

Protel 99 SE

***The complete Board-Level Design System
for Windows 95, 98 and NT***

Includes:

Design Explorer

Schematic Capture

Circuit Simulation

PLD Design

PCB Layout

PCB Autorouting

Signal Integrity Analysis

Protel 99 SE transforms your desktop PC into a complete, integrated, 32-bit printed circuit board design environment – providing everything you need to take your electronic design idea from concept all the way through to a completed board, regardless of the complexity of the circuit.

Software, documentation and related materials:

Copyright © 1994, 2001 Altium Limited.

All rights reserved. Unauthorized duplication of the software, manual or related materials by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permissions of Altium Limited.

Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment.

The Protel logo is a registered trademark of Altium Limited. Design Explorer, SmartDoc, SmartTool, and SmartTeam and their logos are trademarks of Altium Limited.

Microsoft, Microsoft Windows and Microsoft Access are registered trademarks of Microsoft Corporation. Orcad, Orcad Capture, Orcad Layout and SPECCTRA are registered trademarks of Cadence Design Systems Inc. AutoCAD is a registered trademark of AutoDesk Inc. HP-GL is a registered trademark of Hewlett Packard Corporation. PostScript is a registered trademark of Adobe Systems, Inc. All other brand or product names are trademarks of their respective owners.

Printed by Star Printery Pty Ltd

Contents

SECTION 1	1
SETTING UP AND GETTING STARTED	1
<i>Printed Circuit Board Design with Protel 99 SE</i>	3
<i>Installation</i>	6
<i>System Requirements</i>	6
<i>What is Supplied With Your Protel Product?</i>	6
<i>Installing the Software</i>	7
<i>Enabling the Software</i>	7
<i>Register Your Software</i>	7
<i>Review the README Document</i>	7
<i>Document Conventions</i>	8
SECTION 2	9
THE DESIGN EXPLORER	9
<i>The Protel 99 SE Design Explorer</i>	11
<i>How to work in the Design Explorer</i>	12
<i>Using the Navigation Panel</i>	13
<i>Working in the Integrated Design Window</i>	14
<i>What is a Design Database?</i>	17
<i>Creating a New Design Database</i>	18
<i>Selecting the Storage Type</i>	18
<i>Password-Protecting a Design Database</i>	19
<i>Creating a new Document in the Design Database</i>	20
<i>Editing a Document</i>	21
<i>Importing an External Document into the Design Database</i>	22
<i>Linking to an External Document</i>	22
<i>Exporting Documents from the Design Database</i>	23
<i>Managing Documents in the Design Database</i>	24
<i>Using Folders to Create Hierarchy</i>	24
<i>Moving, Copying and Deleting Documents</i>	24

Contents

<i>Renaming a Document or Folder</i>	24
<i>Setting up a shared Database for your Design Team</i>	25
<i>Creating Team Members</i>	25
<i>Defining Team Member Permissions</i>	26
<i>Monitoring what Documents are Open</i>	27
<i>Locking a Document</i>	28
<i>Working with your Existing Protel Designs</i>	29
<i>Converting an Existing Design into a Design Database</i>	29
<i>Linking to External Documents</i>	30
<i>Working with Existing Company Libraries</i>	30
<i>Synchronizing an Existing Schematic Project and the PCB</i>	31
<i>The Technology inside Protel 99 SE</i>	32
<i>SmartTool Technology</i>	32
<i>SmartDoc Technology</i>	34
<i>SmartTeam Technology</i>	35
<i>Customizing the Design Explorer</i>	36
<i>Resources – Menus, Toolbars and Shortcut Keys</i>	36
<i>Understanding Processes</i>	43
<i>Working with Servers</i>	45
<i>The Text Editor</i>	48
<i>Languages</i>	48
<i>Syntax Highlighting</i>	49
<i>Document Options</i>	49
<i>Macros</i>	50
 SECTION 3	 51
 SCHEMATIC CAPTURE	 51
<i>Schematic Capture – Feature Highlights</i>	53
<i>Schematic Sheet Editor</i>	54
<i>Schematic Library Editor</i>	54
<i>Design Capabilities</i>	54
<i>Fundamentals of Schematic Capture</i>	57
<i>The Computer Model of a Circuit</i>	57
<i>The Schematic Component Model</i>	58
<i>Connectivity</i>	59
<i>Using Connectivity</i>	60
<i>Design verification</i>	62
<i>Linking from the Schematic to the PCB Layout</i>	63
<i>Setting up the Schematic Workspace</i>	64

<i>Environment Preferences</i>	64
<i>Schematic Document Options</i>	66
<i>Sheet Templates</i>	68
<i>Working in the Schematic Editing Window</i>	72
<i>Changing Your View of the Sheet</i>	72
<i>Moving Around the Schematic</i>	72
<i>Placing Schematic Objects</i>	75
<i>Editing Schematic Objects</i>	78
<i>Global Editing</i>	85
<i>Quick-Copying an Object</i>	88
<i>Activity - Finding and Replacing Text</i>	89
<i>Aligning Objects</i>	91
<i>Moving and Dragging</i>	92
<i>Schematic Editing Shortcuts</i>	95
<i>Re-entrant Editing</i>	95
<i>Canceling a Screen Redraw</i>	95
<i>Mouse Shortcuts</i>	95
<i>Keyboard Shortcuts</i>	96
<i>Frequently Used Shortcut Keys</i>	98
<i>Undo and Redo</i>	98
<i>Schematic Design Objects</i>	99
<i>Electrical Schematic Primitives</i>	100
<i>Non-Electrical (Drawing) Primitives</i>	102
<i>Fonts</i>	105
<i>Schematic Components and Libraries</i>	106
<i>What is a Schematic Library?</i>	106
<i>What is a Component and what is a Part?</i>	106
<i>Where are the Schematic Component Libraries?</i>	107
<i>Accessing the Components You Need for Your Design</i>	107
<i>Placing Parts on the Schematic Sheet</i>	110
<i>The Attributes of a Schematic Part</i>	111
<i>The Schematic Library Editor</i>	114
<i>Multi-Sheet Design and Project Management</i>	122
<i>Overview</i>	122
<i>Managing Multiple Sheet Projects</i>	123
<i>Structure of a Multi Sheet Schematic</i>	123
<i>Master Sheets and Sub-Sheets</i>	124
<i>How Connectivity is Created in a Multi-Sheet Design</i>	125
<i>The Different Methods of Structuring a Multi-Sheet Design</i>	127
<i>Working With a Hierarchical Project</i>	134
<i>Schematic Design Verification</i>	136
<i>Marking Points That You Do Not Want Flagged as Errors</i>	136

Contents

<i>Verification Options</i>	137
<i>Setting the Net Identifier Scope</i>	138
<i>Setting up the Electrical Rules Matrix</i>	138
<i>Resolving Errors</i>	140
<i>Preparing the Design for PCB Layout</i>	142
<i>Assigning and Re-assigning Designators</i>	142
<i>Check for Missing Footprints</i>	144
<i>Perform an ERC</i>	144
<i>Including PCB Layout Specifications</i>	145
<i>Ready for PCB Layout</i>	145
<i>Transferring the Design Information to the PCB</i>	146
<i>Transferring the Design Information</i>	146
<i>How the Synchronizer knows which PCB to Use</i>	149
<i>Where the Components are Placed in the PCB Workspace</i>	149
<i>Transferring PCB Layout Information</i>	150
<i>Passing Forward Schematic Design Updates</i>	150
<i>How the Synchronizer Associates the Schematic and PCB Components</i>	150
<i>How the Synchronizer Transfers the Design Information</i>	152
<i>Printing Your Schematic Design</i>	154
<i>Overview</i>	154
<i>Generating a print or plot</i>	154
<i>Creating Schematic Reports</i>	158
<i>Bill of Materials</i>	158
<i>Cross Reference</i>	158
<i>Project hierarchy</i>	158
<i>Netlist Compare</i>	158
<i>Linking to Databases</i>	159
<i>Hot Linking to a Database</i>	159
<i>Importing and Exporting to a Database</i>	163
<i>Interfacing to Third-Party Tools</i>	167
<i>Translating an Orcad Capture® Design</i>	167
<i>Mechanical CAD Interface</i>	169
<i>Creating a Netlist</i>	169
 SECTION 4	 175
 MIXED-SIGNAL SIMULATION	 175
<i>Mixed-Signal Simulation – Feature Highlights</i>	177
<i>Getting Started with Simulation</i>	180
<i>Example Circuits</i>	180

