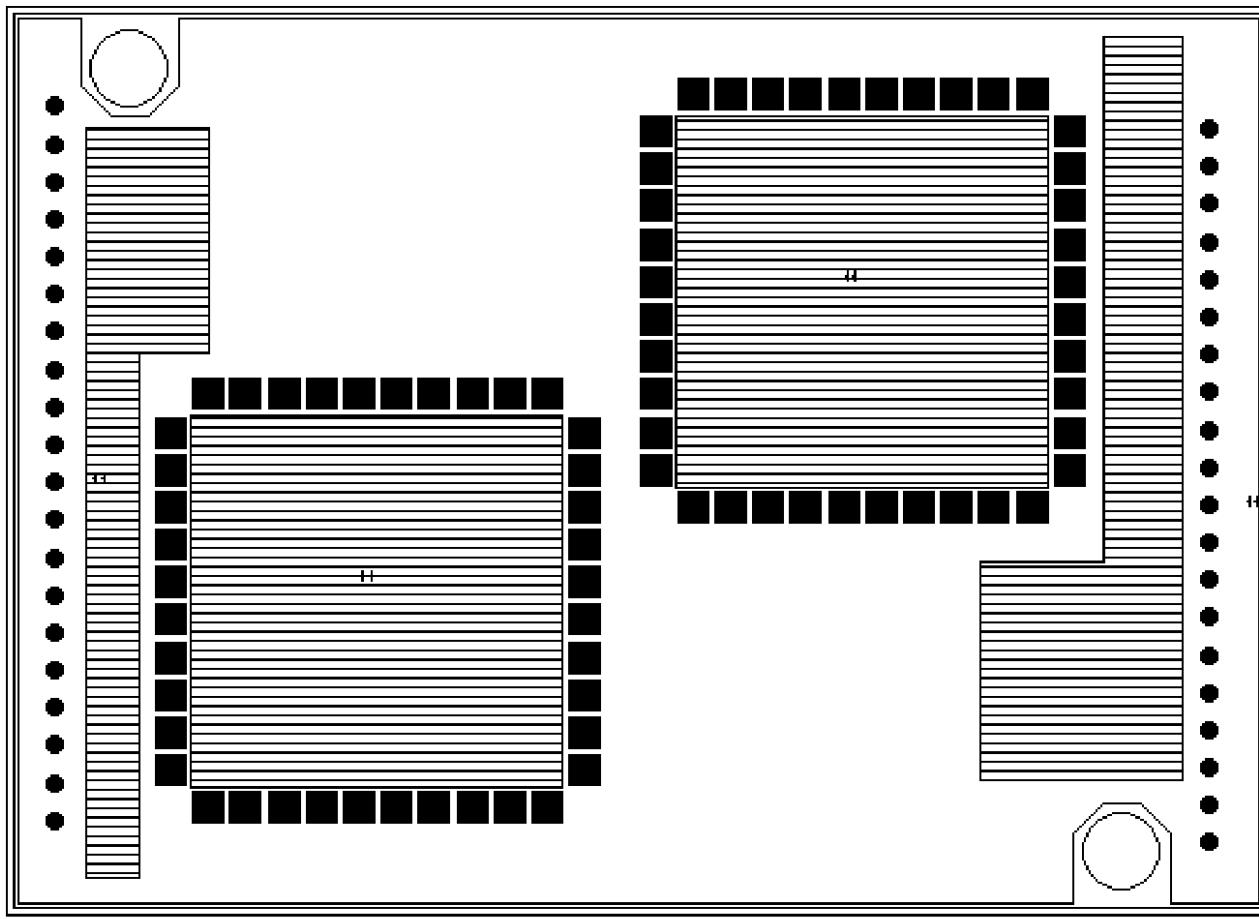
Technology files include:

- Net classes
- Physical constraints
- Electrical constraints
- Stackup Information
- Design size, units, and origin
- Special attributes and properties created within the design

Although every design has some unique requirements, template files can minimize the constraint definition effort by allowing you to modify an existing data file rather than starting a new file.

Placement

This phase is often very simple because there are few design elements to place, but sometimes can be difficult because of geometric restrictions of the substrate, such as cavities or geometric centering between all of the component I/Os.



If the component has multiple die, some latitude may be given to the designer for placement. In this situation, the designer is responsible for optimizing for die-to-die interconnect while at the same time optimizing for die-to-component fanout routing.

Die-to-Die Placement

For die-to-die placement optimization, the designer should attempt to minimize crossing of signals between die and the number of signals that pass through another die to reach its destination. To accomplish this, you should attempt to rotate dies, align dies that interconnect with each other, and place dies relatively close to each other to optimize the routing real estate. In most cases, however, component designers do not have much freedom in die rotation and placement so if you cannot achieve optimized routing with imposed placement, it is best to consult with the logic and manufacturing engineers before making changes.

Die-to-Component Placement

For die-to-component placement optimization, die should be placed as close to the geometric center of the component as possible. If multiple dies are used, then each die that connects to the component should be placed so that the dies' I/O is on the outside edges, close to the component edge, and no obstructions exist between the die and the component I/O.

You should carefully place signals that require performance constraints, such as delay or crosstalk, or require special routing. Acknowledging these requirements allows for optimized real estate to handle discrete components.

Thermal Analysis

Once you have placed components and you defined constraints and stackup, you can perform thermal analysis. In this phase, you use thermal analysis to predict the junction and case temperatures within the component being designed. Through thermal analysis, you can quickly identify component temperatures that violate constraint criteria.

You can rectify thermal violations by applying one or more of the following corrective measures:

- Modify die placement, if multiples are being used
- Add plane layers to the stackup
- Use alternative substrate materials
- Add thermal vias
- Add heat sinks
- Experiment with alternative boundary conditions (estimate performance under various environmental conditions)

You can apply these measures in multiple "what-if" scenarios to arrive at an optimal solution.

Since junction temperature also impacts buffer drive characteristics, this information is important for the Pre-Route Signal Integrity Analysis phase, described later.

Die-to-Component I/O Net Assignment

Introduction

At this phase of component design, the only logic in the design database is the die and, in the case of FCMs, the die-to-die interconnect. This provides for efficient component design since die-to-component logic can be optimized only after you have defined component description and placement. Without either piece, component I/O assignment becomes a blind exercise which results in poor interconnect efficiency.

With only die logic defined, you can optimize I/O assignment for routing and performance with minimization of interconnect length and logic criss-crossing.

Pin Assignment

The first step in optimizing pin assignment is to determine which component I/Os feed power and ground connections. Most companies preassign pins in a netlist. The power/ground pin assignment determines the remaining available component pins that can be used for signal assignment. Of course, there must be enough I/O pins remaining to accommodate the number of signal I/Os.

Priority Nets

Prior to signal pin assignment, you should identify critical signals as priority connections. Depending on performance requirements, these signals may need to be the "shortest possible distance" to the component I/O, in which case manual pin assignment may be required. Many ECAD systems allow you to attach a special attribute to a net that requires a priority connection.

Routing Concerns

Although the signal may require the shortest possible assignment, you must also consider routing. If the resulting shortest assignment results in connectivity crossing, you may need an additional routing layer. If an additional layer is not possible (or desired), then you should make an assignment that selects the closest component I/O that also minimizes any signal crossing.

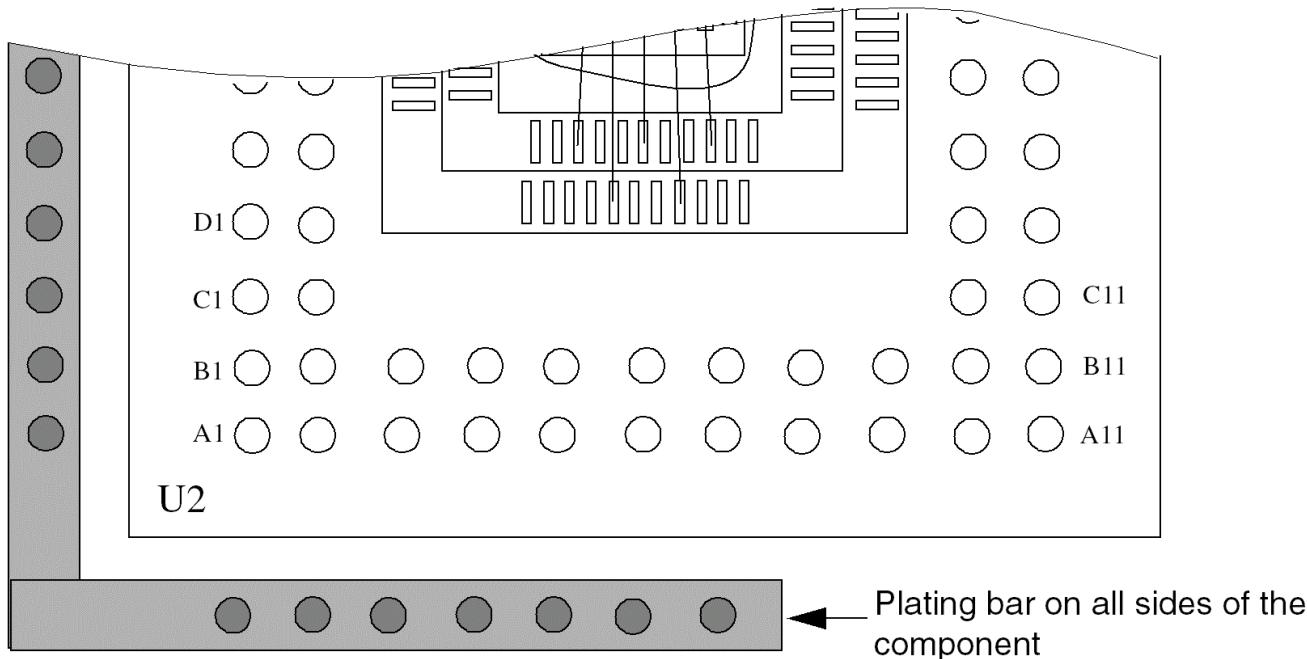
For general component I/O assignment, a utility should be available that will scan both the die and the component I/O, take into consideration power, ground, and priority signals, and develop an optimized die-to-component netlist. Optimization should be based on overall routability of the component. For multiple dies, this process may be incremental. During each step, you select the die I/O of one or more dies and direct the assignment to a specific location of the component.

Otherwise, poor assignment may result.

Plating Bar (Optional)

Only packages that require gold electroplating need plating bar connectivity. Typical items that require electroplating include component pins, bond wire pads, and die pins.

You accomplish the plating process by connecting all necessary elements to one common point, which is usually in the form of a bar around the periphery of the component, hence the name. In most applications, the plating bar connections are made through conductive traces from the component pins through the edge of the substrate outline to some point that intersects the plating bar geometry.



In an ECAD system, the plating bar may be made up of separated pins. This allows correct connectivity checking to ensure that signals are not shorted at any point to each other. The footprint for the plating bar may be designed in an MCAD component and transferred, through DXF; however, it is typically developed directly in the ECAD environment.

Pre-Route Signal Integrity Analysis

Pre-route signal analysis helps you get your component to the market sooner by identifying and correcting signal integrity and timing problems before you invest time and effort in routing the component. It can be increasingly costly and time-consuming to address these issues later on in the design cycle.

Interconnect routing is based on your assumptions for percentage of manhattan distance, characteristic impedance, and propagation velocity. To calculate thermal shift, you can incorporate temperature data derived from the thermal/reliability analysis. This also increases simulation accuracy, as temperature impacts the drive strength of output buffers. Some considerations for signal integrity analysis at this stage of component design include:

- Modeling Wire bonds
- Die-to-Die Connections versus I/O Connections
- Reflection and Delays
- Simultaneous Switching Noise (SSN)

Modeling Wire Bonds

From a modeling perspective, wire bonds require special consideration. In reality, wire bonds are three-dimensional structures, curving up from the die, then down to the die pins on the substrate. However, in a component layout database, wire bonds appear as flat, planar connections with ratsnest lines. While this correctly represents the connectivity, these simple point-to-point connections do not accurately represent the wire bond length. Also, extracting the parasitics of a wire bond is not a straight-forward procedure, because the three-dimensional wire bond is essentially represented as a two-dimensional entity in the layout.

Because of these issues, wire bonds are typically characterized up front, and modeled as a Resistive Inductive Conductive Capacitive (RLGC) parasitic matrix, with a particular wire length. Rather than extracting parasitics for the wire bonds in the layout, the wire bond's characteristics are captured in an electrical model, which is then assigned to the wire bond. This enables you to accurately simulate signals running through wire bonds.

Die-to-Die Versus I/O Connections

Simulating die-to-die signals in an FCM is straight-forward, as all the parasitic information required to model the interconnect is contained within the component layout database. However, for signals that run from a die to a component I/O pin, this is not the case. All SCM signals fall into this category, so this actually represents the majority of cases encountered in a component design.

To accurately model the behavior of an I/O signal:

- Simulate the entire component-on-PCB enclosure

—or—

- Estimate the off-component PCB connections

Simulation

To simulate an I/O signal connection, you link the component and PCB layout databases. This lets you trace an I/O signal through the component, onto the PCB, and to its final destination. Parasitics are then extracted for interconnect on both the component and PCB; one circuit is built and simulated.

This is the most accurate approach, as the actual interconnect on the PCB can be included in the simulation circuit. However, since you must have access to both the component and PCB layout databases, this approach is feasible for few engineers.

Estimation

Estimation, though less accurate than simulation, is more commonly employed. You must assume that the signal I/O pins are inputs, then model them as a single device including each lumped RLC (resistance, inductance, capacitance) parasitic that is associated with each signal I/O pin.

The goal is to estimate the characteristics of the corresponding routed interconnect on the PCB through these lumped parasitics. This approach works fairly well if the routed interconnect on the PCB is electrically short, but decreases in accuracy as the interconnect becomes longer.

The lumped parasitic approach works well when the following relationship holds:

$$T_r > 2 * T_d$$

T_r represents the rise (or fall, whichever is shorter) time of the driving signal from the die. T_d represents the propagation delay of the interconnect on the PCB, from the component-PCB junction to the receiver.

Reflections and Delays

Once you complete electrical modeling, your next step is to quickly scan the entire design and compare it against electrical constraints to identify marginal or failing signals. The rapid scanning process is a key time-saver because it is important to concentrate on problem nets first and not waste time on signals that initially meet their constraints.

First you need to identify and address all reflection and timing-related issues such as overshoot, undershoot, and interconnect delay. You should incorporate interconnect delays from transmission line simulation into the component and board-level timing analysis. You must calculate slacks and skews and identify violations. The easiest and most natural way to analyze this data is by spreadsheet, where you can examine the results of many simulations.

Typically, reflection or interconnect delay is a function of the circuit topology. The circuit topology is comprised of several aspects of the physical interconnect:

- Net schedule
- Propagation delays
- Characteristic impedance
- Termination

Circuit topology for an I/O signal includes interconnect on the PCB as well. It is quite possible that a signal driving from a die on the component may require termination at corresponding loads on the PCB. You must modify circuit topologies to produce an optimum or acceptable result. This demands an iterative use model in which you employ quick "what-if" capability. Certain circumstances may dictate that you select an alternate driving buffer on the die. If so, you should communicate this back to the die supplier.

If you require special wiring rules or other changes for specific classes of signals, it is *much* easier to address this at the placement stage rather than after you complete routing.

Once you make all necessary edits to the layout, you should do the following before proceeding to the next phase:

- Re-scan the design and verify that all reflection-related issues are under control
- Rectify all interconnect-delay constraint violations
- Rectify thermal constraint violations

Simultaneous Switching Noise

When a number of drivers switch simultaneously in a digital system, a sudden change in current occurs through the power and ground connections to the die. Because of the parasitic inductance that exists in this path, any current change produces a temporary fluctuation in the power and ground voltages as seen by the die. This is typically referred to as Simultaneous Switching Noise (SSN), or Ground Bounce.

Simultaneous switching noise can cause noise at the output of non-switching drivers. This noise then propagates to loads on the net and potentially cause false switching.

- Overshoot and Undershoot (signal distortion) on output driver
- Switching delays on active drivers
- Reduction of noise margin
- Increased Electromagnetic Interference (EMI)

SSN Simulation

The pre-route stage is an opportune time to perform initial SSN simulation. This usually involves the following:

- Selecting a number of drivers to switch together
- Extracting a circuit which contains the drivers, the nets they are connected to, and the parasitics of the power and ground connections to the die
This includes the parasitics of the power and ground planes themselves.
- Simulating the circuit

Reducing SSN

The results of the simulation reveal the level of SSN in the component for that particular set of drivers. Any major groups of drivers that switch together should be simulated in this manner. You can reduce or eliminate SSN as follows:

- Stagger the switching times of banks of drivers (reducing the number of drivers that switch simultaneously)
- Reduce the current draw or edge rates of the drivers, either by using buffers with lower current drive, or by adding series termination
- Add decoupling capacitors to the design, providing a local charge supply for the initial current draw
- Use additional power and ground planes to reduce the effective inductance of the power and ground distribution system

Adding more power and ground planes significantly adds to the total cost, so you must examine this carefully. Scrutinizing the current densities on the planes is a prime consideration.

Reducing EMI

From an EMI perspective, it is critical to keep I/O signals out of "noisy" areas. For example, locating a signal via in a noisy area can cause "common mode noise" to couple onto the signal, which then causes the via to act as a radiating antenna as the signal travels off the component.

You should identify the high current density areas of the planes and carefully isolate I/O signal traces and vias from these "hot spots."

For extreme cases, you can use an isolation strategy. In this approach, you use a separate set of power and ground planes for core logic (associated with die-to-die signals) and I/O logic (associated with signals that travel off the component to the PCB).

Connections between these sets of planes exist in only a single area, where the core ground and I/O ground are shorted together. The power planes are handled in a similar manner. This is done with a strict decoupling and bypassing scheme.

Power and Ground Plane Definition

Introduction

Based on results from the pre-route signal integrity analysis (described in the previous section), you are now ready to define the power and ground distribution for the component (plane design).

During this phase of SCM/FCM design, you must define the plane regions for each layer and the type of plane (solid or crosshatched). You then assign the plane to the appropriate power/ground net.

Scope

Plane design can be simple or complex, depending on the type of component that is being designed or the application for which the component is being designed. For example, a standard ASIC component may have a single ground plane and a single power plane, while an FCM for a mixed-signal application may have split planes for analog and digital. For manufacturing concerns, other applications may require crosshatched instead of solid planes. In any case, because of its potential impact on signal integrity and EMI, plane design has become a critical aspect of high-speed component design.

Geometry

The physical geometry of the plane starts at a specified distance from the component edge. This distance is usually determined by the manufacturing foundry. If the layer contains a single plane shape, then the geometry becomes a simple shape. For split planes, you can determine the split line by the placement of die. By knowing the different power/ground nets feeding various die, you can determine the split line.

 A split plane should leave enough room so that it does not cause starvation, provides adequate isolation, and allows enough room for the power/ground feed-throughs.

Once you define the physical geometry of the plane, you can assign a net to the shape. This assignment is critical for accurate connectivity during the routing phase.

SSN Effects

In designing the planes, you must also consider the effects of SSN. You may need to distribute power and ground pins, attached to the same logic, across different layers as discussed in the previous section, [Reducing SSN](#). You must also determine the appropriate die pin to connect to each plane.

Editing Planes

When completed, each plane encompasses the entire layer. After routing, you must edit each plane so that antipad clearances and connections can be physically built into each plane. Therefore, you must know the appropriate antipad clearances and, for connectivity, the thermal relief configuration. For mesh planes, the antipad and connectivity geometries vary depending on the mesh type (horizontal or diagonal), and are defined by your manufacturing and/or engineering requirements.

Routing

Routing is the most time-consuming phase of component design. Routing tasks can be divided as follows:

- Wire bonds
- Component I/O
- Die-to-die
- Die-to-component

- Component-to-plating bar

Each phase (when necessary) of routing has a different set of requirements and issues.

 For detailed information on routing concepts generic to the Allegro layout editors, see the Allegro User Guide: Routing the Design.

Wire Bond Routing

The requirements for wire bonding vary from design to design and by technology. In some cases, wire bonds begin at a die inside a cavity and finish on a multiple cavity shelves. In other cases, wire bonds begin on a die mounted on the surface layer and end on the surface layer. In both cases, wire bonds can take the form of a staggered, radial, or straight orthogonal pattern. These requirements are primarily driven by the component design that you are using.

Wire Bond Routing Constraints

Once you establish the configuration and pattern, guiding constraints include:

- Bond finger dimensions and bond wire connect location
- Bond finger X or Y location (only for orthogonal)
- Bond finger-to-bond finger spacing
- Min and max wire bond length
- Max wire bond angle (only for radial)

With these constraints, the system can generate an appropriate pattern for the given design, ensuring that all conditions are met. If special wire bond locations or bond finger designs are necessary (for example, the corner wire bonds), then use interactive utilities for fine tuning.

Component I/O Z-direction Routing

Component I/O Z-direction routing is necessary to establish the connectivity between the surface brazing location and an internal layer (routing layer or a power or ground plane). You would typically make these connections using single- or multiple-vias within the perimeter of the brazing pad.

Some applications may require that these vias pass through the entire stackup while others may allow the "best fit" feed-through. The "best fit" via transcends only the minimum necessary layers to satisfy its associated connectivity. For example, a VCC I/O via would begin at the surface layer and

end at the last VCC plane encountered. A signal I/O via would begin at the surface and end at the last signal routing layer encountered.

This "best fit" scenario allows for maximized routing usage throughout the component by eliminating unnecessary punches through layers. This, however, may not be allowed for all packaging technologies and should be confirmed with manufacturing.

For power and ground connections, you may specify multiple vias. To determine the maximum allowable number of vias that can be used, you must know the geometries of both the via and the brazing pad. A special grid may also impact the number of vias that can be added.

Die-to-Component Interconnect

Die-to-Component interconnect involves fanning out all of the die-to-component interconnections and routing to the destination component pin. Depending on the technology being implemented, this routing may take on different forms:

- Routing patterns for Quad Flat Packs (QFP) and Pin Grid Arrays (PGA) are typically a triangular fanout pattern from the wire bond pads coupled with an "any angle" connection to the component pin.
- Routing patterns for Ball Grid Arrays (BGA) are typically an intricate weaving of traces through the flip chip pin locations to an edge pattern of vias coupled with an intricate pattern into BGA ball locations.
- Routing patterns are a result of an optimized usage of routing real estate, resulting in the minimum number of routing layers.

Interactive routing is more suited for intricate routing patterns.

ECAD systems offer a host of interactive routing capabilities that you can use to semiautomatically build fanout patterns and then complete with "any angle" routing.

Through graphical representations of connectivity lines and online design rule checking, routing efficiency is easily realized as manufacturing concerns are minimized.

Die to Die Interconnect

When routing among dies, you must identify critical signals (clock, high speed buses) and route them first. Once you have routed critical signals, you can run signal analysis as described for the [Pre-Route Signal Integrity Analysis](#) phase. You then model the actual routed traces, replacing the manhattan distance estimates. You can use the new interconnect delays to verify that the timing budget has been met. Finally, you can edit the routed traces to reach the desired level of signal integrity for the critical signals.

Once you establish the acceptability of critical net routing, you can route the remainder of the layout, either interactively or automatically. You should take advantage of constraint-driven routing for delay- and distortion-based rules, such as delay and crosstalk. Adherence to these rules during routing sifts out any potential signal integrity issues, thus minimizing rework after performing detailed analysis.

An autorouter can route acceptably with minimal input, however, you must specify manufacturing rules prior to routing. These rules include:

- Spacing (by layer, if applicable)
- Line widths
- Line impedance
- Legal via selection
- Blind and buried via spacing
- Min/Max stagger size

Power and Ground Vias

You must also route *power* and *ground* via connections. Depending on the type of design, you may route them before or after you route the signals. You must consider the via type (blind, buried, through) and the specific plane to which they attach.

Component-to-Plating Bar (Optional)

Component-to-plating bar is the final step of routing. During this process, you route component I/Os through the component periphery to outside connection points (which represent the actual plating bar). Using "any angle" interactive routing techniques is the optimal way to achieve 100% connectivity.

After the component is manufactured, the plating bar is cut off and dangling traces exist for all signals. For critical signals, these dangling traces should be kept to an absolute minimum so signal distortion is not introduced into the component.

Optionally, after you route the entire component, you can rerun thermal analysis to increase the accuracy of the predicted temperatures. You can then use more accurate temperatures to update the earlier [Thermal Analysis](#), and increase the accuracy of buffer models used in the next phase.

Post-route Signal Integrity Analysis

Crosstalk Effects

You spend the majority of the time in this phase dealing with crosstalk. After routing, you should rescan the entire design to verify that you have not introduced excessive ringing, interconnect delay, or SSN. Follow the procedures described in the [Pre-Route Signal Integrity Analysis](#) section. If you performed pre-route signal analysis, you should only be concerned with verification, in which case you would use the full detail of the routed interconnect as opposed to manhattan-based estimates.

False Triggers

When screening for crosstalk problems, it is critical to minimize the number of false alarms, as this can create a tremendous amount of additional work. In doing so, considering the relative switching times of neighboring nets is extremely important. If you do not consider relative timing, you are forced to assume that all neighbor nets switch simultaneously, which can lead to extremely pessimistic and not very useful results.

Resolving Crosstalk

Once you identify a problem net, you can run a simulation using computationally-intensive, multiline algorithms. This accounts for termination and cancellation effects that may exist in the circuit. If excessive crosstalk does exist, you can run additional simulations to isolate the contributions from individual neighbor nets.

You typically solve crosstalk issues by increasing the spacing between the victim net and its offending neighbor nets, or by re-routing the net to minimize the parallelism between it and its neighbors. Online design rule checks can also be very helpful.

Constraints and Requirements

A *constraint* is different than a *requirement*. For example, assume that there is a signal in which the crosstalk constraint for a net is in violation. The crosstalk constraint is not a requirement; it is simply a "target" to help facilitate a requirement. In this case, the actual requirement mandates that the noise margin of the driver/load connections on the net remain positive. If the crosstalk constraint is violated by 25 mV, but there is still 75 mV of noise margin, the net is probably within the requirement. Therefore, no further modifications are necessary.

This selective "constraint relaxation" strategy is typical of trade-offs that you must make in a

performance-driven design methodology.

Allegro X Advanced Package Designer Flows

This appendix presents design flows that illustrate the use of the Allegro Package Designer (APD) tool. The Package-Design Flow is described in Figure 2-3 in Chapter 1 of this user guide.

IC-Driven Flow

The IC-driven flow uses an example to describe the creation of the co-design dies within the context of multi-component packaging, and the management of constraints between the IC under design and the rest of the components. Note that this flow is only one example of how to complete co-design tasks.

To complete the example that shows the IC-driven flow, it is important to get a working configuration of the die I/O layout, bump pattern, component component layout, and component ball assignment. The result is a co-design I/O and bump layout, and a multi-component packaging design that meets the physical and Signal Integrity (SI) constraints, which can then be sent to the IC and component designers for final design.

Using the Virtual System Interconnect Model (VSIC), an up-front SI analysis establishes the I/O driver selection, wire bond or flip-chip connectivity, and component substrate selection (either specific component, or component technology). Once selected, the tool places an instance of the co-design die, and the placement and connectivity are propagated throughout the design.

Figures C-1 and C-2 describe the steps to complete one example of an IC-driven flow. Note that there are many ways to complete the co-design process.

Figure 8.1: IC Driven Flow - SI Analysis

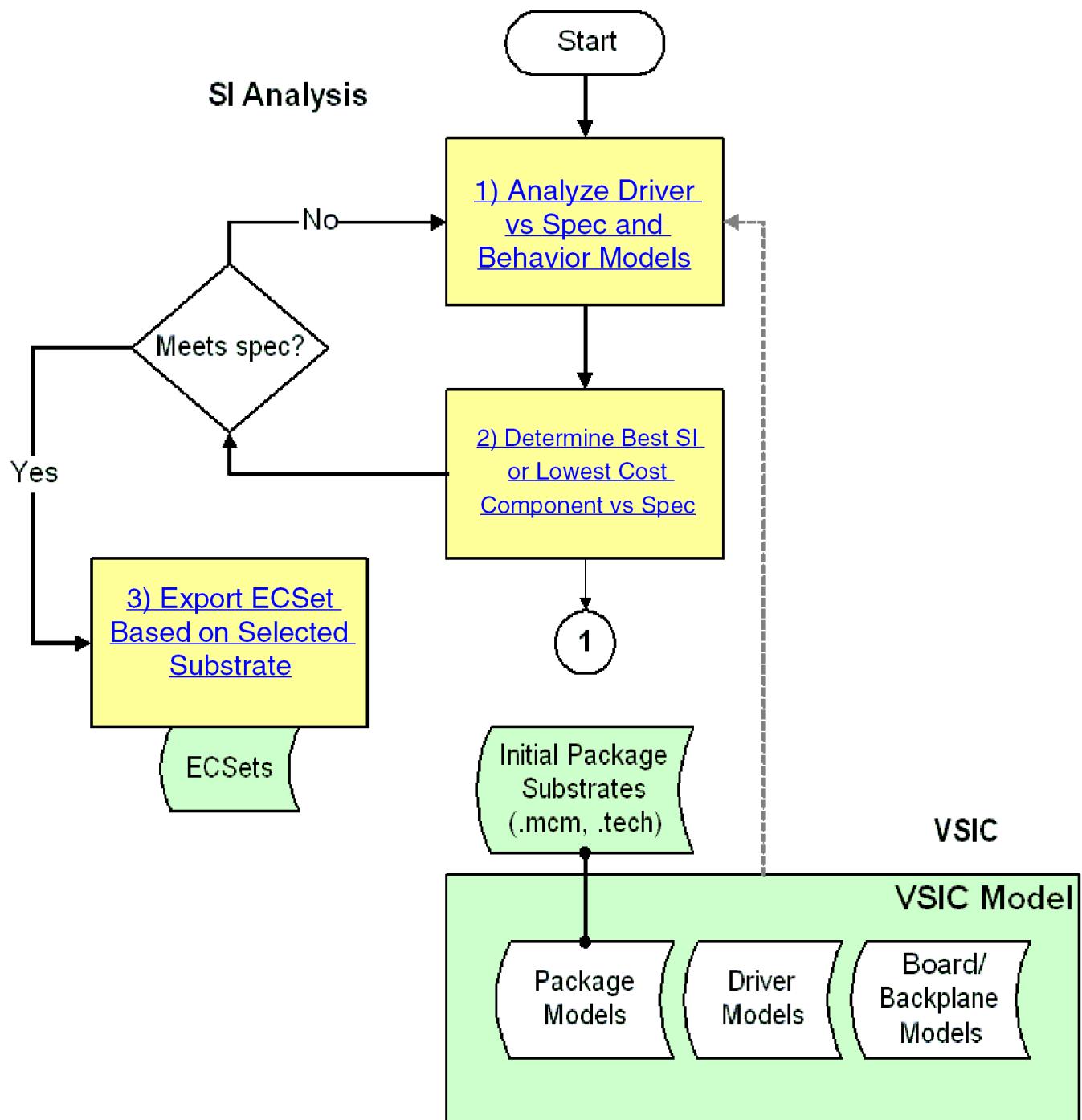
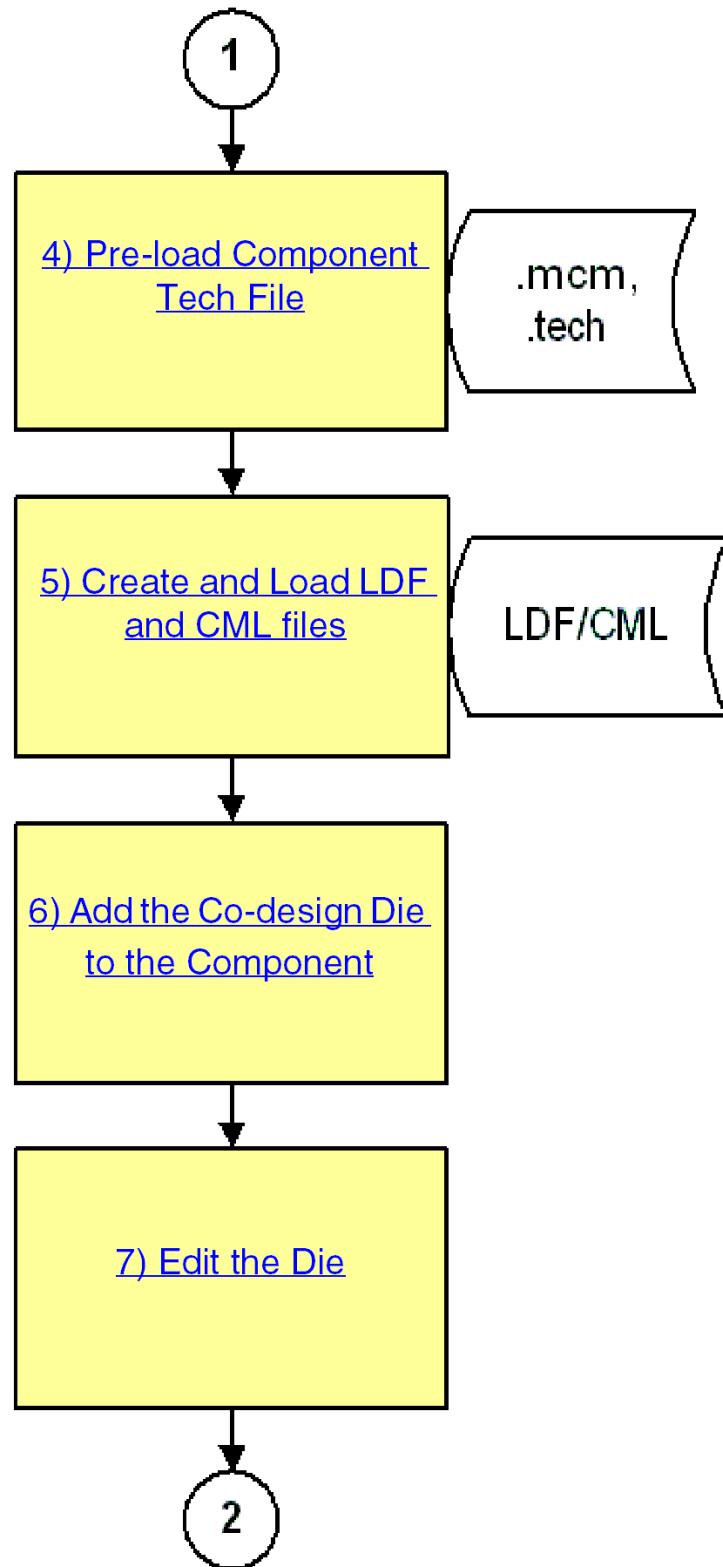
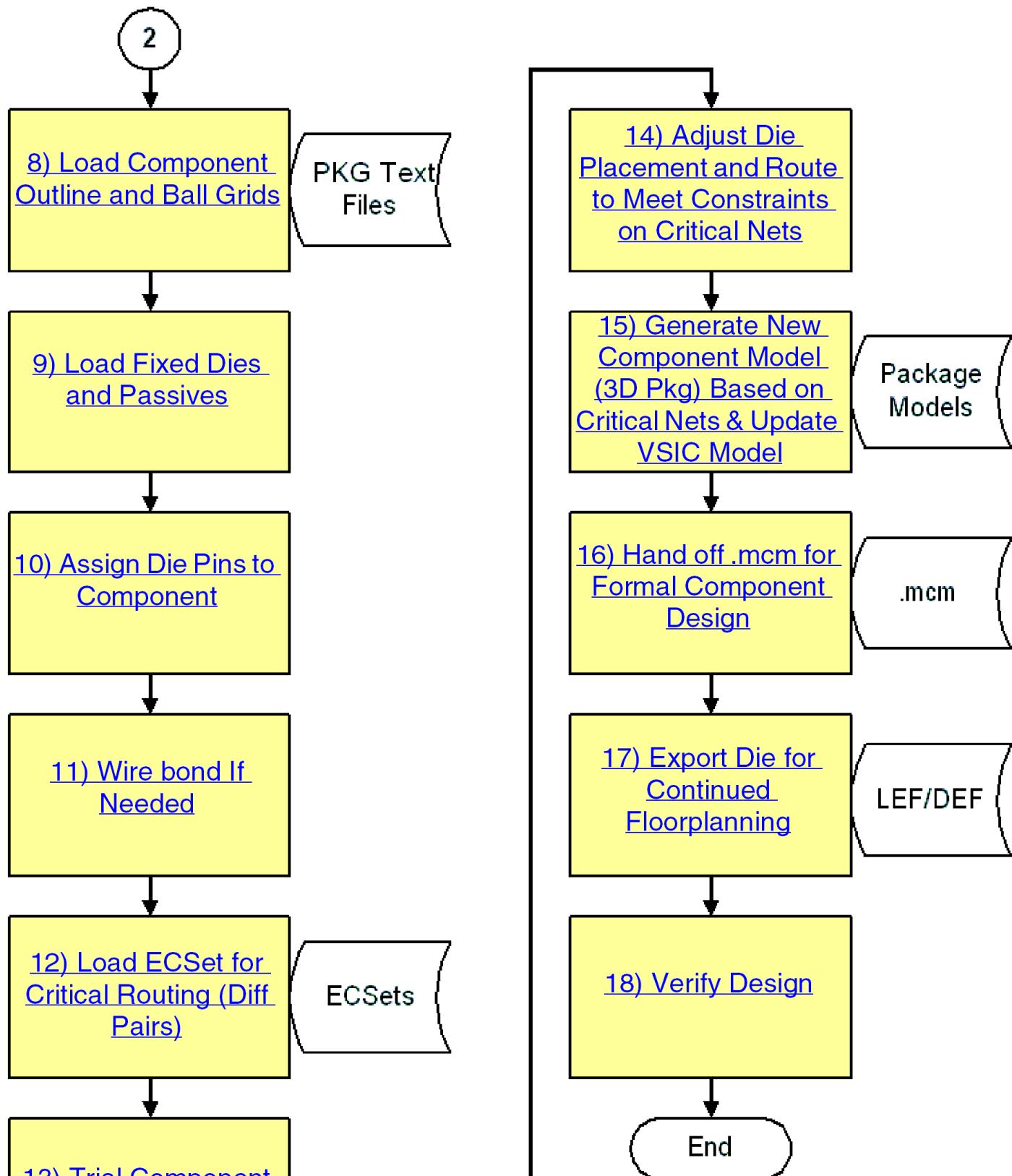