<i>Creating the Circuit</i>	180
<i>Adding the Source Components</i>	181
<i>Defining the Points to be Plotted</i>	181
<i>Setting up the Analyses</i>	182
<i>Running a Simulation</i>	182
<i>Viewing the Simulation Results</i>	182
<i>Setting up and Running a Simulation</i>	183
<i>Specifying the Simulation Data that you want Collected, Displayed and Stored</i>	184
<i>Transient Analysis</i>	185
<i>AC Small Signal Analysis (AC Sweep)</i>	188
<i>DC Sweep Analysis</i>	191
<i>DC Operating Point Analysis</i>	193
<i>Monte Carlo Analysis</i>	194
<i>Parameter Sweep</i>	198
<i>Temperature Sweep</i>	201
<i>Fourier Analysis</i>	202
<i>Transfer Function Analysis</i>	203
<i>Noise Analysis</i>	204
<i>Impedance Plot Analysis</i>	205
<i>Mathematical Functions and Waveforms</i>	206
<i>Including extra SPICE Information in the Netlist</i>	209
<i>Creating and Simulating from the Netlist</i>	209
<i>Specifying Initial Circuit Conditions</i>	210
<i>The Waveform Analysis Window</i>	211
<i>Displaying Waveforms</i>	211
<i>Resizing the Waveforms</i>	212
<i>Viewing Waveforms in Separate Cells</i>	215
<i>Displaying Multiple Waveforms in the same Cell</i>	215
<i>Displaying a Waveform Scaled Two Ways</i>	216
<i>Scaling the Waveforms Axes</i>	217
<i>Displaying the Simulation Data Points</i>	217
<i>Identifying Waveforms on a Single Color Printout</i>	217
<i>Using the Measurement Cursors</i>	218
<i>How Multi-pass Results are Displayed</i>	218
<i>Voltage and Current Sources</i>	219
<i>Locating the Source Components</i>	219
<i>Constant Source</i>	219
<i>Sinusoidal Source</i>	220
<i>Periodic Pulse Source</i>	222
<i>Piece-Wise-Linear Source</i>	224
<i>Exponential Source</i>	226
<i>Frequency Modulation Source</i>	228

Contents

Linear Dependent Sources	230
Non-linear Dependent Sources	233
Components and Models	235
Component Symbol Libraries	235
Models and Subcircuits	236
Device Descriptions	237
Creating your own Simulation-Ready Components	247
Working with Circuits that will not Simulate	252
When The Netlist can not be Created	252
Understanding SPICE Convergence	252
Strategies for Solving Convergence Failures	253
Solving DC Analysis Failures	254
Solving Transient Analysis Failures	255
Warning and Error Messages	255
SPICE Variables and Analog Options	256
Integration Method	256
Digital Power Supply and the Reference Ground	257
SPICE Options	257
Suggested SPICE Reading	261
Simulating Digital Designs	262
Modeling Digital Components	262
Digital Power and Ground	262
Creating New SimCode Devices	263
SimCode Language Definition	269
SimCode Language Syntax	272
 SECTION 5	 311
 PLD DESIGN	 311
PLD Design – Feature Highlights	313
Schematic-based PLD Design	314
Designing a PLD with CUPL	314
The PLD Compiler	314
The PLD Simulator	314
Viewing the Simulation Waveforms	314
CUPL High-level Language	315
Universal Device Support	315
An Overview of the PLD Design Process	316
Entering the Design	316
Compiling the Design	317
Simulating the Design	318

<i>Where to Set-up and Run the PLD Compiler</i>	318
<i>Schematic-based PLD Design Entry</i>	319
<i>Using the Symbol Library</i>	319
<i>Designing the Internal Logic</i>	333
<i>Creating a Multi-Sheet Design</i>	334
<i>Interfacing from Internal Logic to Component Pins</i>	334
<i>Compiling a Schematic-based Programmable Logic Design</i>	335
<i>CUPL HDL Design Entry</i>	336
<i>Overview of CUPL Source File Syntax</i>	336
<i>Source File Directives</i>	338
<i>Language Elements</i>	342
<i>Language Syntax</i>	344
<i>Compiling the Programmable Logic Design</i>	351
<i>Selecting the Target Device</i>	351
<i>Compiler Options</i>	354
<i>Output Formats</i>	357
<i>Simulating the Programmable Logic Design</i>	359
<i>Input to the Simulator</i>	359
<i>Output from the Simulator</i>	359
<i>Creating a Test Specification Source File</i>	360
<i>Header Information</i>	360
<i>Specifying a Device</i>	360
<i>Comments</i>	361
<i>Statements</i>	361
<i>Variable Declaration (VAR)</i>	372
<i>Simulator Directives</i>	372
<i>Advanced Syntax</i>	375
<i>Looping Constructs</i>	378
<i>Virtual Simulation</i>	385
<i>Fault Simulation</i>	385
<i>Viewing the Simulation Waveforms</i>	386
<i>Sample PLD Design Session</i>	389
<i>The Design Steps</i>	389
<i>Step 1 - Examining the Design Task</i>	390
<i>Step 2 - Creating the CUPL Source Code</i>	391
<i>Step 3 - Formulating the Equations</i>	392
<i>Step 4 - Choosing a Target Device</i>	394
<i>Step 5 - Making the Pin Assignments</i>	395
<i>Step 6 - Compiling the PLD Source File</i>	397
<i>Step 7 - Creating a Schematic-based version of the Design</i>	399
<i>Step 8 - Creating a Simulation Test Vector File</i>	403
<i>Step 9 - Simulating the Device</i>	407
<i>Step 10 - Viewing the Simulation Waveforms</i>	410

<i>Summary</i>	410
<i>PLD Design Examples</i>	411
<i>Example 1 - Simple Logic Gates</i>	412
<i>Example 2 - Two-Bit Counter</i>	418
<i>Example 3 - Simple State Machine Design</i>	420
<i>Example 4 - Decade Up/Down Counter</i>	422
<i>Example 5 - Seven-Segment Display Decoder</i>	430
<i>Example 6 - 4-Bit Counter with Load and Reset</i>	433
 SECTION 6	 437
 PCB DESIGN	 437
<i>PCB Design – Feature Highlights</i>	439
<i>The PCB Layout Editor</i>	440
<i>The PCB Library Editor</i>	440
<i>Shape-Based Autorouting</i>	440
<i>Design Capabilities</i>	440
<i>Setting up the PCB Workspace</i>	446
<i>Coordinate System</i>	446
<i>Workspace Size and Accuracy</i>	446
<i>Switching from Imperial to Metric</i>	446
<i>Grids</i>	447
<i>Layers</i>	449
<i>PCB Workspace Preferences</i>	453
<i>Creating, Opening and Saving PCB Documents</i>	460
<i>Creating a New PCB</i>	460
<i>Save Copy As Option</i>	460
<i>Importing and Exporting older Protel PCB File Formats</i>	460
<i>Working in the PCB Design Window</i>	461
<i>Changing Your View of the Workspace</i>	461
<i>Moving Around the PCB Workspace</i>	463
<i>Editing PCB Objects</i>	466
<i>Selection</i>	469
<i>Global Editing</i>	478
<i>Moving and Dragging</i>	483
<i>Deleting</i>	485
<i>Measuring Distance in the PCB Workspace</i>	486
<i>PCB Editing Shortcuts</i>	487
<i>Re-entrant Editing</i>	487
<i>Canceling a Screen Redraw</i>	487
<i>Mouse Shortcuts</i>	487

<i>Slider Hand</i>	488
<i>Keyboard Shortcuts</i>	489
<i>Special Mode-Dependent Keys</i>	491
<i>Locating Components</i>	491
<i>Undo and Redo</i>	491
<i>PCB Design Objects</i>	492
<i>PCB Primitive Objects</i>	493
<i>Group Objects</i>	504
<i>PCB Component Footprints and Libraries</i>	509
<i>What is a Footprint and what is a Component?</i>	509
<i>Where are the PCB Footprint Libraries?</i>	509
<i>Accessing Component Footprints</i>	509
<i>Finding and Placing Components</i>	511
<i>PCB Components Attributes</i>	512
<i>Changing the Footprint a Component is Using</i>	515
<i>Modifying a Component Footprint on the Board</i>	515
<i>Including Routing in Component Footprints</i>	516
<i>Un-Grouping a Component</i>	516
<i>Copying Components from the PCB to a Library</i>	516
<i>The PCB Library Editor</i>	517
<i>Creating a Project Library</i>	520
<i>Defining the Board</i>	521
<i>Creating the mechanical definition of the PCB</i>	521
<i>Defining the Placement and Routing Outline</i>	522
<i>Defining the PCB Layer Stack</i>	522
<i>Defining the drill pairs</i>	524
<i>Using The Board Wizard</i>	524
<i>Transferring the Design from the Schematic</i>	525
<i>Specifying the PCB Design Requirements</i>	526
<i>What are Design Rules?</i>	526
<i>Defining the Design Rules</i>	527
<i>How Rules are Applied</i>	534
<i>Working with Design Rules</i>	535
<i>Object Classes</i>	537
<i>Routing Rule Definitions</i>	539
<i>Manufacturing Rule Definitions</i>	544
<i>High Speed Rule Definitions</i>	548
<i>Placement Rule Definitions</i>	550
<i>Signal Integrity Rule Definitions</i>	552
<i>Other Rule Definitions</i>	556
<i>Examples of Using Design Rules</i>	557
<i>Component Placement Tools and Techniques</i>	563
<i>Important Placement Options</i>	563