Figure 8.2: Design Start: IC Driven Flow - Physical Design





13) Trial Component Route (Feasibility)

1. Analyze the Virtual System Interconnect (VSIC) model.

Evaluate and compare the specification or actual driver models with behavior models for the die-to-component interconnect, PCB, and backplane using the VSIC model and Allegro PCB SI. Take each file from the library and evaluate it with each other in different combinations.

2. Determine the best combination.

Based on the evaluation of different combinations of the elements listed in Step1, you need to establish the drivers and component substrate type that provide the best compliance to the signal integrity (SI) specification. The SI specification may include timing, matched propagation, waveform integrity, signal strength, cross-talk, simultaneous switching noise (SSN) and so on.

The SI engineer informs the physical/system designer of the substrate selection.

3. Export the electrical constraint set.

Once you establish a best-case combination driver and component substrate, you can export an electrical constraint set (ECset), that you use in the Constraint Manager to manage the routing and placement of elements in the design, to meet the system designer's expectations. You give the ECSet, like the component substrate selection, to the physical designer to use when designing the substrate.

4. Pre-load the component technology file.

Based on the analysis conducted in Steps 1 through 3, establish an initial component technology to use in the overall design.

⚠ If you have a complex digital or mixed-signal IC, and you are co-designing the die pin matrix with the component and IC layout editors, you should add the die as a co-design die, and not as a standard die.

- a. Open a new design (.mcm) file.
- b. Choose *Import – Techfile* to import the stackup, design-level constraint data, and user-defined properties.
- c. Set the drawing size, units, accuracy, text and grid, subclasses, subclass colors and

visibility, and any other pre-design parameters required to design the component (*Setup – Design Parameters, Setup – Subclasses, Display – Color/Visibility*).

- d. Save the database, for example, `codesign_die.mcm`.
5. Before placing an IC design, you must create and load `.ldf` and `.cml` files. This is required even if you use an OA database instead of DEF. The reason for this is that the `.cml` files are the mechanism that the tool uses to keep track of which information in the `.oa` database defines the die-connect shapes in the IC. In fact, it is necessary to define the location of the `.ldf` file each time you start up the tool. Use the following command to set up your LEF files: *Setup – LEF Libraries*.
6. Use the *Add – Co-Design Die* (`add_codesign_die`) command to add the co-design die to the component.
If you select the option to load the design from a DEF file, you must give the DEF file name. If you are loading from an OA file, follow the sequence of steps described below:
From the OA Import command dialog box:
 - a. Select the library definition file to use, normally `lib.defs`, in the current working directory.
 - b. Select the OA library to read the IC layout for the co-design.
 - c. Select the cell from the chosen OA library for the co-design IC.
 - d. Select the view of the chosen cell that contains the IC layout for the co-design die. Note that FE must have previously written the library/cell/view using its `saveOaDesign` command, or the co-design die does not work correctly.
 - e. Specify the reference designator for the co-design die.
 - f. Specify the orientation, location, and rotation for the co-design die.
 - g. Choose *Import* to import the IC data from OA and add an instance of it to the component as a co-design die.
If the import is successful, the die is added to the component according to the placement parameters specified. IOP does not launch because you are not making any changes to the die; you are only adding the existing die to the component from OA.
 - h. If you select an OA design that already exists in the component, an error message appears since currently, the tool does not allow multiple instances of a co-design die in a component.
 - i. Once the `add codesign die` command has worked successfully, save the design using

the *File – Save* command.

7. To edit the die in IOP, use the *Edit – Die* command. Then follow the sequence of steps described below:
 - a. From the first screen of the Die Editor dialog box, select the co-design die for editing. The tool determines which OA library/cell/view contains the IC design for the die. It then launches IOP and instructs IOP to open the IC layout from that OA library/cell/view. Note that FE must have previously written the OA library/cell/view or this operation fails.
 - b. IOP launches, performs a handshake with the layout editor to make sure it was launched correctly and can communicate successfully with APD, then reads the OA library/cell/view using its *Restore OA Design* capability.
 - c. Change the IOP view to *Floorplan view* (from *Physical View*).
 - d. Modify the die.
Typical modifications involve power or signal assignment, bump/bond finger placement, I/O cell placement, and RDL routing.
 - e. When you complete the current set of changes, use the IOP *updatePackage* command. IOP saves the current IC layout to a temporary OA library/cell/view using its *Save OA Design* capability. Then IOP sends a message to APD to instruct it to import the data from OA and update the die instance in the component design.
APD reads the specified temp OA library/cell/view; and replaces the previous version of the die representation with a new one according to the original placement location and orientation.
 - f. To make changes to the IC layout, invoke the *updatePackage* command in IOP several times to investigate the impact of the changes on the component.
 - g. When you are satisfied with the latest set of changes that has been updated from IOP back to the component, save the design using the *File – Save* command in APD.
APD saves the current database (*.mcm*), and then for each co-design die that has unsaved changes stored in temporary OA library/cell/views, it replaces the original OA library/cell/view with the latest temporary version written by IOP.
8. Load the component outline and ball grids.
Depending on the operation used to pre-seed the component design at the beginning of the flow, the component symbol may or may not exist. If not instantiated, you can either interactively create the component symbol or load the spreadsheet containing the component extents and ball pattern using the BGA Text-in Wizard.

- a. Choose *Add – BGA – BGA Generator* in the APD Design Window to invoke the BGA Generator.
 - b. Enter the *Name, Refdes, Origin*, and component dimensions in the BGA Generator - General Information dialog box.
 - c. Based on the dies in your component, click *Next* and enter the appropriate *Number of Pins, Arrangement, and Spacing* in the BGA Generator - Pin Arrangement dialog box.
 - d. Choose *Next* and establish the appropriate power and ground to signal ratios to match the power requirements of the dies on your component in the BGA Generator - Pin Use Ratio dialog box.
 - e. Choose *Next* and select padstacks from existing libraries or create padstacks for both the perimeter and core pads in the BGA Generator - Padstack Information dialog box.
 - f. Choose *Next* and choose an appropriate pin numbering scheme and associated display settings in the BGA Generator - Pin Numbering dialog box.
or instead of performing steps a through f, perform the following step:
Choose *Add – BGA – BGA Text-In Wizard* in the APD Design Window and import the spreadsheet file using the BGA Text-in Wizard.
9. Load fixed dies and passives.
- You can load the rest of the components and then place them in the multi-component packaging design. These may include standard dies or non-die components such as surface mount passives. You can adjust placement using the *Move* and *Spin* commands to offer the best routing solution.
- You can also create and adjust the die stacks due to space restrictions using the layer stack editor.
- The assumption in this flow is that you use a netlist to define the connectivity and that the net names between the components align.
- a. If you are adding a second die, be sure to add the appropriate layers to the cross-section to support the wire bonds and bondpads of the second (stacked) die. Step 15 shows an example.
 - b. To add a second die, choose *Add – Standard Die – DEF In* in the APD Design Window to create a standard die component for the second die in the design. This die will be wire bond (chip-up) and stacked on top of the (chip-down) flip-chip co-design die already present in the component.
 - c. To create a standard die, choose *Add – Standard Die – Die Text-In Wizard in the APD*

Design Window and import a `.txt` file to create a standard die component for the third die in the design.

10. Assign components to each other and to the component balls.

Create or load the connectivity between the die and passive components, as well as the connections to the component balls.

 When you start a new multi-component packaging design, you need a netlist. You can:

- Take the IC netlist information read into the die pins by `def in`, and use a combination of the *Logic* menu net assignment commands to establish interconnect between dies and the BGA balls of the component, based on nets from the DEF files. You may even create some new nets for unassigned pins that do not have appropriate nets from dies that can be assigned to them.
- Create the netlist using the `assign multi nets` and `auto create net` commands to create nets on the BGA balls and possibly some of the dies. Then use the other *Logic* menu commands to establish the interconnect, based on the newly created nets. This new netlist overrides (reassigns) nets that were brought in from DEF. You can purge those nets from DEF using the *Purge Unused Nets* command.
- Use a third-party tool (even some non-EDA tool such as Excel) to establish interconnect. Write that interconnect into a text file and then read the text file into APD. This overrides the nets from DEF, which again you need to purge.
- Use a schematic to create the interconnect.
- Choose *Logic – Auto Assign Nets* and *Logic – Assign Multiple Nets* in the APD Design Window and complete the net assignment between the dies, and then from the dies to the component pins.
- Use the interactive assignment commands (*Logic* menu) to optimize the assignment for routing.

11. Wire bond, if needed.

Use the auto and manual wire bonding tools to generate the wire bond pattern on the dies that were designed for the wire bonding process.

Verify that the interactive wire bond command set can create the appropriate wire bond pattern/configuration for the die that is mounted or stacked on top of the flip-chip mounted co-design die.

- a. Create padstacks for the bondpads.

- b. Make sure that the die pads of the die to be wire bond are on the correct layer to facilitate the wire bonding process.
 - c. Use the new wire bonding tools for the wire bonding process. See the *Allegro Package Designer User Guide: Routing the Design*.
 - d. Save the APD database (.mcm).
12. Load the ECset for critical routing.
Based on the SI analysis and the design content, some portion of the nets may have specific routing constraints. Load the ECsets generated in the SI analysis and apply them to the critical nets. The critical nets may address matched delay, maximum length, SSN, crosstalk, routing over split planes, and so on. Once loaded, the tool flags any existing rats and trials with DRC markers.
 - a. Choose *Setup – Electrical Constraint Spreadsheet* in the APD Design Window.
 - b. Choose *File – Import – Electrical CSets* in the Constraint Manager Window and import the ECset that was created in the VSIC/SI Analysis phase prior to the physical layout.
 - c. Apply the appropriate CSets to the special nets that require electrical rules. This set includes any differential pairs in the design as well as those nets that require delay rules.
 - d. Based on the new ECSet rules, review the DRC markers on the nets in the design that report the electrical violations.
13. Conduct a trial component feasibility route study (done after the differential pairs routing).
Use PCB Router or the manual routing tools to place initial routes based on the pin-to-pin assignment. Use the *Pin Swap*, *Move*, *Spin*, and component editing commands (where permissible) to resolve conflicts or congestion. This facilitates a more accurate 3D component model.
 - a. Using the manual route and interactive etch edit commands, route the differential pairs and other critical routes using the heads-up display that indicates the compliance to the electrical constraints.
 - b. Run some routing passes with the PCB Router and evaluate the results. If the route can be completed in compliance with the design rules, finish the routing using the manual route and interactive etch edit commands.
14. Adjust the die placement and route to meet constraints.
You may have to adjust the placement of components on the substrate to meet certain constraints. Likewise, you may have to adjust the trial routes to meet these constraints.

- a. Choose *Edit – Move* in the APD Design Window to adjust the placement of both dies and passives to reduce or eliminate the electrical DRC markers, which indicate violations to the ECSets applied to the critical nets.
 - b. Use the assignment tools (*Logic* menu) to adjust the component pin assignment and reduce or eliminate remaining electrical DRCs.
 - c. Save the APD database (`.mcm`).
15. Generate a new component model.
- Once you place tentative routes, especially for the critical nets, you can generate a new net-based or full-component, 3D model and update the VSIC model to include a more accurate representation of the component model. From the physical tool, you can generate a full 3D component model using the *Analyze – Si/EMI Sim – 3D-Package Model* command. In the SI version of the tool, you can generate a net-based model using the Optimal Pak-si tool with the *Analyze – Si/EMI Sim – 3D-Modeling* command.
- a. Choose *File – Change Editors* and choose APSI
 - b. Generate net-based models for all of the critical signals of interest.
 - c. Add these new models to the Device Model Library (DML).
 - d. Run a 2D SI analysis individually on the nets critical nets.
 - e. Evaluate at these models with SigXP.
 - f. Based on the electrical analysis of these models, make appropriate adjustments to the routing where necessary.
 - g. Once you have resolved net level issues, generate a full 3D component model using the Optimal Pak-si tool.
 - h. Run 3D analyses using the full component model.
 - i. Make final adjustments to the design to correct for any design violations revealed by the net and design analysis.
16. Hand off the `.mcm` database for formal component design.

Once you are confident that the overall initial design works from layout, organization, placement, routing, and SI perspectives, save the `.mcm` database. Also save a backup copy. Then return the database to the component designer to finalize the design based on manufacturing constraints, add design finishing edits, and generate the manufacturing outputs.

17. Export die for floorplanning core.
18. Export DEF so that the IC floorplanning engineering can finish the overall floorplanning of the die and continue the die design process.

Component Design Task Flows

This section contains use models for performing some common flows in APD. The flows use multiple functions as well as references to Cadence's digital IC layout editor, First Encounter (FE). You can find details related to specific APD commands in the *Allegro PCB and Package Physical Layout Command Reference*. Links to the document appear in blue, underlined text; you can find details related to First Encounter in that product's user documentation.

The flows described in this chapter include:

- [Performing Component Route Feasibility Based on Die Pin Matrix from IC Layout](#)
- [Establishing Component Route Feasibility in a Standard Component, both Manually and Automatically](#)
- [Creating a Set of Split Rings Around a Complex Wire Bond Die](#)

Performing Component Route Feasibility Based on Die Pin Matrix from IC Layout

Goal	Perform a pre-route analysis of a component, based on a die pin matrix brought in from FE or a similar IC floorplanning tool.
Conditions	A preliminary layout has been created in FE or a similar IC tool, with an initial pin pattern created and assigned to the nets. A custom component is being created for the die.