Contents

<i>Moving Components</i>	564
<i>Working Between the Schematic and PCB</i>	566
<i>Using Component Unions</i>	567
<i>Working with Placement Rooms</i>	568
<i>Using the Interactive Placement Tools</i>	569
<i>Automatic Component Placement</i>	571
<i>Understanding Connectivity and Topology</i>	576
<i>How the Component and Connectivity Information is Transferred</i>	576
<i>How the Net Connectivity is Displayed</i>	577
<i>Net Topology</i>	578
<i>Changing the Net Topology</i>	578
<i>User-defined From-Tos</i>	578
<i>Displaying Pin-to-Pin Connections</i>	581
<i>Managing the Netlist</i>	581
<i>Changing Net Attributes</i>	582
<i>Identifying Nets</i>	582
<i>Manually Routing the PCB</i>	583
<i>How the PCB Editor Manages the Connectivity as you Route</i>	584
<i>Preparing to Route</i>	586
<i>Manually Routing the PCB</i>	588
<i>Interactive Routing Modes, including Push and Shove</i>	592
<i>Re-routing</i>	593
<i>Using Internal Power Planes</i>	594
<i>Creating a Copper Plane on a Signal Layer</i>	598
<i>Autorouting the PCB</i>	599
<i>Setting Up to Autoroute</i>	599
<i>Autorouting Options</i>	602
<i>Inside Protel 99 SE's Autorouter</i>	603
<i>Including Testpoints and Teardrops on the PCB</i>	605
<i>What is a Testpoint?</i>	605
<i>Setting up the Testpoint Design Rules</i>	606
<i>Locating Existing Testpoint Sites</i>	607
<i>Adding Testpoints with the Autorouter</i>	607
<i>Reporting the Location of Each Testpoint</i>	608
<i>Reporting Nets that Failed to Receive a Testpoint</i>	608
<i>Clearing all Testpoints from the Board</i>	608
<i>Adding Teardrops to Pads and Vias</i>	608
<i>Verifying the PCB Design</i>	609
<i>What is the Design Rule Checker?</i>	609
<i>Online DRC</i>	609
<i>Setting Up for a Batch Mode Design Rule Check</i>	610
<i>Running the Batch DRC</i>	611
<i>The DRC Report</i>	611

<i>Resolving Design Rule Violations</i>	611
<i>Checking the Signal Integrity</i>	613
<i>Generating Reports</i>	617
<i>3 Dimensional PCB Visualization</i>	618
<i>Printing to a Windows Printing Device</i>	620
<i>Setting up a Print Preview</i>	621
<i>What is a Printout?</i>	621
<i>Changing the set of Printouts</i>	622
<i>Changing the Target Printer</i>	623
<i>Setting the Paper Orientation, Scaling and other Printer Options</i>	623
<i>Printing</i>	624
<i>Copying from the Preview Window to other Applications</i>	624
<i>Exporting the Printouts as WMF files</i>	624
<i>Color, Font and other Preferences</i>	625
<i>Generating the Manufacturing Files</i>	626
<i>Working With the PCB Manufacturer</i>	626
<i>Generating Manufacturing Output with the CAM Manager</i>	626
<i>Creating a new CAM Document</i>	627
<i>Adding Output Setups to a CAM Document</i>	627
<i>Configuring the CAM Generation Options</i>	628
<i>Generating the Output Files</i>	629
<i>Gerber File Output Setup</i>	629
<i>NC Drill Output Setup</i>	632
<i>Bill of Materials Output Setup</i>	633
<i>Pick and Place Output Setup</i>	633
<i>Testpoint Report Output Setup</i>	634
<i>Design Rule Check Output Setup</i>	635
<i>Transferring the CAM Document to Another Design</i>	636
<i>What is Artwork?</i>	636
<i>Passing Design Changes Back to the Schematic</i>	642
<i>Re-assigning the Component Designators</i>	642
<i>Updating the Schematic from the PCB</i>	642
<i>Interfacing to Third-Party Tools</i>	644
<i>Loading a Netlist</i>	644
<i>Exporting the Netlist</i>	649
<i>File Importers and Exporters</i>	649



Section 1 —

Setting up and getting started

Printed Circuit Board Design with Protel 99 SE

3

Installation

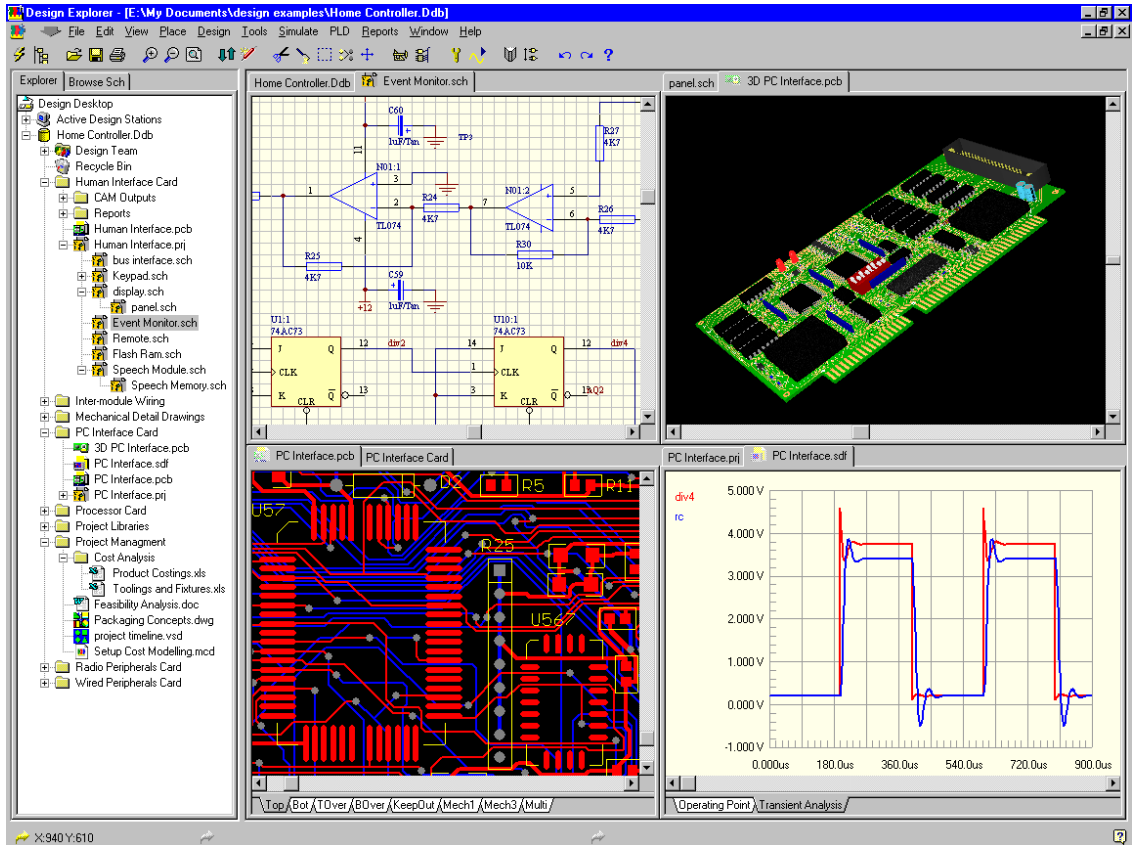
6

Document Conventions

8

Setting Up and Getting Started

Printed Circuit Board Design with Protel 99 SE



Welcome to the Protel 99 SE Designer's Handbook, a comprehensive guide to learning and using Protel 99 SE. Whether you are new to the world of EDA design or a seasoned veteran, this book will help you get the most out of this powerful design environment by providing full instructions on all Protel 99 SE's features, as well as tips and hints on streamlining the design process and getting the most from this unique design environment.

◆ Refer to the Readme file for any late-breaking updates and additions.

Protel 99 SE transforms your desktop PC into a complete, integrated, full 32-bit printed circuit board design and document management environment, providing everything you need to take your electronics project from concept, all the way through to completed boards.