1. In APD, open a new database:
 - a. Run *File – New* ([new](#) command).
Use the *Package/multi-chip* drawing type (not the wizard).
 - b. When the New Drawing Configuration dialog box appears, select a component configuration appropriate for the die that you are importing.

- You choose a component on the basis of the die's complexity, density, and pin pattern.
- c. Using these criteria as factors, decide which packaging technology is most appropriate; for example, BGA full or perimeter matrix, lead frame, and so on.
2. Having determined the appropriate component technology, set up the layer structure for the component, either by importing a tech file with *File – Import – Techfile* ([techfile in](#) command) or if a tech file is not available, run *Setup – Cross-section* ([xsection](#) command) to set up the cross-section.
3. To set constraints manually, run *Setup – Constraints and choose an item from the submenu*.
4. Import the information into the new database by performing the following steps:
-  You may want to add the die as a co-design die.
- a. In APD L, choose *File – Import – DEF* ([def in](#) command) to display the **IC Import from DEF** dialog box. In APD XL, choose the *Add – Standard Die – DEF* command to display the DEF Import dialog box.
 - b. Verify that the correct library definition file (.ldef) and library are selected. If they are not, bring up the LEF Library Manager from the Import DEF dialog box to navigate to the appropriate .ldef and library.
 - c. Choose a DEF file, placement information, and the die type. The type of die you use should match the type you specified earlier in the New Drawing Configuration dialog box.
 - d. Choose *Import* to begin the process of bringing the data into APD.
If you are importing a complex design, the process may take several minutes.
5. When setup is complete, create a basic component in APD L using *Generate – BGA Generator* ([bga generator](#) command) to establish an initial ball field. In APD XL, choose *Add – BGA – BGA Generator*.
6. Use *Edit – BGA* ([bga editor](#) command) to make changes to the initial component. You typically need to change power/ground assignments or add or delete balls. In some cases, you may have to change some padstacks such as the corner pin in one corner to indicate visually the orientation of the component.
7. When you are satisfied that you have a viable component for the die, map the die pins to the corresponding component balls using *Logic – Auto Assign Net* ([auto assign net](#) command).

⚠ Select the route feasibility option in the **Automatic Net Assignment** dialog box to view details on problem nets that you need to address. Otherwise run *Route – Route Feasibility* ([route feasibility](#) command) as a separate command later.

8. Choose *File – Save* to save your design.

The following table lists some problems you may encounter and how you can address them:

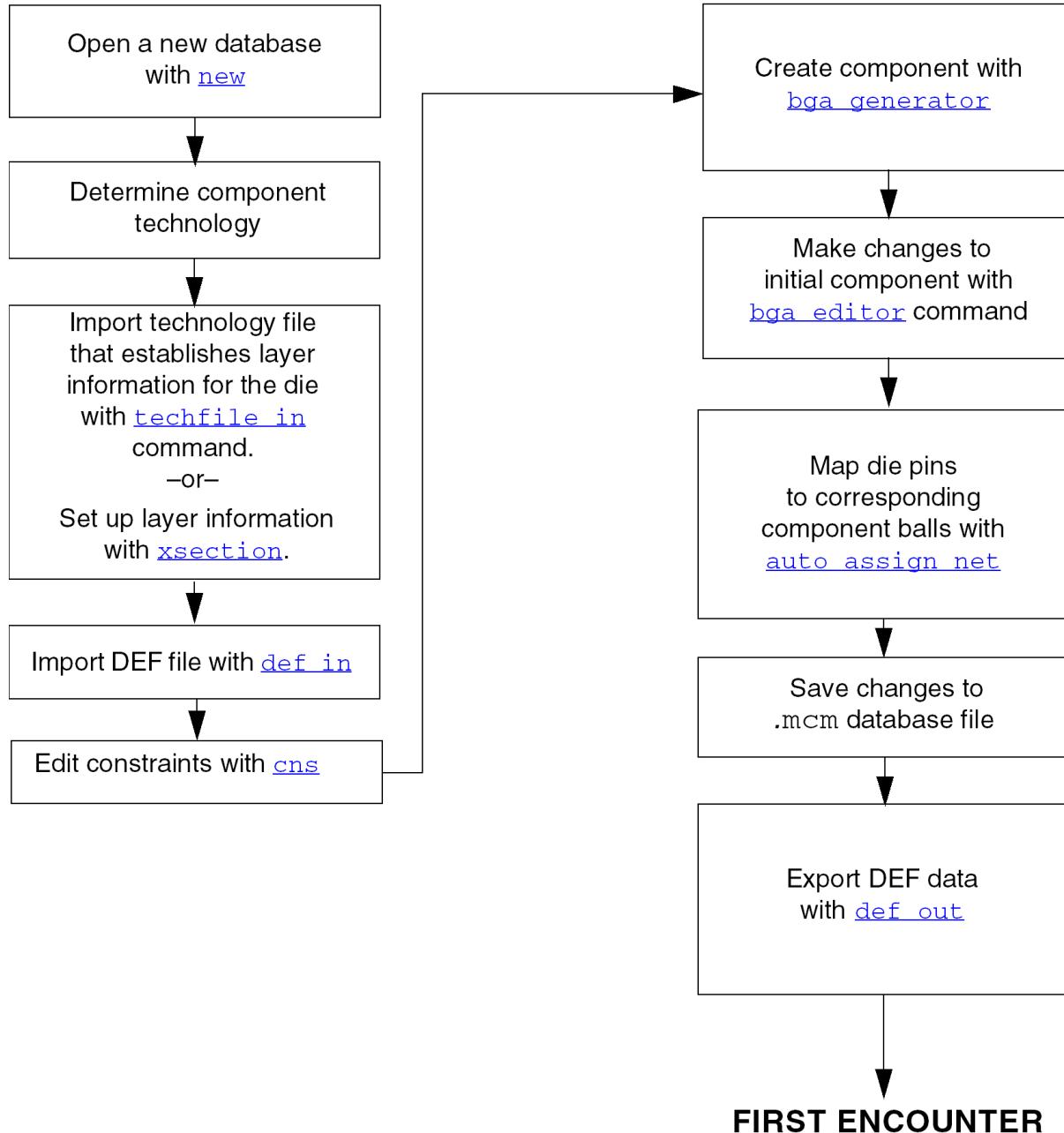
Problem	Address using...
Bad ball mapping	<ul style="list-style-type: none">• <i>Edit – BGA</i> (bga editor command)• <i>Logic – Assign Net</i> (assign net command)
Questionable layer mapping	<i>Route – Routing Layer Assign</i> (assign route layer command)
Die pin layout	<ul style="list-style-type: none">• <i>Logic – Assign Net</i> (assign net command) if neighboring pins' ratsnests lines cross and need to swap.<ul style="list-style-type: none">Otherwise...• <i>Edit – Die</i> (die editor command) to:<ol style="list-style-type: none">1. Swap the two affected die pin locations.2. Regenerate the die symbol and component information.• <i>Route – Connect</i> (add connect command) to change routing manually.

1. When you complete troubleshooting all routing problems, save your database and export changes you made to the die data back to the IC designer using *File – Export – DEF* ([def out](#) command). If you are performing this step, you should have added the die as a co-design die (Step 4).

⚠ Your DEF file accurately translates only information on physical pins, cover bump cells, or drivers. Because APD cannot communicate I/O cell placement information, data on die pins built into an I/O cell may not be understood by the IC tool.

Once you establish component route feasibility and finalize the die pin matrix, the layout of the IC and component can proceed.

Component Route Feasibility Based on Die Pin Matrix Flow



Establishing Component Route Feasibility in a Standard Component, both Manually and Automatically

Goal	Perform a manual pre-route analysis of a die in development inside a standard component, then compare this method with an automated process using Cadence's IC tool, FE. Keep in mind that this process lets you determine the least expensive standard component to satisfy your design requirements.
Conditions	A library of pre-defined standard packages, including: <ul style="list-style-type: none">• A defined component layer stackup• Defined design rule constraints for the component layer• Existing component component of class IO The component design may optionally include: <ul style="list-style-type: none">• A power and ground net assignment for the component balls• Component pin escape routing in the form of offset vias or via structures designs• Routing layer assignments for component pins• Component routing and wire bond bond finger tiers

Manual Flow with APD

1. Based on the signal count, pin density, and other factors governing the makeup of your die, choose the most appropriate standard component from your library. Use *File – Open* ([open](#) command) to display the component in APD.

2. Import the die information into the new database by performing the following steps:

 If you have a complex digital or mixed-signal IC, and you need to co-design the die pin matrix with the component and IC layout editors, you should add the die in APD as a co-design die, and not as a standard die.

- a. Choose *File – Import – DEF* ([def in](#) command) to display the **IC Import from DEF** dialog box. In APD XL, choose the *Add – Standard Die – DEF* command to display the DEF Import dialog box.
 - b. Verify that the correct library definition file (.ldf) and library are selected. If they are not, bring up the LEF Library Manager from the dialog box to navigate to the appropriate .ldf and library.
 - c. Choose a DEF file, placement information, and the die type. The type of die you use should match the type specified in the standard component (.mcm) that you opened.
 - d. Choose *Import* to bring the die data into APD. If you are importing a complex design, the process may take several minutes.
 - e. Verify and save the design to a new file name with *File – Save As* ([save as](#) command). This safeguards you from overwriting your standard component database.
 - f. Map the die pins to the corresponding component balls. There are different options you can choose to best accomplish this task, based on the constraints and definition of your component:
 - g. For designs composed of a standard single die in a BGA component, use *Logic – Auto Assign Net* ([auto assign net](#) command) with the route feasibility option selected to view details on problem nets. The balance of this flow follows this path.
 - h. For other component types, use *Logic – Auto Assign Net* ([auto assign net](#) command) with the route feasibility option unchecked.
 - i. Note that this option does not invoke the router automatically.
 - j. For wire bond designs in which the component defines all routing between the bondpads to the component balls, use *Route – Wire Bond – Select* ([wirebond select](#) command, right-click and choose *Reconnect* from the menu).
- If you are unfamiliar with the functionality of the ICP wire bonding process, see Chapter 8, "Wire Bonding Toolset." in the *Allegro Package Designer User Guide: Routing the*

Design.

3. If the feedback derived from your use of the route feasibility option in *Logic – Auto Assign Net* indicates an insufficient number of balls for the signal I/Os or a probability of an insufficient quantity of routing channels or layers, return to step 1 and select a more appropriate standard component from your library.
4. Once you determine the best component for use with the die information imported from the DEF file and the results obtained from *Logic – Auto Assign Net*, manually correct any nets highlighted by the route feasibility option.
5. If the component is standard, therefore unchangeable, you have to make changes to the die component (instead of the component) or to the routing layer assignments. If you are performing this step, you should have added the die as a co-design die (Step 2).



In this case, use:

- *Edit – Die* ([die editor](#) command) to make changes to the die component.
 - *Route – Layer Assign* ([assign route layer](#) command) to make changes to the routing layer assignments.
6. Invoke route feasibility again to verify your fixes.
 7. When you complete troubleshooting all routing problems, save your database and export changes you made to the die data back to the IC designer using *File – Export – DEF* ([def out](#) command).

The IC and component designers can now load the changes into their respective tool environments and the layout of the IC and component can proceed.

Automatic Flow with First Encounter (FE)

This flow is performed from Cadence's IC layout editor, First Encounter (for flip-chip dies only) and System on a Chip (SOC) Encounter.

1. With the die pin matrix defined and assigned in FE, run the `chipio route feasibility` command. (See the appropriate First Encounter user documentation for details on this and other FE commands.)
2. Set the following fields in the route feasibility dialog box:

APDFFile	Specifies the name of the APD <code>.mcm</code> database containing the standard component.
----------	---------------------------------------------------------------------------------------------

<i>LDF File</i>	Specifies the file containing a set of one or more LDF libraries. Each library contains a set of LEF files with data to be used when working with a specific DEF placement file.
<i>LDF Libs</i>	Specifies the pull-down list of all the libraries inside the selected .ldf file. Select the library that corresponds with the current DEF file, allowing APD to pick up the correct LEF information when importing the die.
<i>Die Type</i>	Specifies the attachment type and orientation to be used by APD when the die data is imported.

3. Choose *OK*.

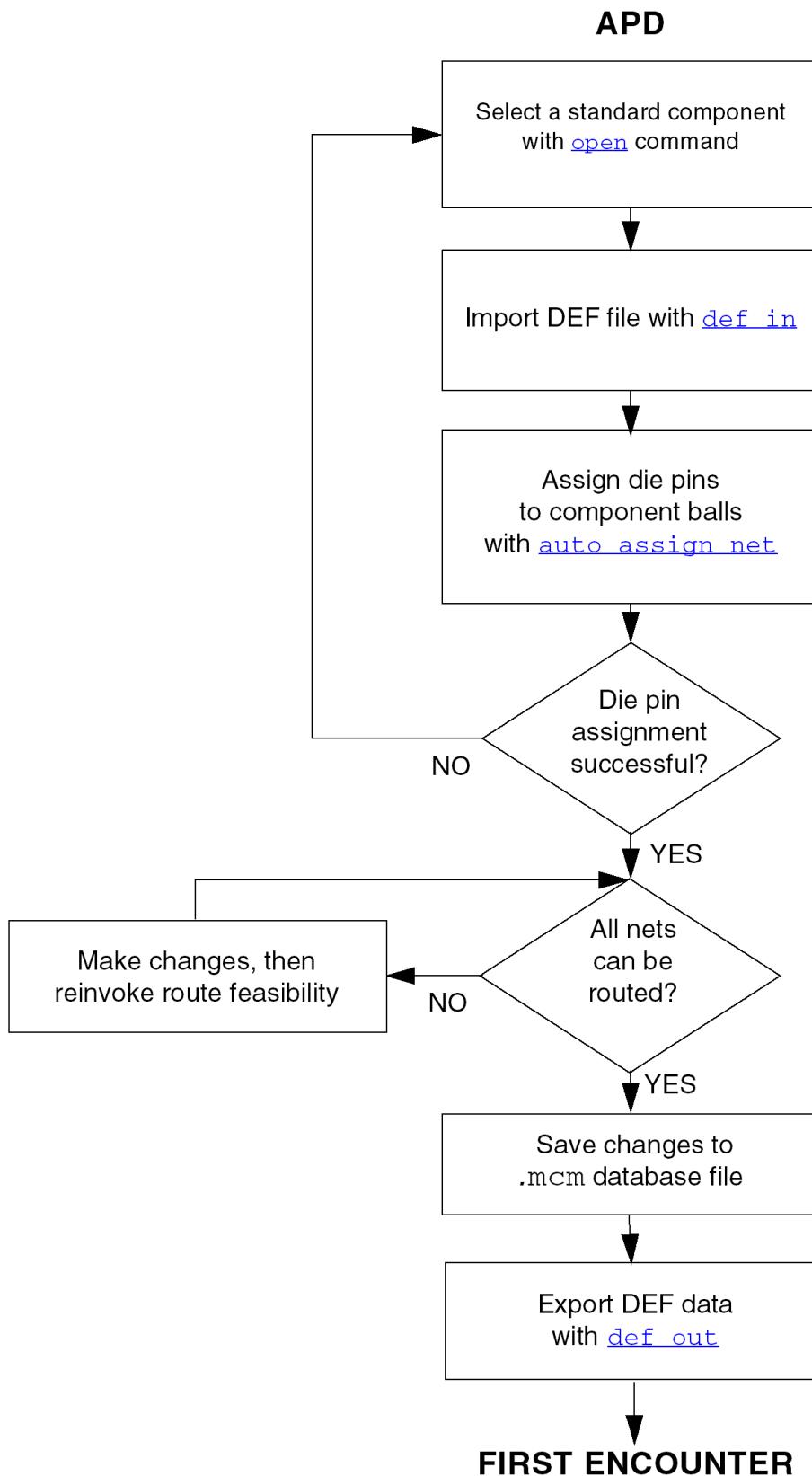
FE generates a DEF file defining the IC and automatically launches APD in the background, initiating the following actions:

- a. Checks that all files are present.
- b. Imports the IC data into the component using a batch version of the `def in` command.
- c. Checks that all required data is present and viable; specifically a class IC die pin and class IO component exist, corresponding number of component signal balls to die signal pins are acceptable, and net assignments to the die pins are present.