Capture your circuit as a "connectivity-aware" schematic, choosing components from a vast array of libraries. Perform circuit simulations directly from your schematic at the touch of a button. Port elements of your design for incorporation into programmable logic devices. Layout your schematic as a printed circuit board (PCB), maintaining electrical connectivity and design rule compliance throughout. Autoroute your PCB to produce a professional single or multi-layer board that complies with your manufacturer's mechanical layout specifications, then test the signal integrity before the design leaves your PC. Protel 99 SE lets you perform all these tasks easily from within a single, integrated design environment.

The heart of Protel 99 SE is its unique architecture, built on *Smart* integration technologies – *SmartTool*, *SmartDoc*, and *SmartTeam*.

SmartTool is client/server technology that brings all your favorite design tools together, into a single user interface. These tools include Protel 99 SE's Schematic Editor, Mixed-Signal Circuit Simulator, PLD Compiler, PCB Placement and Routing, and PCB Signal Integrity Analysis.

SmartTool technology also integrates OLE compliant editors. These include Microsoft Word®, Microsoft Excel®, and the Visio® tools, to name a few. Now you can store the handbook that you are writing in Word in the Design Database right next to the other design documents, then simply double-click to open in Word, ready for editing.

SmartDoc technology finally solves the task that every designer dreads – document management. No longer do you need to keep track of 30 or more design documents; from schematic sheets, simulation results, PLD source files, PCB layouts, Gerber and drill files, the bill of materials, various Word documents, mechanical drawings – the list goes on.

With *SmartDoc* technology *all* the design documents are stored in a single Design Database. You can import and export documents at any time – when it is time to send the files off to the PCB fabrication house just select them, right-click and Export.

SmartTeam technology brings a new way of working to the design team. With SmartTeam technology all the team members can work in the same Design Database, at the same time, with complete confidence. All the document access and control features that you need are there – create members, specify their access rights, keep track of which designer is working on what document, and lock documents to prevent accidental overwriting. Better still, all these features are defined and controlled at the Design Database level, not at the network level.

Protel 99 SE brings these three powerful technologies together in the *Design Explorer*. This single intuitive interface is all you need to manage and edit all the documents in the integrated Design Database. If you can use the Windows File Explorer then you

already know how to use the Design Explorer. Navigate and manage all the documents in the Design Database, create folders, drag and drop documents, import and export documents, all with a few mouse clicks.

You can automate tasks in Protel 99 SE using macro scripts written in Client Basic, a simple to learn yet powerful macro programming language. Develop your macro scripts using the Text Editor and you have full Client Basic syntax highlighting, and a range of debugging tools at your disposal. Macros operate at the environment level, meaning you can control any system process, regardless of which Editor provides it.

If you are new to the world of Protel 99 SE, please take some time to get to know the capabilities of the Protel 99 SE environment. All toolbars, menus and shortcut keys in Protel 99 SE are highly configurable. We understand that different designers like to work in different ways, and Protel 99 SE allows you to tailor your environment to suit.

Whenever you are working on a particular type of document, a schematic sheet or a PCB layout for example, Protel 99 SE will present you with the appropriate toolbars, menus and shortcut keys. Switch to a different type of document and Protel 99 SE will change the toolbars, menus and shortcut keys to those appropriate to the new document. You can easily configure the toolbars, menus and shortcut keys that are active for any particular document type. So regardless of whether you are routing a PCB, calculating manufacturing costs in a spreadsheet, or running a transient circuit analysis, you will always have the tools you want directly at hand.

Components are the foundation of any electronic circuit, and Protel 99 SE comes with comprehensive schematic, PCB footprint and simulation model libraries that include a vast range of components from all leading device manufacturers. The Altium Library Development Center is continually updating these libraries and creating new ones to give you access to the very latest devices and packaging. For instant component updates, the Protel Web site at www.protel.com maintains the latest libraries for direct download. With Protel 99 SE, you can also create your own component libraries. Component Wizards make the job of creating your own custom components a breeze.

At Altium, we understand that great designs are not created by computer, they are created by designers. That is why we have created Protel 99 SE to be the easiest to use, most integrated and configurable electronics design suite for the Windows platform. Protel 99 SE works the way you want to work.

So sit back, put your feet up and let Protel 99 SE go to work for you.

Installation

System Requirements

Minimum

- Microsoft Windows 95 or NT running on an IBM PC or compatible
- Pentium processor
- 32 MB of RAM
- SVGA display, 16 color (1024x768 resolution)
- 200 MB of hard disk space for minimum installation

Recommended

- Microsoft Windows NT 4 (or later)
- Pentium II processor, 300 MHz or faster
- 64 MB of RAM
- True color display (1024x768 resolution or higher)
- 300 MB of hard disk space for complete installation (all libraries)

What is Supplied With Your Protel Product?

Your Protel package includes the following materials:

- Installation CD
- Designer's Handbook (this book)
- Letter with the required access codes (this may be delivered separately)
- Protel Software Registration Card (unless purchased as an upgrade)
- License agreement (removable attachment on the Registration Card)

If any of these items are missing from your package, contact your Altium representative immediately to arrange a replacement.

Installing the Software

When you place the Protel 99 SE CD in your CD drive the installation shield will appear automatically. Follow the installation instructions from there.

To run the installation process manually select the Run option in the Start menu. In the Run dialog box, type in the following:

`<drive_name>:\Protel99SE\Setup.exe`

where <drive_name> is the drive you are installing from. This is typically D or E when installing from CD ROM. Follow the installation instructions from there.

The last page of the installation Wizard may report that your PC will be restarted to complete the installation process. You will not be able to run the software until this is done.

♦ It is advisable to close all applications and re-start Windows prior to installing the software.

Enabling the Software

If the Lock dialog appears when you try to edit a document (after installing the software), it means that the Access Code for that editor has not been entered correctly. You will not be able to use that editor until the correct code has been entered. After entering the code in the Lock dialog press the Test button to confirm that it has been entered correctly.

You can check the access status of each editor that requires an Access Code in the Security Locks dialog. Select **Client menu » Security** from the menus to display the dialog, select the Editor, and press the Unlock button. The Lock dialog will list each 16-character Access Code that has been entered, and display the current Access Rights.

Register Your Software

Sign and return the enclosed License Registration Card. By returning this card you acknowledge that you have accepted the terms of the license. You should also notify your Altium representative if your address changes. Maintaining your registration ensures on-going access to technical support, upgrade notices and other important product information.

Review the README Document

Please review the README document for product information that may have changed after this manual went to print.

Document Conventions

Windows	Refers to Microsoft Windows 95, 98, or NT4.
DOS	Refers to MS-DOS® or PC-DOS™ version 5.0 or later.
<i>italic</i>	Indicates anything to be typed. Always enter the italicized information exactly as it appears. Italics are also used to flag chapter and topic names, and to add emphasis to an important point.
CAPITALS	These are used to indicate directory or filename.
SMALL CAPS	These are used to indicate key names, such as ENTER or ESC.
Initial Caps	These indicate dialog box names (e.g. Document Options) or option names in a dialog box (Snap To Center).
menu » menu item	Menu items are shown in bold. Each menu level is separated by a ». For example, if you read Client menu » Servers you should click on the Client menu (the down arrow on the left of the menu bar), then click on Servers.
SHIFT+ALT	The + sign means: hold down the SHIFT key then press the ALT key.
F1, F2	The comma (,) means press and release the F1 key then press and release the F2 key.

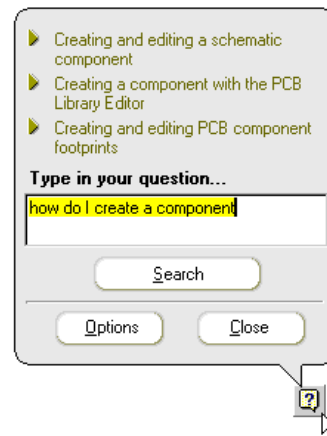
◆ Tips are shown in a highlight box like this.

On-line Help

The Help menu provides instant access to on-line information about the various features of the Protel software. Protel 99 SE includes an easy-to-use natural language query feature, the Help Advisor. To run the Help Advisor click on the button on the Status bar, or select **Help » Search for Help on** from the menus.

Simply type a question in the Help Advisor as if you were asking a person. A list of appropriate topics will appear, click on a topic to launch the help system and display that topic. You can also tailor the Natural Language interface to search particular topics, press the Options button to configure this.

- Press F1 when a menu item is highlighted to pop up a description of the process that is launched when you select that menu item.



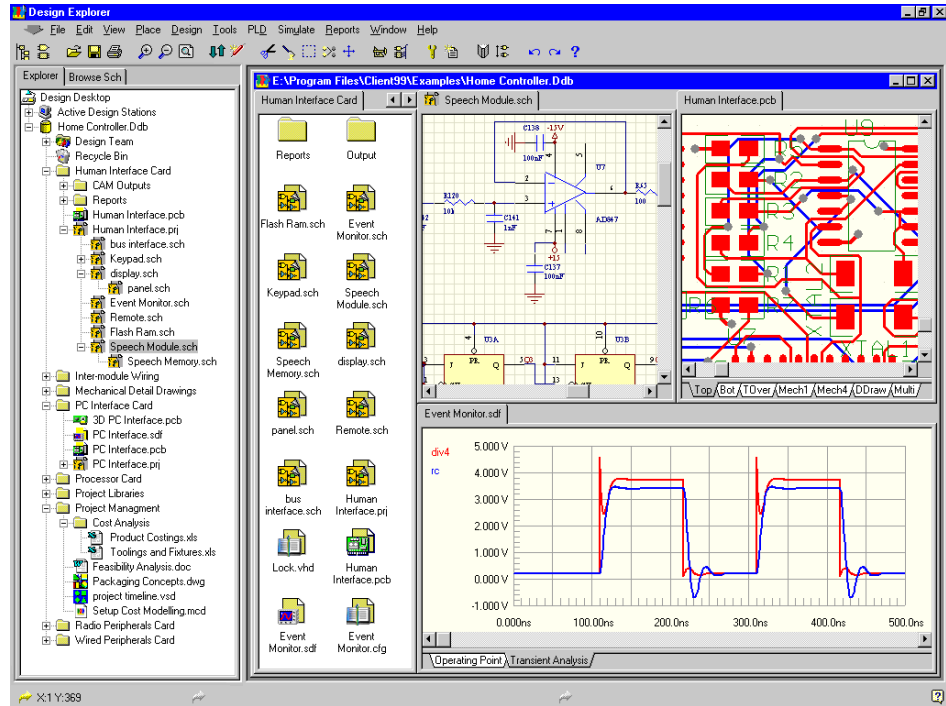
Click on the Help Advisor button on the Status bar to popup the Help Advisor



Section 2 ——— **The Design Explorer**

<i>The Protel99 Design Explorer</i>	11
<i>How to work in the Design Explorer</i>	12
<i>What is a Design Database?</i>	17
<i>Creating a New Design Database</i>	18
<i>Creating a new Document in the Design Database</i>	20
<i>Editing a Document</i>	21
<i>Importing an External Document</i>	22
<i>Linking to an External Document</i>	22
<i>Exporting Documents from the Design Database</i>	23
<i>Managing Documents in the Design Database</i>	24
<i>Setting up a shared Database for your Design Team</i>	25
<i>Working with your Existing Protel Designs</i>	29
<i>The Technology inside Protel99</i>	32
<i>Customizing the Design Explorer</i>	36
<i>The Text Editor</i>	48
<i>Macros</i>	50

The Protel 99 SE Design Explorer



The Design Explorer is your interface to your Design Database

When you select Protel 99 SE from the Windows Start menu, the *Design Explorer* opens. The Design Explorer is your interface to your designs, and the various design tools that you use to create your designs.

The Design Explorer has a number of features which distinguish it from other Windows applications. These include:

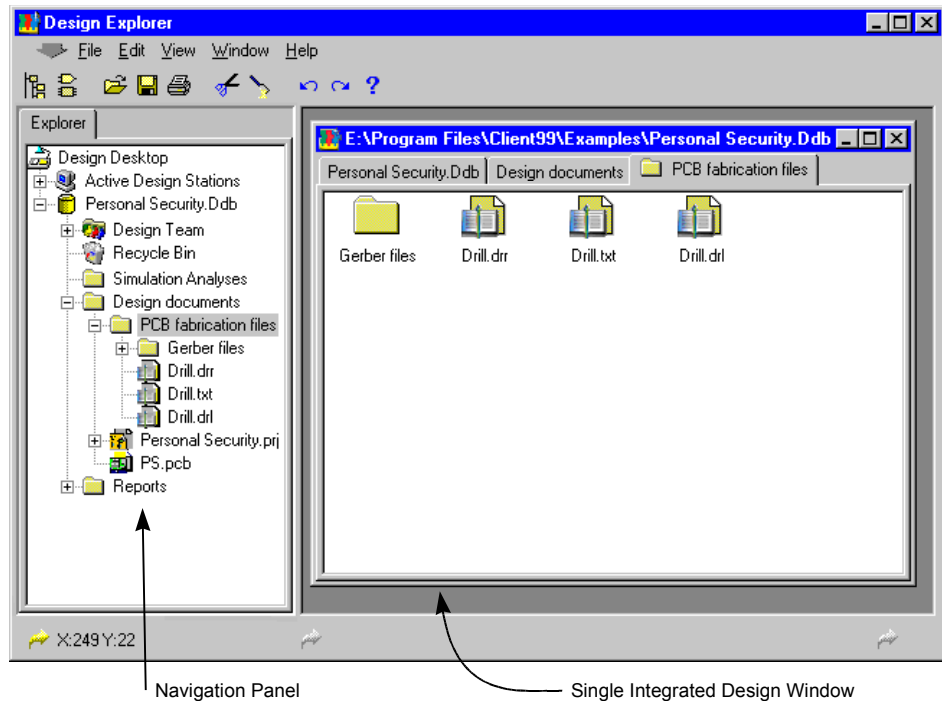
- The ability to store multiple documents (files) in a single *Design Database*. These can be Protel documents, such as schematic sheets, PCB files, etc, as well as any other kind of document created by any application in Microsoft Windows.
- A single document editing window for each open Design Database, referred to in this handbook as a *Design Window*. Each document that you open from a *Design Database* is opened on a separate Tab, within the same *Design Window*. This allows you to easily manage anything from a single schematic sheet design, through to a large project which includes multiple sets of schematics and PCBs, as

The Design Explorer

well as other project documents, such as Microsoft Word and Excel documents, Visio documents, and so on.

- The *Design Explorer Navigation Panel* which you use to manage your design – in it you can create a design hierarchy of any depth, navigate the design hierarchy, and perform all the standard document operations, such as copy, paste, move and delete.

How to work in the Design Explorer



Working in the Design Explorer is easy. In fact if you are familiar with the Windows File Explorer, then you are ready to go!

Like the File Explorer, there are two regions to work in: the navigation tree in the panel on the left; and the view of where you currently are in the tree, on the right. In the Design Explorer this is your window into the open Design Database, and is called the *Design Window*.

Using the Navigation Panel

The Navigation Panel on the left shows the tree-like relationship between all the documents in your design. Like the Windows File Explorer, it shows how all the various design documents are stored in the Design Database.

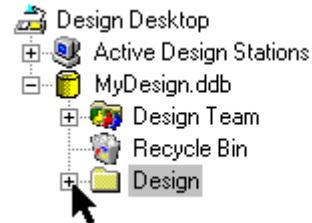
The Navigation Panel also shows any logical relationship between the schematic sheets in each schematic project.

In the Navigation Panel you can:

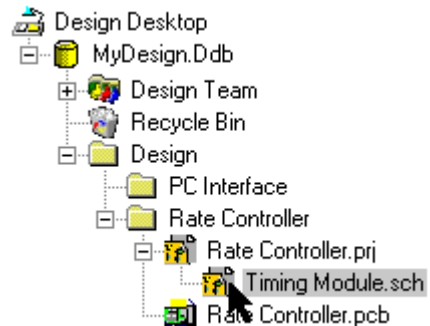
- Click once on the small “+” symbol to expand that branch of the tree.
- Click once on the small “-” symbol to collapse that branch of the tree.
- Double-click on a folder to expand the tree, as well as open that folder in the Design Window.
- Click once on a document to open the document in the Design Window.
- Click and hold on a document or folder, then drag and drop it on another folder to move it. If you do this with a folder all the contents are moved too.
- Right mouse click and hold on a document or folder, then drag and drop it on another folder. When you release the mouse a small menu will appear, where you can select Move, Copy or Create Shortcut.



click once on the + symbol to expand the tree



continue to click on each + to expand the tree



click once on the document to open it

- ◆ The best way to navigate through the Design Database is to use the Navigation Panel.
- ◆ By clicking on the small + and – symbols in the design tree you can explore the various branches of the tree, without actually having to open a folder or document. When you locate the document you wish to work on, simply click once on the document icon or name to open it, ready for editing.

Working in the Integrated Design Window

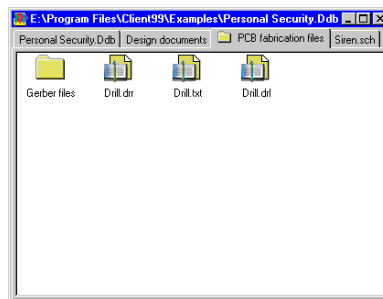
The Design Window has two different types of views – *folder views*, which display all the documents and sub-folders within that folder; and *editor views*, which display the document that is open for editing.