Component Route Feasibility in a Standard Component: Manual Method Flow

Getting Started with Physical Design

Allegro X Advanced Package Designer Flows--Establishing Component Route Feasibility in a Standard Component, both Manually and Automatically

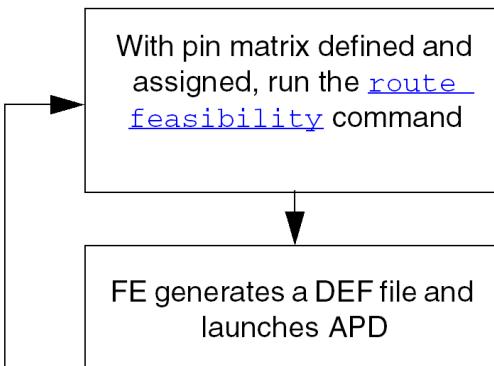


Component Route Feasibility in a Standard Component: Automatic Method Flow

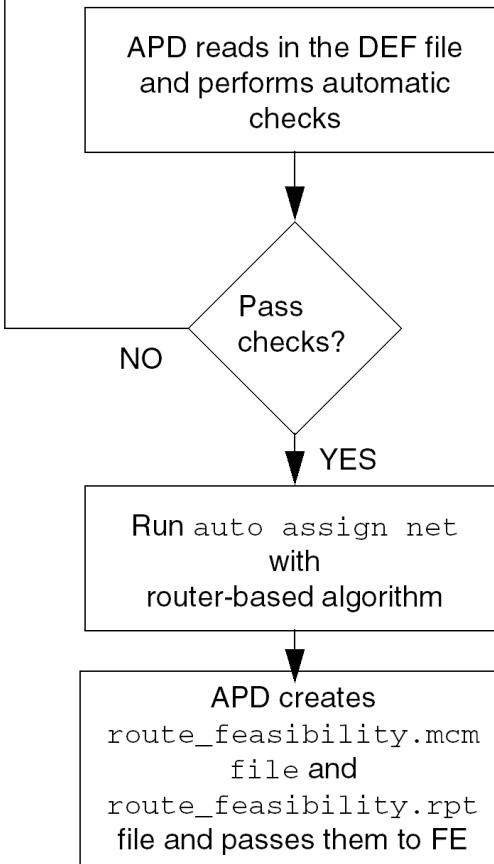
Getting Started with Physical Design

Allegro X Advanced Package Designer Flows--Establishing Component Route Feasibility in a Standard Component, both Manually and Automatically

FIRST ENCOUNTER



APD



FIRST ENCOUNTER

FE highlights any failed nets

Creating a Set of Split Rings Around a Complex Wire Bond Die

Goal	Using the Power and Ground Ring Wizard and dynamic shape functionality, create a set of split rings for use around a complex wire bond die in a component.
Conditions	You have an existing component design containing the die around which you want to create the split ring pattern. This flow assumes that you already know where the major ring breaks (where the rings change nets) will be.

1. Open the design with *File – Open* ([open](#) command).
2. Create a set of whole rings that you will cut up to form your split ring pattern:
 - a. Run *Generate – Power/Ground Ring Generator* ([ring wizard](#) command).
 - b. In the wizard, specify the number of whole rings you want to create.
 - c. Select the reference designator for the die around which you want the rings.
 - d. For each ring, specify the width, corner type, and radius (or length, if you select a chamfer-corner type). Specify also the gap between each ring (or, for the first ring, from the edge of the die).Summary information on the **Result Verification** page of the wizard informs you if DRCs were created as a result of ring creation.
 - e. When you are satisfied with the results of the ring generation, click *Finish* to complete the wizard process and instantiate the rings into the design.
3. Correct any DRCs that occurred following creation of the rings.
4. Ensure that the shape suppression value is less than the minimum ring piece size you will create:
 - a. Run *Shape – Global Dynamic Params* ([shape global param](#) command).
 - b. In the **Global Shape Parameters** dialog box, bring forward the *Void* controls tab and set the *Suppress shapes less than* field to the required minimum value. If you are unsure what this value should be, enter 0 for now. This prevents any shapes in your design being suppressed.
Record the original suppression value so that you can reset it toward the end of this process.

c. Complete the command.

5. Use the line drawing commands of your choice (*Add – Line*, *Add – Circle*, *Add – Arc*, and so on) to create lines on the same layer as the ring shapes. It is recommended that you set your shape-to-line DRC spacing to half the desired width of the gap between ring sections, and set your line width to 0. This ensures the exact gap you require. Record the original shape-to-line DRC value so that you can reset it toward the end of this process.

 The lines must cut through the shapes completely in order to split the ring section into two distinct pieces.

6. Convert each ring into a static shape:

- a. Run *Shape – Change Shape Type* ([shape change type](#) command).
- b. In the *Options* tab of the Control Panel, set the *Shape Fill Type* setting to *To static solid*.
- c. Select the ring segment that you want to convert.
A warning message informs you that you will lose the original shape boundary, parameter settings, and defined voids on the **BOUNDARY** class.
- d. Choose *Yes* to acknowledge the warning.
- e. Repeat steps **a** through **d** for each ring you want to convert.
- f. Choose *right* and choose *Done* to complete the command and return to the idle state.

7. Delete the marker lines that you used to split the rings.

8. Reset the shape suppression and DRC values that you changed in steps 4 and 5 to their original settings.

9. Assign each ring segment to a net of your choice with *Logic – Assign Net* ([assign net](#) command).

10. Reconvert each ring segment into a dynamic shape:

- a. Run *Shape – Change Shape Type* ([shape change type](#) command).
- b. In the *Options* tab of the Control Panel, set the *Shape Fill Type* setting to *To dynamic copper*.
- c. Select the ring that you want to convert.
- d. Repeat steps **a** through **c** for each ring that you want to convert.
- e. Choose *right* and choose *Done* to complete the command and return to the idle state.

Getting Started with Physical Design

Allegro X Advanced Package Designer Flows--Creating a Set of Split Rings Around a Complex Wire Bond Die

The system can now dynamically void the ring segments to add or remove clearances around vias, routing, and other design elements.

Classes and Subclasses in Layout Editors

A PCB design file consists of several drawing layers such as routing layers, plane layers, soldermask layers, and so on. Each layer has its own color and visibility settings. The objects present on each of these layers can be selectively colored and their visibility can also be selectively turned on or off.

To control the color and visibility of the layers, Allegro combines these drawing layers and stores them into a two-level hierarchy in the design database. The first level of grouping is called class. Under each class there is a second level of grouping called subclass. All graphical items in a database are associated with a class and a subclass.

For example, Board Geometry is a class and Outline is a subclass that is used to create an outline for a board. Similarly, Etch is a class, and Top and Bottom are subclasses of Etch that are used to create routing on a board.

Some subclasses are distinct but have the same name. They are distinct because they belong to different classes. For example, the Silkscreen subclass that belongs to Board Geometry includes lines and text that belong to the board; the Silkscreen subclass that belongs to Refdes includes text only, belonging to the components only; and the other Silkscreen subclass belongs to Manufacturing and includes grouping of all types of the other two subclasses. By having separate subclasses of silkscreen, you can control the color and visibility of each type of silkscreen object.

The following table lists some of the commonly used Classes and Subclasses.

Table 9.1: Table for Class/Subclass Definitions

CLASS	SUBCLASS	USED FOR
BOARD GEOMETRY	Assembly_Detail	Creating Assembly Drawing of PCB. For example, milling details, special hardware requirements, and so on.
	Assembly_Notes	Creating Assembly Drawing of PCB, includes general notes for assembly manufacturer.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Both_Rooms	Adding room boundaries as closed polygon, to define a region for a group of components on both top and bottom layers. Works in conjunction with component room properties. "Package to Room" check can be used to verify components place in intended areas.
	Bottom_Room	Adding room boundaries as closed polygon, to define a region for a group of components on the bottom layer. Works in conjunction with component room properties. "Package to Room" check can be used to verify components place in intended areas.
	Dimension	Adding dimensioning parameters to board.
	Off_Grid_Area	Defining off grid region.
	Outline	Creating the physical shape and dimension of physical board. By default, this is a rectangle of the size of working area.
	Place_Grid_Bottom	Defining user-defined grids on the bottom layer which are used for Rename Refdes setup UI, and with auto-placements routines.
	Place_Grid_Top	Defining user-defined grids on the top layer which are used for Rename Refdes setup UI, and with auto-placements routines.
	Plating_Bar	Creating plating features. For example, edge plating, edge fingers, and so on.
	Silkscreen_Bottom	Creating text and line to the board on the bottom layer.
	Silkscreen_Top	Creating text and line to the board on the top layer.
	Soldermask_Bottom	Creating geometries on the Bottom layer to eliminate solder mask in designated areas. Example application is the need to remove soldermask from internal slots and cutouts.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Soldermask_Top	Creating geometries on the Bottom layer to eliminate solder mask in designated areas. Example application is the need to remove soldermask from internal slots and cutouts.
	Switch_Area_Bottom	Defining an area on the bottom side in which all etch is routed in the direction perpendicular to the preferred direction of most of the etch on that etch/conductor subclass. ⚠ This subclass is no longer in use.
	Switch_Area_Top	Defining an area on the top side in which all etch is routed in the direction perpendicular to the preferred direction of most of the traces on that etch/conductor subclass. ⚠ This subclass is no longer in use.
	Tooling_Corners	Creating Board Layer film orientation for fabrication processing.
	Top_Room	Adding room boundaries as closed polygon, to define a region for a group of components on the top layer. Works in conjunction with component room properties. "Package to Room" check can be used to verify components place in intended areas.
	Wb_Guide_Line	Defining the wire bound boundary for bond pads in APD+.
COMPONENT VALUE	Assembly_Bottom	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist manual assembly and rework.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Assembly_EMBEDDED	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist manual assembly and rework.
	Assembly_TOP	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist manual assembly and rework.
	Display_BOTTOM	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist in custom or non-standard processing.
	Display_EMBEDDED	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist in custom or non-standard processing.
	Display_TOP	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce drawings to assist in custom or non-standard processing.
	Silkscreen_BOTTOM	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce PCB silkscreen labels indicating component values.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Silkscreen_Top	Layer of text which defines the component value. The actual text string is typically driven from schematic input data and not added manually in the PCB editor. This layer is used to produce PCB silkscreen labels indicating component values.
DEVICE TYPE	Assembly_Bottom	Displaying device type on bottom layer of assembly drawing.
	Assembly_EMBEDDED	Displaying device type of embedded component in assembly drawing.
	Assembly_Top	Displaying device type on bottom layer of assembly drawing.
	Display_Bottom	Additional subclass. Used for displaying device type on bottom layer.
	Display_EMBEDDED	Additional subclass. Used for displaying device type of embedded components.
	Display_Top	Additional subclass. Used for displaying device type on top layer.
	Silkscreen_Bottom	Creating a text for device type on the bottom layer.
	Silkscreen_Top	Creating a text for device type on the top layer.
DRAWING FORMAT	Drawing-Origin	Creating co-ordinates (00,00) of drawing origin.
	Outline	Creating documentation of design sheet.
	Revision_Block	Adding details about differences between versions of a design.
	Revision_Data	Adding revision history of design.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Title_Block	Lines and text defining a fixed standard format typically unchanged by the user from one design to the next, such as: Company name, address, sheet size.
	Title_Data	Adding design information, such as: Title , Author, Drawing Number, etc.
MANUFACTURING	Autosilk_Bottom	Destination layer for Automatic Bottom silkscreen results. Manual edits to this layer are not recommended.
	Autosilk_Top	Destination layer for Automatic Top silkscreen results. Manual edits to this layer are not recommended.
	Ncdrill_Figure	Previous releases utilized this layer for drill figures. ⚠ This subclass is no longer used and is purged when nc_drill legends are recreated.
	Ncdrill_Legend	Quantifying the number, type, and tolerance of plated and non-plated holes.
	No_Gloss_All	Creating a polygon shape to prevent automatic glossing on all etch layers.
	No_Gloss_Bottom	Creating a polygon shape to prevent automatic glossing on bottom layer.
	No_Gloss_Internal	Creating a polygon shape to prevent automatic glossing on internal etch layers.
	No_Gloss_Top	Creating a polygon shape to prevent automatic glossing on top layer.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	No_Probe_Bottom	Creating a probe keepout area on the bottom layer into the library package definitions in symbol mode to control the proximity of test vias to surrounding tall parts.
	No_Probe_Top	Creating a probe keepout area on the top layer into the library package definitions in symbol mode to control the proximity of test vias to surrounding tall parts.
	Photoplot_Outline	Creating a rectangle to define the area in the design for photoplotting (Artwork generation).
	Probe_Bottom	Creating an automatic test point on bottom layer.
	Probe_Top	Creating an automatic test point on top layer.
	Xsection_Chart	Creating layer stack-up chart for fabrication.
ANALYSIS	High_Isocontour	<div style="border: 1px solid #f0e68c; padding: 5px;">⚠️ This subclass is no longer in use.</div>
	Low_Isocontour	<div style="border: 1px solid #f0e68c; padding: 5px;">⚠️ This subclass is no longer in use.</div>
	Medium1_Isocontour	<div style="border: 1px solid #f0e68c; padding: 5px;">⚠️ This subclass is no longer in use.</div>
	Medium2_Isocontour	<div style="border: 1px solid #f0e68c; padding: 5px;">⚠️ This subclass is no longer in use.</div>

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Medium3_Isocontour	<p>⚠️ This subclass is no longer in use.</p>
	Pcb_Temperature	<p>⚠️ This subclass is no longer in use.</p>
PACKAGE GEOMETRY	Assembly_Bottom	Depicting boundary of package on assembly drawing.
	Assembly_Top	Depicting boundary of package on assembly drawing.
	Body_Center	Creating a text point that defines the component center.
	Dfa_Bound_Bottom	Creating boundary on the bottom layer for Real Time Design for Assembly (DFA) to check clearances between components driven by a Spreadsheet based matrix of components. This boundary can be different from Place_bound_Bottom boundary. You can define this boundary only at the symbol level (.dra). If not defined, then the DFA checks use the Place_Bound_Bottom boundary.
	Dfa_Bound_Top	Creating boundary on Top layer for Real Time Design for Assembly (DFA) to check clearances between components driven by a Spreadsheet based matrix of components. This boundary can be different from Place_bound_Top boundary. You can define this boundary only at the symbol level (.dra). If not defined, then the DFA checks use the Place_Bound_Top boundary.
	Display_Bottom	Additional subclass. Used for displaying package assembly of bottom layer.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Display_Top	Additional subclass. Used for displaying package assembly of top layer.
	Modules	Creating auto-generated regions around the perimeter of a placed Design Reuse module.
	Pad_Stack_Name	Additional layer. Used for displaying Pad stack name.
	Pastemask_Bottom	Creating stencil for assembly. Only needed for surface mounted devices.
	Pastemask_Top	Creating stencil for assembly. Only needed for surface mounted devices.
	Pin_Number	Displaying component pin numbers.
	Place_Bound_Bottom	Creating filled rectangle on the bottom layer that define package-part boundary and govern placement restrictions. These rectangles are used as keepin and keepout areas while placement. They are also used by DRCs to check for violations of package-to-package overlapping.
	Place_Bound_Top	Creating filled rectangle on the top layer that define component areas which may or may not include pins of surface mount devices. Used by DRCs to check for violations of package-to-package overlapping. You can define this boundary only at the symbol level. If not defined it, then it will be automatically created based on assembly_top outline.
	Silkscreen_Bottom	Creating arcs, shapes, text and lines to the bottom layer of the package geometry of the footprint.
	Silkscreen_Top	Creating arcs, shapes, text and lines to the top layer of the package geometry of the footprint.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Soldermask_Bottom	Special solder mask opening requirements of bottom components.
	Soldermask_Top	Special solder mask opening requirements of top components.
REFDES	Assembly_Bottom	Adding component identifier name in Assembly Bottom.
	Assembly_EMBEDDED	Adding component identifier name in Embedded Layer.
	Assembly_Top	Adding component identifier name in Assembly Bottom.
	Display_Bottom	Additional subclass. Used for displaying refdes values on bottom layer.
	Display_EMBEDDED	Additional subclass. Used for displaying refdes values of embedded components.
	Display_Top	Additional subclass. Used for displaying refdes values on top layer.
	Silkscreen_Bottom	Creating text for reference designator on the bottom layer.
	Silkscreen_Top	Creating text for reference designator on the top layer.
TOLERANCE	Assembly_Bottom	Displaying tolerance values of bottom components.
	Assembly_EMBEDDED	Displaying tolerance values of embedded components.
	Assembly_Top	Displaying tolerance values of top components.
	Display_Bottom	Additional subclass. Used for displaying tolerance values of bottom components.
	Display_EMBEDDED	Additional subclass. Used for displaying tolerance values of embedded components.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Display_Top	Additional subclass. Used for displaying tolerance values of top components.
	Silkscreen_Bottom	Creating text for the tolerance value on the bottom layer.
	Silkscreen_Top	Creating text for the tolerance value on the top layer.
USER PART NUMBER	Assembly_Bottom	Displaying user part number values of bottom components.
	Assembly_EMBEDDED	Displaying user part number values of embedded components.
	Assembly_Top	Displaying user part number values of top components.
	Display_Bottom	Additional subclass. Used for displaying user part number values of bottom components.
	Display_EMBEDDED	Additional subclass. Used for displaying user part number values of embedded components.
	Display_Top	Additional subclass. Used for displaying user part number values of top components.
	Silkscreen_Bottom	Creating text for the part number on the bottom layer.
	Silkscreen_Top	Creating text for the part number on the top layer.
PACKAGE_KEEPOUT	Top	Creating boundary at the board level on the top etch layer. You can define this boundary at the symbol level only for Mechanical part. Used to ensure that there is no violation of placement keepout areas or high restricted areas in a design.