You can have multiple views of each type open at the same time. The name of each document or folder that is currently open in memory is displayed on a tab along the top of the Design Window. Click on a Tab to make that folder or document active.

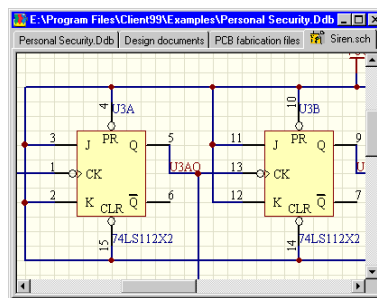
The *folder view* in the Design Window behaves just like the Windows File Explorer. In the folder view you can:

- Double-click on an icon to open that file or folder.
- SHIFT+click, CTRL+click, and click-and-drag-a-box around, to select documents and folders.
- Click and drag to move documents and folders from one folder view to another (even to another Design Database).
- Right-click and drag on documents and folders – when you drop them on the destination folder a small menu will appear, asking if you would like to Move, Copy, or Create a Shortcut to the objects you dragged.
- Right-click on an icon to display a file/folder manipulation menu, where you can; export the document, copy or cut the current (or selected) document, or import an object (a non-Protel document).
- Right-click elsewhere in the folder view to display a different menu, where you can; create new documents or folders, import documents, link to external documents, paste from the clipboard, or change the view.

Various types of document *editing views* are available. Each different type of document is edited in a different editing view. For example, schematic documents are edited in a Schematic Editor view, PCBs are edited in a PCB Editor view, ASCII reports are edited in a Text Editor view, and so on. You will notice that as you click on the Tabs at the top of the Design Window to move from one editor view to another, the menus and toolbars change.



A Folder view in the Design Window



An Editor view in the Design Window

Working with Multiple Open Documents

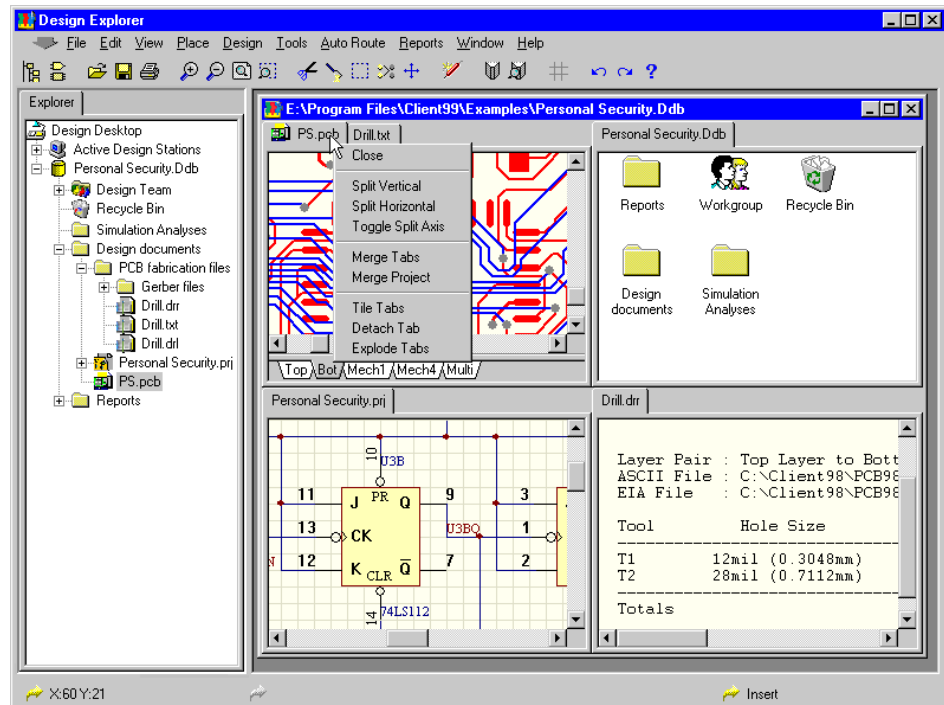
A single Design Window is convenient because it binds all the open design documents together. You can easily move between open documents, by clicking on the appropriate Tab at the top of the Design Window, or by using the CTRL+TAB and CTRL+SHIFT+TAB shortcuts.

Closing an Open Document

To close the active document you can either; right mouse click on the document Tab, then select **Close** from the menu that appears, or select **File » Close** from the menu bar.

Splitting the Design Window

When you need to view two or more documents at the same time, you can split the single window into multiple regions.



Right-click on the Tab to display the Design Window splitting options

The easiest way to split the Design Window is to right mouse click on the active Tab. The active Tab has a small icon on the left.

In the menu that appears there are a number of Split options. In the figure shown above the Tile Tabs option has been chosen, splitting the Design Window into four regions, one for each Tab that was open when the option was chosen.

Rearranging Tabs in a Split Design Window

The Tabs in a split Design Window can be rearranged by clicking, holding, and dragging a Tab from the current split region, and dropping it onto another Tab, in a different split region. With a bit of practice you will find it easy to quickly rearrange the Tabs, to display exactly as you require.

♦ Look for the icon to see which Tab is currently active.



Resizing Regions in a Split Design Window

Position the cursor over the edge where two regions of a split window meet. When you do small double-headed arrows will appear – click and drag to resize these two regions.

♦ When you position the mouse where two regions of a split window meet, the cursor changes to a resize cursor.



Restoring a Split Design Window to display as a Single View

If you have split the Design Window into a number of regions you can quickly restore the Design Window to a single view. To do this right-click on one of the Tabs and select Merge Tabs from the menu.

Detaching a Tab into a separate Window

You can also detach a Tab from a Design Window to display it in a separate window. To detach the active Tab, right-click on it to display the floating menu, and select Detach Tab. This Tab will appear in a separate window frame.

Restoring detached Tabs back into the Original Design Window

To return all detached Tabs back into the original Design Window, right-click on a Tab in the original window and select Merge Project.

What is a Design Database?

As you have read this chapter you would have noticed the term *Design Database* – you are probably wondering what exactly *is* a Design Database?

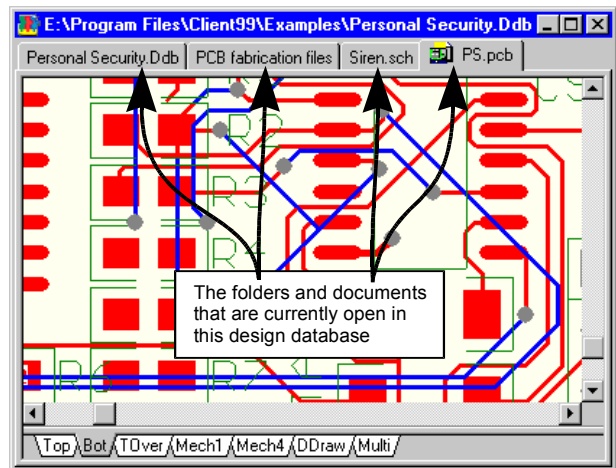
The dictionary definition of a database is, “a comprehensive collection of related data organized for convenient access”, and that is exactly what a Protel 99 SE Design Database is – it is all your design documents, organized and stored in a single file on your hard disk, for convenient access and management. A Protel 99 SE Design Database has the file extension *.Ddb*.

All the documents that have been created in the design are stored in this database, such as the schematic sheets, PCB, BOM, DRC report, Gerber files, and so on. The database also stores the folders, as well as information about the design hierarchy.

When you first open the Design Database only the top-level folder is opened in the Design Window, none of the documents inside the Design Database are opened. This means that even the largest Design Database, containing thousands of documents, opens very quickly.

You can then selectively open the various documents in the Design Database. When you want to open a document simply click once on the document icon in the tree, or double-click on the document icon shown in the folder view in the Design Window. The document will open in a new editor view in the Design Window.

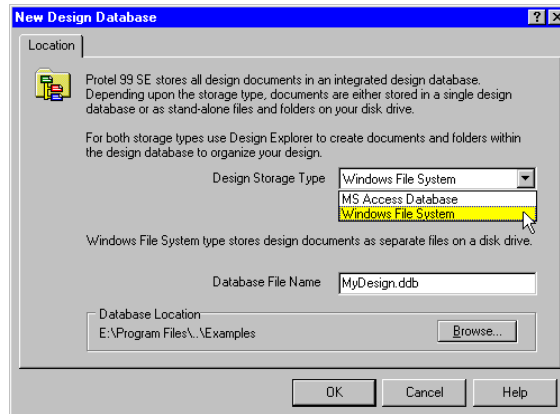
To check what documents are currently open look at the Tabs displayed at the top of the Design Window.



The Tabs at the top of the Design Window show you what documents and folders are currently open

Creating a New Design Database

To create a new Design Database select **File » New Design** from the menus. The New Design Database dialog will pop up. Set the Storage Type, enter the Database name, and change the location if required.