Getting Started with Physical Design

Classes and Subclasses in Layout Editors--Creating a Set of Split Rings Around a Complex Wire Bond Die

	Bottom	Creating boundary at the board level on the bottom etch layer. You can define this boundary at the symbol level only for Mechanical part. Used to ensure that there is no violation of placement keepout areas or high restricted areas in a design.
	All	Creating boundary at the board level on all the etch layers. You can define this boundary at the symbol level only for Mechanical part. Used to ensure that there is no violation of placement keepout areas or high restricted areas in a design.

Glossary

A

active devices

Transistors or diodes that can change the basic character of a circuit.

active side

The surface of an IC containing the I/O drivers and pads used to interface the additive process

A process that creates or screens-on a circuit by adding a conductor in a precise pattern.

alias

A user-defined abbreviation for a command. See [script files](#).

allegro

The UNIX command that provides complete design functionality, including automatic placement, routing, post-processing, and third-party database translators.

Allegro Design Entry CIS

A Cadence tool that runs on a PC and is used for PCB schematic capture.

anti-pad

A negative pad (clear, surrounded by black), usually a circle, to prevent the connection of a pin to an embedded metal layer.

APD option L

The Allegro Package Designer single die co-design product.

AP SI

Allegro Package Signal Integrity.

API

Application Procedural Interface

area optimization

A Thick/Thin-Film Resistor Synthesizer command file directive that generates resistors with the smallest possible area.

area resolution

The process Allegro PCB Editor uses to decide which constraint applies when two or more constraint areas overlap or an element such as a line extends over more than one constraint area.

Allegro PCB Editor finds all possible constraints that may apply to a spacing or physical situation, then selects the most conservative constraint value.

ASIC

Application Specific Integrated Circuit (compare; General Purpose Microprocessor Integrated Circuit)

aspect ratio

The ratio of the length to the width of a resistor. For example, if the length is 2 and the width is 1, the aspect ratio is 2:1 or 2 squares of resistance.

automatic placement

An Allegro PCB Editor function that places components in a design, based on controls provided by the user.

automatic routing

A function that automatically connects pins with ETCH/CONDUCTOR.

autosave

A built-in Allegro PCB Editor facility that regularly saves an active design or symbol. You must activate the autosave utility in Allegro PCB Editor or in your local environment file.

B

ball grid array (BGA)

A type of die component whose pins are solder balls arranged in a grid pattern.

ball pitch

The distance between the centers of adjacent solder balls of a BGA. Note that this is different from ball spacing, the distance between adjacent ball edges.

BASE

A CONDUCTOR subclass; an outer layer of a design.

bi-directional

A pin on an ECL net that sometimes acts as a load and sometimes as a driver.

blind via

A hole used to connect ETCH/CONDUCTOR subclass that does not go all the way through a design. A blind via can connect either outer ETCH/CONDUCTOR subclass to an inner ETCH/CONDUCTOR subclass. See [buried via](#). In the industry, blind and buried are often used interchangeably to describe vias that do not go all the way through the design.

board geometry

The physical definitions of the design's base material.

bond finger

A metal pad on the outer layer of component substrate to which a wire bond will be attached to form an electrical connection between the component and die.

bond finger guide path

A path which bond fingers must follow when placed. This line is where the connection point of the bond finger (usually in the center of the padstack) is placed when a bond finger is instantiated. Guide paths are any lines drawn on the Substrate Geometry / Wb_Guide_Line layer.

bond stepping

In stacked die situations, a single, unbroken wire bond may travel from one die pad to a second pad on a lower die and ultimately ends up on a bond finger on the substrate surface. This process is known as stepping.

bond wire

A wire (usually gold) which connects a die pad to its respective bond finger on the component substrate or to another die pad on another die.

BOTTOM

An ETCH/CONDUCTOR subclass; an outer layer of a design.

boundary

A line that defines the outside edge of a window.

broadside differential pair

Two nets that are routed on adjacent layers and follow nearly the same path; one on top of the other. Opposite of an edge side differential pair. Broadside differential pairs are also known as

tandem differential pairs. See [differential pair](#).

bump

See [solder bump](#).

bump pitch

The distance between the centers of adjacent solder bumps of a flip-chip IC. Note that this is different from bump spacing, the distance between adjacent bump edges.

buried resistor (passive)

Terminates high-speed nets. You can construct buried resistors by replacing pin pads with a new pad in the shape of a resistor plus a rectangle that represents the resistor paste. The insulation of the resistor is represented in the database (the lack of copper between the resistor and the power plane) and not the internal geometry of the resistor. Other methods include adding a symbol representing a resistor and creating positive artwork film for that ETCH/CONDUCTOR layer or creating a stand-alone resistor routed to the terminating pin. The latter consists of two pins: one via-like (a pin connected to the terminating pin). A second pin consists of a single layer pad that flashes on an imbedded plane.

buried via

A hole used to connect ETCH/CONDUCTOR subclasses that does not go all the way through a design. A buried via can connect any internal ETCH/CONDUCTOR subclass to another internal ETCH/CONDUCTOR subclass. See [28535](#) blind via . Blind and buried are often used interchangeably to describe vias that do not go all the way through a design.

C

capacity

The maximum number of channels between two given obstacles.

cell

Also referred to as IC cell, library cell, or standard cell. An element of a cell library; an alternative name for macro block. A standard cell is a cell designed to fit into a regular row structure of a digital IC. Its width may vary, but its height must usually be an integral multiple of the row height.

channel

The space between two obstacles required to route a single trace.

characteristic impedance

At a given instance in time, a transmission line appears to an electrical signal as a resistance

whose value is called the characteristic impedance. The resistance, capacitance, and inductance of a transmission line combine to impede the flow of charge.

check box

A check box is used on an Allegro PCB Editor dialog box to specify whether an item is to be used or selected. A check mark indicates a selected check box.

chip-on-board

A chip that is glued directly to the board. Usually, a chip or integrated circuit is enclosed in a package and mounted on the board. Bonding wires attach pinouts to pads.

circuit

A set of electronic functions, such as gates and buffers, that when connected together constitute the electronic description of a printed circuit design. When this description is provided in ASCII dialog box, it is called a netlist. Allegro PCB Editor requires a readable netlist as input for automatic design and checking.

class

A category used to identify and refer to elements in a design. It eliminates the requirement of referring to elements by layer number. You can have up to 64 subclasses that further define a class.

clip-on-chip

A chip that is glued on a board that is very small and enclosed in a package. One example is a CPU and cache memory together in the same package used to reduce delays.

chip-down

A packaging technology where the active surface of the die faces downward (cf. chip-up). May use with flip-chip on top of the component substrate (common), or wire bonding on bottom (rare).

chip scale package (CSP)

A very small packaging technology where the component is essentially the same size as the die. The die covers the entire surface of a CSP.

chip-up

A packaging technology where the active surface of the die faces upward (cf. chip-down). May use with flip-chip on the bottom of the component substrate (not common) or wire bonding on top (common).

co-design die

A die that is concurrently designed with its end component to ensure that the combination of die

and component meets all design requirements, while at the same time minimizes the overall cost of production. APD+ and IC tools work together to support co-design.

co-fired ceramics

Refers to the clay substrate .

color palette selector

An application window that controls colors used in design display on the Allegro PCB Editor desktop.

color editing

You can select colors from the color pad that is displayed when you select the color menu option. Two other dialog boxes appear as well as the color pad. Pick a color from the color pad and then select the subclass color box in the subclass dialog box to apply that color.

command

A string of characters typed at the operating system prompt that perform a specific action. See [78302](#) option .

command line

The line, identified in the console window by the > prompt, at which the user can enter commands.

component

An element that represents where a packaged electrical device will be added on the actual board. There may be many logical parts used in the front-end or schematic entry tool that represent a single package or component in Allegro PCB Editor. Additionally, the component may represent a single discrete or active electrical device.

component pin

Conductors that protrude from packages. Pins allow the component to be connected electrically to the circuits in the printed circuit design.

Condensed Macro Language (CML)

An ASCII file format used by APD+ to store extra information about I/O cells in LEF that is required in order to interpret I/O driver cells correctly when imported into the system-in-package design.

conductance

A measure of the heat transfer rate of an object for a given temperature difference across a measured area (in W/cm-deg C).

Conductor

A routing layer (Allegro PCB Editor). For example, Surface or Base. See [routing subclass](#).

conductivity

A material property that describes a heat transfer rate through a volume of the material for a given temperature difference (in W/cm-deg C).

conductors

Materials with a low resistivity that conduct electricity easily.

connection

The smallest logical unit the automatic routing tool considers when routing a net. See [28588](#) connect line.

connection point (into the IC)

Referred to as a pin in IC layout tools, the part of the I/O driver extending into the silicon circuitry. The input side of the I/O driver, connecting the driver to the internal die interconnect of the IC.

connection point (out of the IC)

Referred to as a pin in IC layout tools, the part of the I/O driver extending to the solder bump. The output side of the I/O driver. Redistribution layer routing is used to connect this point to the solder bump or wire bond diepad.

connect line

The line of ETCH/CONDUCTOR that connects two pins on a net. See [90751](#) stub.

constraint

A restriction that the DRC process applies to a physical element in a design. Allegro PCB Editor searches for constraint violations during automatic and interactive processing and flags violations with DRC markers. Allegro PCB Editor has 130 types of constraints. Each constraint type:

- Has a name (for example, Stub Length)
- Has a DRC mode that determines when the DRC process applies the rule
- Can have one or more values associated with that constraint

constraint region

A shape or rectangle on the constraint region class, which has four fixed subclasses in addition to the existing ETCH/CONDUCTOR layers: All, Inner Plane, Inner Signal and Outer layers. Three types of region objects are available: region, region-class, and region class-class.

Region relationships can be at the design level, affecting all nets traversing it, or granularly applied against class based objects.

constraint set

A predefined group of constraints organized by the behavior and type of element to which the constraints apply. Allegro PCB Editor has three types of constraint sets:

- Spacing
- Physical
- Electrical

The number and type of constraints in each set are fixed. When you create a constraint set, you give it a unique name, then specify values for each constraint in the set.

construct

An ASCII character string in a Allegro PCB Editortechnology file that starts with an opening parenthesis followed by a keyword, followed by one or more values or a nested construct, and ending with a closing parenthesis. The keywords in a technology file identify Allegro PCB Editor design parameters and constraints.

converters

See [18059](#) data translators .

crosstalk

Signal transmittal from one wire to another by electromagnetic field effects. On a printed circuit, parallel ETCH/CONDUCTOR can exhibit significant crosstalk.

cursor

An element of the graphic display controlled by the mouse and the keyboard.

D

database

An Allegro PCB Editorfile that contains complete information about a design.

data translators

These are Allegro PCB Editoroptions that provide data translation between Allegro PCB Editor and other products, including interfaces to Calma, SciCards, Prime Computervision (CV), Cadnetix, Redac, Gardner Denver, Greenfield, Appicon, and Gerber.

default

A value selected for a parameter that is displayed by Allegro PCB Editor when a dialog box is displayed on the screen or when the user executes a command.

design

In PCB Editor, a database file with a `.brd` file name extension. A design drawing usually contains two outer ETCH/CONDUCTOR subclasses (TOP and BOTTOM), internal ETCH/CONDUCTOR subclasses, padstacks, vias, edge connectors, and components. See [32188](#) printed circuit board .

In APD+, a database file with an `.mcm` file name extension. A design drawing usually contains two outer CONDUCTOR subclasses (TOP and BOTTOM), internal CONDUCTOR subclasses, padstacks, vias, edge connectors, and components.

Design Exchange Format (DEF)

An industry-standard ASCII file format for exchange of digital IC implementation data.

Design Entry CIS

A Cadence tool that runs on a PC and is used for PCB schematic capture.

design rule

A guideline that specifies any of a number of parameters for the printed circuit board. These may include minimum clearance between items that belong to different nets or connection rules. Also, these rules may include specifications for conductor, maximum length for clock lines, termination required for conductor with fast rise and fall times, and so on.

design work window

The window displayed from an option where you can create, edit, and manipulate a design drawing.

device

In Allegro PCB Editor, a device in refers to the set of information used to represent or describe a component such as the footprint, class, and number and type of pins. This information is found in the third-party device files or Cadence-formatted `pstchip.dat` file.

device file

An ASCII file that contains electrical part information. In Allegro PCB Editor, you supply this information for new parts.

die

An unpackaged chip.

DIE

Die Information Exchange format.

die editor

An editor used to edit the placement of pins for standard dies, or gaining access to the Cadence I/O Planner environment for editing co-design dies.

dielectric

Material that does not conduct electricity. It is used for insulating conductors and making capacitors.

dielectric constant

A value that represents a material's ability to store a charge when used as a capacitor.

die interconnect

The routing that connects the internal circuitry of the die, including route connections between the standard cells of a digital ASIC.

die escape

A combination of traces and vias that bring a single flip-chip die bump out a specified distance from the die edge on the desired component routing layer.

diepad

A metal contact on the die of an IC which is used to make electrical connection between the IC and the component (also called I/O pad or die pin. For flip-chip, they are called solder bumps, while for wire bond ICs they may be called wire bond pads). In IC tool terminology, wire bonded ICs are often referred to as bondpads. To avoid confusion with packaging tools, that terminology is not used in this document.

diepad pitch

The distance between the centers of adjacent diepads on the IC. Note that this is different from pad spacing, the distance between adjacent pad edges.

die pin

An alternative name for diepad.

die redistribution (or RDL routing)

The routing on the silicon substrate between the I/O drivers and the diepads.

die stack

A vertical stack of dies consisting of one or more dies, spacers, and interposers.

differential pair

A pair of signals that must be routed next to each other as closely as possible and equal in length within a certain tolerance. Usually done to match impedance or to decrease EMI. Differential pair is an example of an electrical constraint.

DIP

Dual-In-line Package.

discrete component

Typically an analog component, for example, resistor, capacitor, or inductor.

display area

The space within the boundary of a window used by the application program to display a design.

doping

The addition of an impurity that alters a material's conductivity.

drafting symbols

Leader-oriented, linear, datum, and angular dimensions (lines, text, arrows, and so on) that are stored in the Allegro PCB Editor database as drafting (.dra) symbols. Like other Allegro PCB Editor symbol types, a drafting symbol consists of lines, arcs, and text that can be individually manipulated.

Unlike other Allegro PCB Editor symbol types, only dimensioning commands create drafting symbols. No .dra files are created. Drafting symbols are created internally to enable you to more easily manipulate dimensions within a design (for example, select, move, and delete them).

drawing

A plot produced by a plotting device or a design drawing.

drawing grid

A dot matrix grid on which the user creates non-ETCH/CONDUCTOR geometries.

DRC

Design Rule Checking. A check on the design for spacing violations based on user-defined rules and standards.

DRC mode I

A user-set control switch associated with each constraint type. It has three possible settings that determine whether the constraint will be computed.

- Every time there is a change to the design (Always)

- Only on batch command (Batch)
- Never (Never)

DRC modes cannot be different for particular Spacing or Physical Constraint Sets or for different constraint areas. A single setting of the DRC mode applies to all instances of a constraint type, such as Line to Line Spacing . For Electrical Constraint Sets, however, DRC modes can be different for each constraint in each constraint set.

DRC rule sets

Minimum spacing definitions between standard design elements.

DRC rule sets by class

Determines how Allegro PCB Editor handles distances between diverse nets.

driver

A source pin on an active component where a signal originates. Important in high-frequency circuitry (ECL).

driver cell

See [I/O driver](#) .

driver terminator

The terminator on the driver end of an ECL net.

DXF

AutoCAD Drawing Exchange Format.

dynamic shape

A shape whose fill is automatically updated when design modifications are made. Shape connectivity, void generation, and design rule checking occur on these shapes when a change is made that affects the shape connectivity.

E

ECL

Emitter-Coupled Logic. In Allegro PCB Editor , refers to high-speed designing.

ECL net

A net designed using the principles of ECL. In Allegro PCB Editor, a net that has the ECL property attached to it.

edge connector

A set of surface mounted pins on the edge of a layout. Edge connectors are used to connect designs to other designs, or to external devices such as front panels.

electrical constraint

A rule or limitation placed on the electrical behavior of a signal, rather than on the physical realization of the signal. Examples would be timing constraints or impedance requirements. A routing length constraint is a physical constraint that may be derived from an electrical constraint.

edge differential pair

Two nets that are routed on the same layer and run beside each other. Opposite of a broadside differential pair. Edge side differential pairs are also known as adjacent differential pairs. See [differential pair](#).

electrical model

An electrical representation of an existing packaged IC used for board-level simulation. Electrical models can be simple such as RLC models, or detailed based on three-dimensional multi-frequency analysis.

elevation view

A view of design entities as seen from either their North, East, South, or West side providing a visualization using a combination of either the X- and Z-axis, or the Y- and Z-axis coordinate systems. Also referred to as the side view

embedded plane

An internal plane. See [plane layer](#).

EMI

Electromagnetic interference emissions of electromagnetic waves by a circuit that may interfere with other circuits or electronic devices.

Engineering Change Order (ECO)

A mechanism for passing design change requests between different teams and mediating the impact of the changes.

environment

Parameters that control the Allegro PCB Editor operating environment. Default settings can be user-defined to meet site requirements.

etch

Conductive material used in manufacturing a design.

ETCH

A routing class.

ETCH subclass

A routing layer. For example, TOP or BOTTOM. See [routing subclass](#).

etch T

A connection that is routed between a pin and another connection. See [stub](#).

execution

An attempt by an automatic tool to complete a step, for example, autorouting, auto swap, and auto placement.