Creating a New Design Database – select the storage type, enter the name and change the location if necessary

Selecting the Storage Type

Design documents can be stored in 2 different ways:

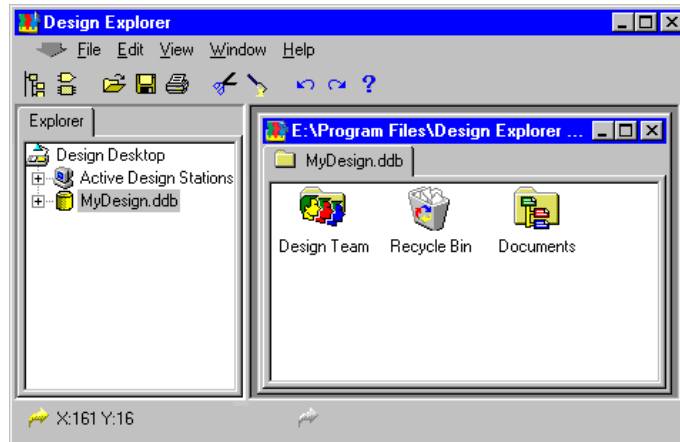
- If the Storage Type is set to MS Access Database then all design documents are stored inside a single Microsoft Access® database file.
- If the Storage Type is set to Windows File System then all design documents are stored directly on the disk drive in the location specified at the bottom of the dialog.

Regardless of the storage system that is chosen, the way you work in the Design Explorer is exactly the same. If your design uses the Windows File System Storage Type you still open the design first – then open the schematic, PCB, or other design documents. You can not move documents into a Windows File System database with the Windows File Explorer, they must be imported into a database before they can be opened. You can import a number of files by importing a folder, or by dragging from the Windows File Explorer directly into an open design.

Note that designs which use the Windows File System storage do not support any of the DesignTeam features, such as document access control. Other design integration features, like synchronization and background document opening when printing and netlisting, are supported.

Password-Protecting a Design Database

If you would like to password-protect the Design Database as you create it, click on the Password Tab and enter the password. You can password-protect a Design Database at any time, by going to the Members folder and entering a password for the Admin member. To unprotect a Database remove the password from the Admin member. Password-protection is not available if the Design Database uses the Windows File System storage type.



The new Design Database, ready to create new folders and documents in

Creating a new Document in the Design Database

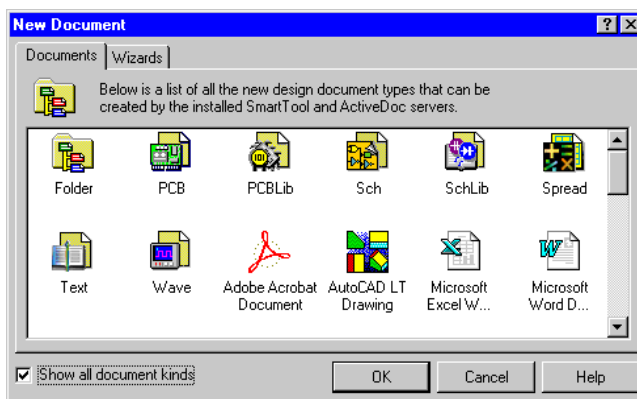
◆ Before you create a new document, first navigate to the folder where you want the document to be stored.

To create a new document in the Design Database right-mouse click in the folder view to display the floating menu, then click on **New** in the menu.

The New Document dialog will appear. The dialog includes an icon for each Document kind that can be created. Select the required document kind and click OK.

The kinds of documents that can be created depends on the available document editors. Enable the Show All Document Kinds option for a complete list.

An icon for a new document will appear in the design. To rename the document click once on the document icon to select it (in the folder view, not the tree), then select **Edit » Rename** from the menus (or press the F2 shortcut). The text will appear highlighted, ready to type in the new name.



Why are there non-Protel Icons in the New Document Dialog?

One of the powerful features of Protel 99 SE is its ability to support both Protel documents, and non-Protel documents. Any document that is created and edited by an OLE Server (currently registered on your PC) will appear in the New Document dialog, when you enable the Show All Document Kinds option.

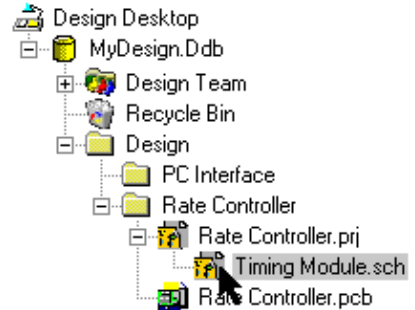
These non-Protel documents are stored inside the Design Database, but can still be edited by the application (OLE Server) that created them.

This powerful feature allows you to store all the documents associated with a project inside the Design Database. By storing these documents inside the Design Database you get all the benefits of a single, shared Design Database – all documents kept together, database and document access control, document monitoring and document locking. Refer to the topic, *Setting up a Shared Database for your Design Team*, later in this chapter for more details on these features.

Editing a Document

When you first open a Design Database none of the documents in the database are opened. The only thing that is opened is the top level folder. There are two approaches to opening a document in the design database for editing.

The first approach is to use the Navigation Panel on the left to locate the document in the Design Database, then click once on its icon to open the document. The advantage of this approach is that you can locate and open the document (regardless of how deep it is in the Design Database), without opening anything else. For tips on using the Navigation Panel refer to the topic, *Using the Navigation Panel* earlier in this chapter.



Use the Navigation Panel to quickly locate and open the document

The Alternative approach is to double-click on the appropriate sub-folder icon in the folder view on the right, then double-click again to open the next sub-folder, and so on, until you locate the document, which you then double-click on to open. The disadvantages of this approach are; you have to open all the folders as well as the document, and you have to know exactly where the document is.

Editing non-Protel Documents

As well as being able to store non-Protel documents inside the Design Database, you can also edit these documents, and have the updated document saved back into the Design Database.

Documents created by OLE Servers (such as Word and Excel documents) can be edited directly from the Design Explorer. When you click to open a Word document that is stored in a Design Database Microsoft Word starts, and the Word document is opened in it, ready for editing. When you save changes made in Word, the document is written back into the Design Database.

Importing an External Document into the Design Database

To import a document into the Design Database right-click in the folder view to display the floating menu, then click on Import in the menu. The Import Document dialog will appear. Navigate to locate the required document, select it, then click Open to import it. A copy of the document is saved into the Design Database, and an icon is added into the current folder.

◆ When you import a document the original copy is not removed from your hard disk.

The document icon shows what document editor is used to edit this document. For example, a Protel PCB document will have a PCB icon, indicating that it will be opened by Protel's PCB Editor. A Microsoft Word document will have the familiar Word icon, indicating that it will be opened by Microsoft Word. An AutoCAD® DWG document will have the AutoCAD icon. When you double-click on an AutoCAD icon the AutoCAD editor will launch outside of the Design Explorer, and the DWG document that you double-clicked on will open.

If there is no recognized editor for the document being imported it is given a special icon, indicating that it can not be opened by double-clicking on the icon. You can change the editor that is assigned to the document, right-click and select Properties.

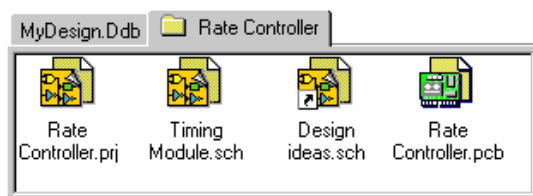
◆ You can also drag and drop documents and folders directly from the Windows File Explorer to import them into the Design Explorer. Any folder hierarchy that exists in a dragged folder is automatically recreated in the Design Explorer.

Linking to an External Document

As well as storing documents within the Design Database, you can also link to external documents. Linking allows you to associate a document with a project, without having to store it within the Design Database.

Linked documents are edited in the Design Window in the normal way.

There are disadvantages to linking to an external document. You must remember that the document is external, so backing it up, moving it, deleting it – managing that



Design ideas.sch is not stored inside the Design Database. It is a linked document – as shown by the small arrow on the icon.

document – must be done separately from managing the Design Database. The other disadvantage is that you cannot lock an external document while you are working on it, so you cannot protect against the possibility that two designers may have the same linked document open, and overwrite each other's work.

To link to an external document right-click in the folder view to display the floating menu, then click on Link in the menu. The Link Document dialog will appear. Navigate to locate the required document, select it, then click Open to Link to it. An icon is added into the current folder. Note that the icon has a small arrow in the bottom left corner, indicating that this is a linked external document.

Exporting Documents from the Design Database

You can export a copy of a document from the Design Database, out to your hard disk (or network drive). To export a document right-click on its icon in the folder view to select it and display the floating menu, then select Export from the menu. The Export dialog will appear – navigate to the location you want to save the document, then click Save. Note that the document is not deleted from the Design Database, it is just copied out to the location you specified.