F

fails

A connection that was attempted by the automatic routing tool but was not completed.

failure rate

In Allegro PCB Editor, quantifies hardware reliability for components. It indicates the number of times a component fails in one million hours of operation. See [MTBF](#).

fanout

See die escape.

fanout routing

The routing required in a component for conducting lines to reach the edge of the IC from bumps in the interior. Also known as escape routing.

field

In a dialog box, displays the text or numeric value for a parameter. In an application menu, a field contains the name of an application option. In pop-up or pull-down menus, a field is a menu option.

fill-in fields

Fields that have a single underline next to the name of the field and are displayed where you are required to supply information. An icon may be attached that displays a pop-up menu that contains one or more choices used in the field.

First Encounter (FE)

A Cadence digital IC implementation tool.

flash

In photoplotting, the process of creating pads using standard apertures.

flip chip

An unpackaged integrated circuit that connects to a hybrid circuit by means of solder bumps on its faces that correspond to its pin-outs.

floorplanning

Allows you to specify locations on a design for automatically placing components. See [placement evaluator](#).

footprint

The physical and external interface aspects of a device placed on a PCB or in a component. It includes the I/O pads for interfacing to the device and the physical body shape of the device, but does not include any of the internal structure of the device or its component. The footprint of an IC in its component layout includes the diepads (solder bumps or wire bond pads), but does not include I/O drivers or other internal layers of the IC.

form

A dialog box that is displayed when you select some menu options. A dialog box sets attributes and operating characteristics for a design.

format symbol

A set of information contained in a file with an `.osm` file name extension used to create the drawing format and represent standard drawing forms such as a border, title block, notes, and all applicable drawing information.

fromto

A routing term for pairs of certain design elements that are scheduled to be interconnected. Elements can be component pins or rat Ts. In Allegro PCB Editor, a ratsnest represents a fromto.

front end

Refers to the logical portion of a design flow. Usually includes logic specification, simulation, synthesis and timing analysis, and sometimes floorplanning.

funcdes

The identification code of a function or gate.

function

A logical unit of an electronic part such as an integrated circuit, also referred to as a gate.

function designator

The identification code for a function or gate.

G

gate array

A geometric pattern of basic gates contained in one chip. These gates can be interconnected during manufacture to form a complex function that can be reproduced.

gate

The schematic description of the logical symbol or symbols in a device.

Gate Ensemble (GE)

A Cadence IC place and route tool.

Genesis

A new database for IC tools that may eventually augment or replace LEF/DEF. Version 2 is often referred to as G2.

glossing

Applications that perform post-processing functions including increasing the width of connections to ensure greater manufacturing reliability, converting corners to arcs, and adding dielectric patches to hybrid designs to insulate intersecting connection.

green tape

DuPont's process for co-fired ceramics. The color of the unfired substrate is green.

guideports

Optional visual checkpoints that suggest potential connections for unrouted nets that cross partition boundaries after a master designer creates design partitions, but prior to exporting them during design partitioning.

H

hard macro

A fixed block-level abstract. Usually reused IP block, for example, SERDES, PLL. See [macro block](#).

HDL (Hardware Description Language)

A high-level language used to describe hardware systems or ICs in a textual format.

Help

Online help describing Allegro PCB Editor in a separate window. The `helpcmd` and `helpmenu` commands entered on the command line display the command table for that design work window and menu option-to-command correspondence.

hot spot

A spot of color at the center of each component. The color of the hot spot indicates the operating temperature range of the associated component.

hug-preferred bubble mode

A bubble mode used by Allegro PCB Editor when encountering an obstacle. Allegro PCB Editor tries to maintain the geometry of the existing ETCH/CONDUCTOR to avoid spacing violations by hugging existing ETCH/CONDUCTOR.

hybrid circuit

A special form of microelectronic design that interconnects passive and active devices. A hybrid circuit responds to semiconductor chip integration and packaging needs. It combines the use of thick film used with printed circuits and thin film used with integrated circuits. Multilayered ceramics or co-fired ceramics is another common hybrid.

I

IC

An integrated or microcircuit (monolithic) that consists of interconnected elements inseparably associated and formed on or within a single substrate to perform an electronic circuit function.

IC cell

See [cell](#).

ICD

The IC Digital business unit of Cadence.

ink

See [paste](#).

Insight

An expert application that automatically sets operating parameters for use during automatic placement and routing.

interfaces

See [18059](#) data translators .

interposer

A substrate with a single conductor layer that is used in the manufacture of a die-stack to support die connectivity, especially to provide the capability to wire-bond dies whose die-pad positions create wire-bond lateral spans that are beyond the physical limits of a wire-bonding machine.

I/O driver

Also called I/O buffer. Inside an IC, the standard cell driver connected to the die pad that acts as the interface between the IC circuitry and the pad for external connection to the IC. Ordinary die pads use a one-to-one mapping between pad and I/O driver. Other die pads such as corner cell or differential pair pads may share a driver. I/O drivers are usually defined within a macro block. The term I/O driver often refers to the macro block or cell containing the I/O driver as well as to the driver itself.

I/O Planner (IOP)

The new FE-based window that provides co-design die editing for APD+.

I/O pad

An alternative name for diepad. In IC tool terminology, I/O pad is sometimes used to refer to the whole macro block containing the I/O driver and possibly the diepad. To avoid confusion, this document does not use the term I/O pad, and uses diepad for the pad itself, and I/O driver for the macro block containing the I/O driver.

I/O pad rings

In wire bond designs, the set of diepads of the IC. In flip-chip designs, a number of rows of solder bumps, spread along the periphery of the chip, which interface the I/O signals for the IC to the component. The pad ring includes not just the contact pads or bumps themselves, but also the I/O drivers and associated connections between the buffers and the pads. Each row of I/O pads and associated buffers circling the IC is referred to as an I/O pad ring.

ISHM

International Society for Hybrid Microelectronics

isotherm

A line or curve that connects points of constant temperature. The color of the line indicates the range of temperature for the locations along the line.

J

jog

A piece of ETCH/CONDUCTOR that runs perpendicular to most ETCH/CONDUCTOR on that ETCH/CONDUCTOR subclass.

K

keepin package

A constraint that specifies the area in which Allegro PCB Editor should place all packages.

keepout package

A constraint that specifies the area in which packages are forbidden.

L

layer

An insulated plane in the design that contains lines of ETCH/CONDUCTOR.

laser trimming

The removal of resistive material by laser that raises the resistance value of a film resistor.

LEF/DEF

Library Exchange Format/Design Exchange Format.

Library Exchange Format (LEF)

A file format for storing generic blocks of IC functionality. LEF together with DEF forms an industry standard ASCII file format for exchange of digital IC implementation data. LEF contains the description of abstracts for library cells used in the DEF.

line

See [connect line](#).

line fattenig

A glossing application that increases the width of connect lines wherever possible for greater manufacturing reliability. See [glossing](#).

line ripup

A feature of the automatic router that removes existing connect lines to make room for new connections.

list picker

A scroll area that displays a list. You can select an item from the list or scroll through it to review

other choices. The chosen item displays in an identification field. Alternatively, keyboard input is permitted.

load

Any pin on an ECL net that is not a driver or a terminator.

load terminator

The terminator on the load end of an ECL net.

log

A file that Allegro PCB Editor creates as a by-product of many processes. For example, when you execute an option in an application menu, Allegro PCB Editor creates a log file to record events that occurred during processing. See [reports](#).

lossy

A transmission line that has resistance, causing it to dissipate some power as current passes through it. See [ohmic loss](#).

lump load

A model of transmission line using a combination of capacitor, inductor, and resistor.

lump loading ratio

The longest distance between any two consecutive loads in a net divided by the total length of the net.

M

macro block

A reusable cell placed inside an IC that contains built-in diepads. The library cell for an I/O driver may also be referred to using the generic macro block terminology. This document often uses I/O driver as a synonym for the macro block containing the I/O driver. Any complex cells containing several diepads, or I/O drivers for several diepads, are referred to as macro blocks, rather than library cells or I/O drivers. Macros of class PAD or ENDCAP contain I/O pad information. Most often these are hard blocks.

macro cell

A block-level abstract containing standard cells.

manhattan distance

The orthogonal distance between two points. The distance calculated as the sum of the distance between the points along the X axis and the distance between the points along the Y

axis. DX + DY.

map

Associates a component with a particular row and column.

mask

A pattern on glass or fine mesh screen that serves as the template for exposing thin film photoresist or for screening thick film material.

master database

Design (.brd, or.mcm) into which the master designer imports and exports partitions.

master designer

Designer in lead role responsible for the design (.brd or.mcm) and the only designer allowed to create partitions and import and export them to partition designers.

mechanical symbol

A set of information contained in a file having a .bsm filename extension used to define mechanical and graphic elements on a design drawing. Typically, design symbols represent non-electrical elements, for example, design outlines, plating bars, mounting holes, or card ejectors. Mechanical-only fixtures with drill holes are represented by pins with no pin numbers. design symbols do not have a reference designator label.

menu option

Any of the choices that appear in a menu.

message area

An area in the command line used by Allegro PCB Editor to display messages to the user. Up to three lines of text can be displayed and a scroll bar can be used to display messages outside the confines of the display area.

Message Passing System (MPS)

The standard Cadence mechanism used to communicate among the tools. MPS is used to exchange messages between APD+ and I/O Planner session that it spawns.

model library

A library of electrical models of existing packaged ICs used for board-level simulation.

module

A module (.mdd) file contains a selected portion of a board that is saved in a way that it can be placed again in its entirety on a board. All routes, components, vias, layers, and so on for the

selected module are stored. A module is similar to a component in that you can place, delete, and move it multiple times with or without logic that represents it. Design Reuse allows you to take full advantage of modules and nested modules by reusing logical hierarchical blocks and the physical modules that represent them multiple times in a design.

MTBF

Mean Time Between Failures. In Allegro PCB Editor, a term used to quantify hardware reliability for designs. MTBF indicates the number of hours of design operation before a failure. See [failure rate](#).

N

net

Any set of pins and vias that are logically connected.

net layer rules

These Allegro PCB Router rules establish line width, neck width, and whether ETCH/CONDUCTOR is allowed on an ETCH/CONDUCTOR layer.

netlist

An ASCII text file that provides the electrical blueprint for the circuit design.

noise immunity

The worst case between output voltages produced by a driver pin and input voltages that a receiver pin interprets correctly when operated in an ideal environment. This presumes equal junction temperatures and no other sources of signal noise other than typical device manufacturing variations.

noise margin

A more thorough calculation of noise immunity, accounting for expected noise sources. This is the "margin of safety" that remains after estimated noise levels are subtracted from the ideal noise immunity.

net schedule

A preferred order for the interconnection of a net's component pins. The schedule may be user-defined or determined by Allegro PCB Editor. When determined by the software, scheduling is based on component placement, types of component pins in the design, timing rules, and so on.

O

ohmic loss

Voltage drop across a resistor as current passes through it. Design ETCH/CONDUCTOR has measurable resistance and, due to the signal current, some voltage is lost on its way to the receiver pin.

OpenAccess (OA)

An open source database format and schema that has been adopted by Cadence and other companies for representation and interchange of IC tool data.

option

A menu choice that you select from an application menu to display a dialog box or execute a process.

Options

In Allegro PCB Editor, a tab display in the right side of the Allegro PCB Editor window. The fields in the Options tab change to match the command or option you have selected. Fields typically identify the class, subclass, and color assigned to the subclass.

P

package

A physical symbol designated as Drawing Type package in the Symbol Editor. Typically used as database element for components that have electrical connectivity. Stored as a library element with an extension of .psm. A package contains the padstacks, labels, outline, silk screen and so on. It visually represents the component in Allegro PCB Editor. Note that a single symbol or multiple logical symbols may comprise a single package.

package geometry

Graphic elements that make up a physical component, commonly referred to as shapes or symbols.

package interconnect

See [pin escape](#) .

package library

A library of existing component designs that are known to be good.

package parasitics

The impact of the component material and geometry on the integrity of the signals of the

packaged IC.

package symbol

A set of information contained in a `.psm` file used to represent an electrical component. The symbol is a physical representation of the logical parts in a schematic design, such as a dual in-line package (DIP), resistor, capacitor, or edge connector. Package symbols have a reference designator label and at least one pin number.

pad

See [75267](#) SMD pad .

padstack

A list of all data for each pad definition in the design drawing; each pin and via refers to a padstack for size, shape, and drill information.

Padstack Designer

A tool that lets you create and edit padstacks and save them to your design, to a library, or to both at once.

parameters

Text and numeric values that control what you see on the screen and the functions performed by automatic programs.

partitions

Separate physical areas of the design database divided by the master designer to allow several designers to collaborate and expedite the design schedule. The master designer assigns each designer a partition, and then exports the design to multiple designers.

partition boundary

A closed polygon that defines the design section assigned to the partition designer. The polygon cannot overlap or lie inside another partition boundary, or contain voids or arc segments in the outline.

partition database

A copy of the Allegro PCB Editor database from which it was exported to which an extension of `.dpf` (Design Partition File) appends.

partition designer

A designer in a subordinate role to lead designer, responsible for completing an assigned partition.

partitions

Separate physical areas of the design database divided by the master designer during design partitioning to allow several designers to collaborate and expedite the design schedule. The master designer assigns each designer a partition, and then exports the design to multiple designers.

passivation opening

In wire bond dies, the wire bond connects to a metal pad which is exposed by the top dielectric/insulating layer of the IC. This hole in the dielectric material for an individual die pad is the pad's passivation opening.

passive devices

Devices such as resistors, capacitors, and inductors that either absorb or store energy.

Passive device integration (PDI)

A procedure whereby passive devices for decoupling, bypass, terminations, and so on, are added. These passive devices can be added either to the IC itself, or into the component. Cost, complexity and design cycle time tradeoffs may influence the decision of where to integrate these devices.

paste

A screenable thick-film material. The three categories of thick film are conductors, resistors, and dielectrics. Also known as [ink](#).

path

A line of travel between two pins in a net.

PCB Editor

A tool for the physical layout system for PCB design.

physical cell

An instance of a cell, such as an I/O driver, that does not exist in the netlist (Verilog) of an IC design. A physical cell is a driver added to the design by the *Add Driver* command of the Die Editor. Sometimes, the drivers added this way during a feasibility study for an IC are called dummy drivers.

physical constraint

A rule or limitation placed on the physical realization of a signal. These are often derived from an underlying electrical constraint. An example is a routing length constraint.

plan view

View looking down onto a design from above showing design entities and their relationships along the X- and Y-axes. Also referred to as the top view.

pick

1. The act of positioning the cursor on a graphic element such as an option and clicking (pressing and releasing) the left mouse button.
2. A phase of execution of the *Swap* application.

pin

The contact point and electrical interface between a component and a PCB. In the case of BGA packages, these are called solder balls. Also sometimes used to refer to the analogous diepads used for interfacing an IC to its component, or the contact point of a macro cell where the dielectrical interface for the cell is made within an IC. The term pin is used for many different purposes in the EDA industry.

pin escape

A line of ETCH/CONDUCTOR and a via used to connect surface-mounted pins to internal ETCH/CONDUCTOR subclasses.

pin pair

A set of two design elements, either component pins or rat Ts, on a net or extended net ([xNet](#)) that is established for the purpose of specifying a timing constraint. Pin pairs do not necessarily form a [fromto](#), since the elements do not have to be scheduled for direct connection.

pin swap

A process to exchange the locations of two pins that are electrically identical.

pin-to-pin connection

A signal path from a particular driver pin to a particular receiver pin. For example, a network with two drivers and three receivers has six possible pin-to-pin connections.

placement

An Allegro PCB Editor function that executes the placement of components in a design drawing. Allegro PCB Editor provides both interactive and automatic placement capabilities.

placement evaluator

Allows you to judge where routing channels are blocked. The placement evaluator calculates statistics for routing a design. The placement evaluator analyzes the potential routing success of a placed design. You can start testing a placement for routability as soon as you place components in the design.

placement grid

A matrix of lines that you create using the Grid option in the Autoplace menu and edit using Edit commands. The grid defines locations for automatic component placement. Interactive

placement uses the non-ETCH/CONDUCTOR grid that you create using the Define – Grid dialog box.

plane layer

A conductive layer in the cross-section editor designated as layer type "plane". These layers are typically used to create shapes for the purpose of Power and GND distribution.

planes

The routing layers within the component substrate that routes the signals and distributes the power from the die to the host PCB.

PLEF/PDEF

Parametric library/design exchange format, parametric LEF and DEF is an extension of the LEF/DEF language that lets you create parametric macros in the library. Parametric macros are generic versions of regular macros.

pop-up field

Displays multiple choices if you either toggle the field (click on it several times) or hold down the left mouse button in the field to display the pop-up. To select an item, release the mouse button on the highlighted item.

pop-up menu

Any of the choices that appear in a pop-up menu that appears when you pick a menu option.

power rings

Conductor rings on a component surrounding the chip that are used to bond power and ground nets and are part of the power distribution network for wire bond packages.

printed circuit board

A single, thin piece of material comprised of many laminations of a substrate, usually epoxy, on which an electrical circuit is printed, usually in copper. Online, an Allegro PCB Editor design.

property

An entity which can be attached to an object to describe some aspect of the object that was not previously described.

R

radio button

A group of buttons that you can select by toggling. A small, filled-in circle indicates a selected radio button.

ratsnest line

In a design drawing, a line that shows a logical connection between two pins, connect lines, or vias. Elements connected by the same ratsnest line are part of the same net. The ratsnest shows the circuit logic and, for ECL circuits, the order in which pins are to be connected.

rat T

A database object used to insert a branch in a net's schedule at some point other than at a component pin. A rat T has a physical location that is often an approximate location for a 'T' or a via in the net's physical interconnect.

rat T cluster

A group of component pins on a single net that are logically connected (that is, specified by the [net schedule](#)) indirectly through one or more rat Ts. A pin can belong to more than one rat T cluster.

redistribution layer (RDL)

A routing layer of conductive metal within an IC that connects the diepad or solder bump to a connection point pin on an I/O driver.

reference designator

The designator, or identification code, for a component.

reflection

When a signal traversing a wire meets a sudden change in characteristic impedance, some of the signal is reflected backwards. This is similar to the splash-back caused when a stream of water is passed through a mesh screen. In ETCH/CONDUCTOR, multiple reflections are possible, producing ringing and overshoot.

refresh symbol

This command replaces existing symbols in a design with newer versions of symbols from a library. Options indicate the symbol type to refresh. You can refresh package symbols, mechanical symbols, a list of symbols that you provide in a text file, or all symbols.

regular pad

A positive pad (black) with a regular shape (circle, square, rectangle, oblong, shape, or aperture flash).

resistance

Extent to which an instance resists the passage of heat.

resistor packs

Components that contain many resistors. On ECL designs, the resistors in resistor packs are used as terminators.

reports

User-defined files that provide specific information about a design. For example, you may execute the report command from the operating system to create an ECL Loading Report, a file that lists any nets that do not meet design specifications. See [74195 log](#).

RF

Radio frequency, typically high frequency analog designs used in wireless applications.

riput

See [54206 line ripup](#).

room

A user-defined area of the design that is treated separately by several automatic programs. For example, the automatic placement program uses rooms to group related components. See [window](#).

route keepin

A route constraint. An area you must add to the design to tell AutoRoute where to contain the routed connections.

route keepout

A route constraint. An area you can add to the design that tells AutoRoute where not to route connections.

route

The ETCH/CONDUCTOR elements (clines, vias, and shapes) that, when combined, form a connection from one pin to another. An incomplete route implies there is a break between the source and destination element.

routing

The conductive paths and vias used to connect pins of various components together; the connection between the pins are defined by the netlist description of the design.

routing area

The area of the design drawing in which you wish to route. Also, an area of the design drawing you can route separately from the rest of the design drawing, such as a room or a window.

routing channels

Horizontal and vertical paths that connect routing grid points.

routing grid

A matrix of dots or grid points that AutoRoute uses to route connections.

The space between routing grid points.

routing layer

A layer on which connections are routed. See [54334](#) routing subclass and [ETCH](#) subclass .

routing subclass

In Allegro PCB Editor, any of the ETCH/CONDUCTOR subclasses that you have designated for routing. Routing subclasses are a subset of the ETCH/CONDUCTOR class. (All routing subclasses are ETCH/CONDUCTOR subclasses, but not all ETCH/CONDUCTOR subclasses are necessarily routing subclasses.) ETCH/CONDUCTOR subclasses that are not routing subclasses are just unused routing layers. See [ETCH subclass](#) and [36294](#) routing layer .

rubberbanding

A feature of interactive commands where, as you move an element of the design drawing with the mouse, lines attached to it stretch as you move.

rules-driven design

User-defined design characteristics that can be specified by the schematic that are recognized by Allegro PCB Editor and determine processing results.

S

scheduling

The process of creating and updating the interactive ratsnest to reflect the order in which pins are to be routed in an ECL net. Schedules are established in a netlist.

script files

Scripts let you perform repetitive tasks in Allegro PCB Editor in a timely fashion. You can build a script by recording and executing the commands that you want the script to execute. You can use scripts to set up dialog boxes for routing, placing, and artwork or executing a series of check plots. Scripts can call other scripts.

scroll area

Scroll areas are used to display data that cannot be displayed within a single window.

scroll bar

A band along the right side of a window that is used to display the contents of a drawing or file that does not fit within the confines of the window.

search pin

In an ECL net, the pin from which the closest terminator is searched, even if that is not the pin to which the terminator is added.

shelf

A term that was commonly used when mounting a wire bond die inside a cavity of the component. The sides of the cavity looked like the seating rows of a Roman Coliseum. These rows are termed shelves, and are where the bond fingers exist on the component substrate. Thus, all the bond fingers on the same shelf were those at the same height from the bottom of the cavity. Although most wire bond dies are mounted on the surface of the component substrate today, the term bond shelf is still used by many to refer to the bond fingers which follow the same guide path around all four sides of the die.

shove-preferred bubble mode

A bubble mode used by Allegro PCB Editor when encountering an obstacle. Allegro PCB Editor tries to maintain the geometry of the existing ETCH/CONDUCTOR to avoid spacing violations by shoving existing etch.

signal

The electrical characteristics of a single circuit within a piece of electronics such as an IC, component, or PCB. Signals travel over point-to-point electrical connections realized by conductor paths such as copper or aluminum traces or gold wire bonds. Also called a net.

signal analysis

An Allegro PCB Editor option that predicts where layout-dependent noise problems such as crosstalk and reflection might occur.

Signal Integrity (SI)

The SPB business unit tools designed for high speed signal integrity engineering.

signal noise

Unwanted voltages that cause a received voltage signal to differ from the signal originally transmitted.

silicon substrate

The silicon "wafer" onto and into which the IC circuitry is placed.

site

An object in a LEF file that describes the physical dimensions of a macro block without defining the internals of the block. Sites may be placed into an IC layout as place-holders for placements of macro blocks of a certain size and shape. Normally these are grouped into rows.

skip

A connection that is not attempted by AutoRoute.

slide bar

The slide bar icon is positioned to the right of a fill-in field and displays a minimum and maximum number at either end of a horizontal bar. These numbers appear when the icon is selected. You can select from a range of values by sliding the bar with the cursor.

SMD

1. Surface-Mounted Device
2. A technology using surface-mounted components which have pins that are glued to the surface of a design. Designs that contain SMDs can have components on both sides. See [through-hole component](#).

SMD pad

A piece of ETCH/CONDUCTOR on TOP/BOTTOM where an SMD component pin is connected to the design.

SMD pin

A component pin that has a component pad belonging to only one ETCH/CONDUCTOR subclass—either TOP/BOTTOM.

solder ball

The pin of BGA packages. It is a ball of solder located on the bottom of the component that is bonded to metal contacts on the surface of a PCB. It makes the electrical connection from the PCB to the component.

solder bump

The solder contacts on the active surface (typically the bottom) of a flip-chip, whereby the chip is fastened to and electrically connected to its component. The more generic term diepad can also be used to refer to the bumps of a flip-chip.

source

A driver.

spacer

Manufactured or molded blocks or deposited material (including adhesives, epoxies, and eutectics) that are assumed to be rectangular in shape and provide clearance, or adhesion, or both between dies or other die-stack objects. Spacers are necessary as part of the process of manufacturing a die-stack.

SPB (Silicon-Package-Board)

The Cadence name for the Research and Development division responsible for packaging and PCB design tools.

SP&R

Synthesis, Place and Route, one phase of digital IC layout design.

sputter deposition

Exposing the chip or board to atomized metal being sputtered at the chip that sticks where there is no mask.

staggered

A matrix-like arrangement of pins, bumps, pads or balls into one or more rows or columns where the elements of two consecutive rows or columns are offset from each other. A regular staggered arrangement is similar to having the elements only located on the white squares of a chessboard, with none on the black squares.

staggered via

A via that spans more than two layers and adds a cline (user-defined stagger size) and another via for each set of layers. One through-drill can produce seven vias and six clines.

standard cell

A reusable component of IC layout (for example, I/O driver). These are usually gathered together into standard cell libraries for use by IC place and route tools. Also sometimes called macro blocks. Designed to fit into a regular row structure of a digital IC, its width may vary but its height must usually be an integral multiple of the row height.

standard die

A die that has already been designed, such as memory chips from external vendors or internal dies already in production. The standard die also refers to any die with fixed die bumps that is not currently being designed along with the component. A standard die is sometimes referred to as an off-the-shelf or third-party die.

static shape

A solid or cross hatched shape used for critical handcrafted conductive areas that you do not want modified automatically.

status area

A three-line area on the design window that displays information about the current activity. The first and second lines display the current directory and filter. The third line identifies the current command, or indicates "idle" if no command is active.

status message

The message displayed in the status area of a design work window. When you are using an interactive tool (for example, the Add Line option), the status message reports the current command ("Add Line"). When you are executing an automatic program (for example, AutoRoute), the status message reports statistics indicating the progress the program has made. See [MTBF](#).

step

A phase of routing with a distinct function or a goal and a unique set of parameters and number of executions defined for accomplishing that goal.

stub

A stub is a physical end pin, connecting to a physical, not logical, end pin deviating from a pin path as shown below. Unscheduled logical end pins become physical end pins once they're routed. An stub-length error occurs when the stub length requirement defined in the Electrical worksheet in Constraint Manager (*Setup - Constraints - Electrical*) is not met.



subclass

Further defines a class. You can define subclasses for a class. Each class can have up to 64 subclasses.

substrate

The material with which an IC, printed circuit, or hybrid circuit starts. For instance, silicon, GaAs, fiberglass, or ceramic.

subtractive process

This process creates a circuit by etching away unwanted conductors already on the substrate.

SURFACE

A CONDUCTOR subclass. One of the outer layers.

surface-mounted pin

See [SMD pin](#).

swap

Swap Function: Exchanges the locations of two functions that are logically identical, either within a component or between components, to minimize the average net length. You can

perform function swap either automatically or interactively in Allegro PCB Editor.

Swap Pin: Exchanges the locations of two pins within a function that are electrically identical to minimize the average ratsnest crossings. You can perform pin swap either automatically or interactively in Allegro PCB Editor.

Swap Component: Exchanges two components in Allegro PCB Editor to improve design placement.

switch area

SWITCH_AREA_TOP and SWITCH_AREA_BOTTOM are areas in which all etch is routed in the direction perpendicular to the preferred direction of most of the etch on that ETCH subclass.

symbol

A graphical drawing and set of data that represents a design element. There are four kinds of symbols: package (.psm), mechanical (.bsm), format (.osm), and shape (.ssm including flash or .fsm). Package symbols are electrical components or devices. Mechanical symbols can be card outlines, mechanical parts, or mounting holes. Format symbols are page size formats, graphics, logos, assembly/fab notes, cross section diagrams and so on. Shape symbols are filled polygons used for customer pads.

System Connectivity Manager (SCM)

A new logic design tool based on TDD for designing APD+ connectivity.

T

TAB

Tape-Automated Bonding. A method for attaching a chip to a substrate for chip-on-board.

technology file (or tech file)

An ASCII file that can be read into a design to specify user preferred units, constraint and parameter values, and user properties.

temporary driver

Sometimes referred to as a dummy driver. An I/O driver cell that is added to an IC design during the feasibility phase with the intent that it will ultimately be replaced by a corresponding driver cell of the IC design. When the new driver cell is available, all instances of the temporary driver cell definition are replaced by instances of the real cell definition.

terminator

A resistor pin where the other pin is attached to a negative voltage. Terminators are used to eliminate signal reflection on high-frequency (ECL) nets. The device file for a terminator always

contains, on one line, PACKAGEPROP TERMINATOR_PACK.

terminator assignment

The process of assigning terminators to the load end and the driver end of every ECL net that has a LOAD_TERM_VAL and/or DRIVER_TERM_VAL property attached to it.

thermal analysis

A Allegro PCB Editor option that lets you analyze the thermal conditions resulting from current design placement and the boundary conditions you specify, and retrieve junction and case temperatures (J_TEMPERATURE) for the components in the design.

thermal-relief pad

A negative pad (clear, surrounded by black), often created with a special aperture flash, to connect a pin to an embedded metal layer that distributes a voltage, such as a power or ground.

thermal shift

A temperature-induced change in operating voltage. A silicon junction at room temperature operates at 0.6 volts. This value increases about 2 millivolts for every 1C of junction temperature rise. The junction temperature difference between driver and receiver is responsible for thermal shift in logic devices.

thick film

A hybrid circuit technology that selectively deposits materials on an insulating substrate. Several masks or layers occur on one or more metal or resistor prints. The conductor, dielectric, and resistor inks are screen-printed in their final circuit pattern and fired at temperatures up to 1000 degrees Centigrade on the ceramic substrate. Thick film is often used to create printed circuits.

thin film

A hybrid circuit technology that deposits metals and resistor materials across a substrate and then removes material through photoetching. The conductor, dielectric, and resistor films are vacuum- or vapor-deposited on a substrate in sheets. The circuit pattern is photolithographically masked and chemically etched. Thin film is more sensitive to assembly processes and more costly than thick film. It is often used to create integrated circuits.

third-party

A drawing or schematic generated by an automated or mechanical process other than a Cadence tool.

through-hole component

A component that has pins that go through all layers in a design. The pins are adhered to the design with solder.

tier

A tier of bond wires is all those which share a common loop profile. That is, all the bond wires are at a common height.

title bar

A band along the top of a window that displays the name of the window and information about that application.

TOP

An ETCH/CONDUCTOR subclass. One of the outer layers.

transmission line

An electric conductor exhibiting series inductance and shunt capacitance distributed along its length. A signal must charge up each chunk or inductance and capacitance before it is passed long to the next chunk, thus reducing the propagation velocity.

U

user-defined net

A net for which you establish the pin order by using the \$SCHEDULE keyword in the \$NETS section of the netlist. The schedule program does not change the pin order.

user unit

The unit of measure you select when creating a new design. The fill-in field for user units is in the Drawing Parameters dialog box. Mils is the default; other choices include inches, millimeters, centimeters, and microns.

V

Verilog

A high level hardware description language (HDL) that is owned by Cadence but is now an industry standard.

vertex

A logical point at which a line is ended and restarted. A vertex is located at each change of direction on the line.

Very High-Level Hardware Description Language (VHLHDL)

An industry-standard HDL that is widely used in Europe.

via

An opening in a dielectric layer that connects adjacent conductor layers. In Allegro PCB Editor, a via is a plated-through hole with ETCH/CONDUCTOR on every ETCH/CONDUCTOR subclass. Vias make it possible to route a single connection through more than one ETCH/CONDUCTOR subclass. Also called a feedthrough.

via pitch

The closest allowable center-to-center distance between vias.

Viable

A Cadence analysis tool that predicts design reliability. Viable uses project, library, method, setup, and template files.

via structure

A combination of design elements that you can treat as a single via entity. You can create a via structure from a single via or connect line, a via and a connect line, or multiple vias of different pin sizes and multiple connect lines of different widths. Via structures provide an efficient way to create fan-outs, particularly in flip-chip designs containing large numbers of nets and highly regular patterns of die pins and BGAs.

via grid

An optional user-defined matrix of dots representing locations where AutoRoute can place vias. Without a via grid, AutoRoute uses route grid points to place vias.

via keepout

A route constraint that specifies to Allegro PCB Editor the area in which vias are forbidden but ETCH/CONDUCTOR is allowed.

visibility

Controls items that are displayed on the screen.

W

wedge

An alternative name for bond finger.

window

Any section, usually rectangular, of the graphic display where, if the cursor is within its borders, the mouse and keyboard assume functions that are different from their functions in the surrounding area. Examples include an application menu, dialog boxes that are displayed when you select the Param option, and borders that surround the design drawing. Also a user-

defined area in which an automatic process is executed.

window cursor

The crosshair that is displayed when the cursor is positioned inside the boundaries of a design window. You can change the cursor shape using the Status dialog box.

wire bond

The combination of a bond wire and a bond finger. Together, they form a wire bond which is a connection from a die pin to the component substrate.

wire bond group

A set of bond fingers and wire bonds which adhere to a common set of physical characteristics. A logical group of objects that have fingers which use the same bond finger padstack, wires of the same diameter and wire profile, and so on. These could include all those wire bonds going out to a power ring, all those following a specific arc and so on.

wire bonded packaging

A packaging technology where the die is fastened to the substrate with its diepads not making contact with the substrate. Wire bonds are used to make electrical connections from the diepads to bondpads on the substrate surface (cf. flip-chip).

wire profile

The general 3D path which a bond wire follows. This refers primarily to the general height and level of curvature of the path, rather than the 3D points occupied by a single wire. Thus, a profile defines the curvature of a set of bond wires. For example, you may have two wire profiles in a design - a lower profile for power/ground ring bond wires and a higher profile for the bond wires connecting to the bond fingers.

Workflow Manager

Interface a master PCB designer uses to manage all sections, or partitions, of the primary (master) design, after they are created with Place – Design Partition – Create Partitions (partition command). For each partition created, an entry appears in the Workflow Manager. An extension of .dpf appends to partition files.

work window

Used to perform design tasks not specifically related to a design drawing that control how a drawing is manipulated. See [design rule](#).

X

xNet

Extended net; a net composed of a group of nets in a system. The nets may be on one or more boards or modules. Typically, an XNet is composed of nets that are separated by passive devices such as resistors, connectors, cabling, etc.