◆ You can also export an entire folder (with sub-folders). A folder of the same name is created on the hard disk, and all documents and sub-folders in the selected Design Database folder are exported.

Managing Documents in the Design Database

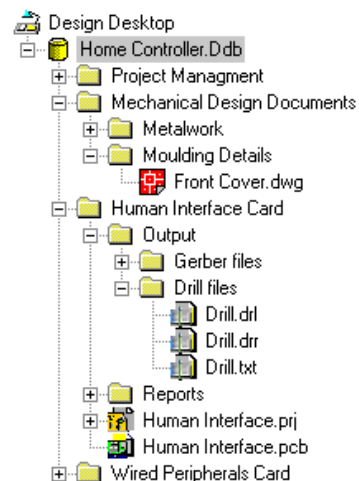
Use the Design Explorer to manage the documents inside your Design Database – just like you use the Windows File Explorer to manage documents on your hard disk.

Using Folders to Create Hierarchy

You can manage the documents within the Design Database by creating folders, to categorize and store the documents in a way that suits your design. Like the Windows File Explorer, you can create a hierarchy of folders, with folders within folders.

To create a new folder, right-click in a folder view in the integrated Design Window, then click on **New** in the floating menu that appears. Click on **Folder** in the New Document dialog, and click OK.

The new folder will appear with a generic name, like *Folder1*. To rename the folder click once on the folder to select it, then select **Edit » Rename** from the menus (or press the F2 shortcut). The text will appear highlighted, ready to type in the new name.



Use folders to organize your documents in the Design Database

Moving, Copying and Deleting Documents

The Design Explorer supports all the familiar File Explorer shortcuts:

- SHIFT+click, CTRL+click, and click and drag a window, to select documents and folders
- Press the DELETE key to delete the current selection
- Drag and drop from one folder into another folder, to move the current selection
- Right-click, drag and drop a selection to pop up a menu, where you can choose between Copy, Move and Create Shortcut
- Press CTRL+C to copy the current selection, then CTRL+V to paste
- Right-click on a selection to display a menu, you can Copy, Cut, Delete, etc.

Renaming a Document or Folder

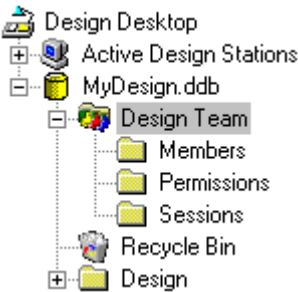
To rename a document or folder first select it (click once on its icon), then select **Edit » Rename** from the menus, or press the F2 shortcut key. An editing box will appear around the name, and the text will be selected. Type in the new name, including any file extension. Protel 99 SE does not use the file extension to identify documents inside the design database – you are free to use any extension you like, or none at all.

Setting up a shared Database for your Design Team

Protel 99 SE incorporates Design Explorer's *SmartTeam* technology. This technology allows document access control to be defined by the design team, and stored in the Design Database. It provides similar functionality to standard network access control, such as setting up members, and defining folder level and document level access permissions.

When you create a new Design Database the access control features will not be enabled. Access restriction is enabled only when the Admin member has a password specified. Once this has been done anyone who attempts to open the Design Database will be prompted to provide a member name and password. To do this they will need to be defined as a Member.

After creating a new Design Database use the Design Explorer Navigation Panel to open the DesignTeam folder. Click on the Members folder to open it and create new Members, click on the Permissions folder to open it and define their Permissions.



◆ A Design Database will present a login dialog once a password is defined for the Admin member.

Creating Team Members

When you create a new Design Database there are two members created by default, *Admin* and *Guest*. To create a new member right-click in the Members folder view, and select New Member from the floating menu.

In the User Properties dialog that appears you need to define: *Name* of the member (used at login, in the Sessions window, and when defining Permissions), their *Description*, and their login *Password*.

◆ **Note:** New Members and their Passwords can only be defined by logging in as Admin.



Defining Team Member Permissions

The default behavior of the Design Database Permissions system is to allow members freedom to Read, Write, Create and Delete, which is then restricted by the creation of Permissions.

Permissions work in a hierarchical manner – permissions defined at a higher level in the database can be overridden by permissions defined at a lower level in the database.

♦ **Note:** Permissions can only be defined by logging in as Admin.

Default Permissions

When you create a new Design Database the following default permissions are created:

Member	Document	Permissions	Definition
Admin	\	[R,W,D,C]	The backslash signifies the root (or highest folder) of the Design Database. This permission means that the Admin member can Read, Write, Delete and Create, in the root of the database, and in any folder below it. This means that they have complete freedom throughout the database.
Guest	\	[R]	A Guest member can Read any document, from the root of the design database down.
[All members]	\	[R,W,D,C]	“*” is the wildcard character, meaning “every Member”. This permission means that every Member (except a Guest) can Read, Write Delete and Create, anywhere in the database. To make the database restricted, and then allow permissions, change this general permission to Read only, or no permissions.
[All members]	\Design Team	[]	This permission means that every Member has no rights in the Design Team folder, or any folder below it. They will be able to look in the Members, Permissions and Sessions folders, but not open anything in any of them. This prevents anyone except Admin from modifying Members or Permissions.
[All members]	\Design Team\Sessions	[R,W,D,C]	This permission means that every member has complete rights in the Sessions folder. This rule overrides the restriction on Sessions created by the Design Team folder rule above.

An Example of Permissions

The following example demonstrates the use of Permissions:

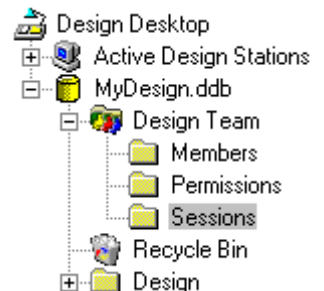
Permissions		
Member	Document	Permissions
Admin	\	[R,W,D,C]
Guest	\	[R]
[All members]	\	[R]
[All members]	\Workgroup	[]
[All members]	\Workgroup\Sessions	[R,W,D,C]
Gustaf Yermoni	\Design	[R,W,D,C]
Gustaf Yermoni	\Design\Rate Controller.pcb	[R]
Joe Citizen	\Design	[R]
Joe Citizen	\Design\Rate Controller.pcb	[R,W]
Vern Huntz	\	[R,W,D,C]

1. Admin has Read, Write, Create and Delete Permissions throughout the database.
2. Guests can Read any document in the database.
3. [All members] can Read any document in the database. The default for this Permission is Read, Write, Create and Delete – in this Design Database this has been changed to Read only, removing this general freedom. This means that permissions that allow access must be added.
4. Team member Gustaf can Read, Write, Create and Delete from the Design folder down, except for the Rate Controller.pcb, which he can only Read.
5. Joe can only Read from the Design folder down, except for the Rate Controller.pcb, which he can Read and Write. Joe cannot Create new files or Delete files (including Rate Controller.pcb).
6. Vern's permissions allow him to Read, Write, Create and Delete throughout the database.

Monitoring what Documents are Open

It is important that you can tell exactly what documents each team member has open when you are working together in a Design Database. To see what documents in the Design Database are currently open, open the Sessions folder.

The Sessions folder will list; the Name of the document, the location in the Design Database, the Team Member that has it open, the name of the machine (PC), and the Locked Status.



Below is an example of a sessions folder:

MyDesign.Ddb Rate Controller.prj Timing Module.sch Sessions				
Name	Location	Member	Machine	Status
Rate Controller.prj	\Design\	Gustaf Yermoni	Gustaf_Y	
Timing Module.sch	\Design\	Gustaf Yermoni	Gustaf_Y	Locked
Rate Controller.pcb	\Design\	Joe Citizen	Joe_C	Locked
Rate Controller.pcb	\Design\	Vern Huntz	Vern_H	
Timing Module.sch	\Design\	Vern Huntz	Vern_H	

1. Gustaf has two documents open, Rate Controller.prj and Timing Module.sch. He has Timing Module.sch locked, and is in the process of locking Rate Controller.prj. Other team members that have permission to open these documents still can, but they will not be able to save these documents, even if their Permissions normally allow this.
2. Joe has the Rate Controller.pcb open, and has locked it.
3. Vern has Rate Controller.prj and Timing Module.sch open. Both of these documents have been opened and locked by other team members, so Vern will not be able to save changes to these documents, even though his permissions allow this.

◆ Select **View » Refresh** from the menus to refresh the list in the Sessions folder.

Locking a Document

Right-click on a document in the Sessions folder to lock or unlock it. The figure at the top of this page shows a document in the process of being locked. Once a document has been locked it can only be unlocked by the team member that locked it. When the team member that locked a document closes it, the document is automatically unlocked.

Locked documents can still be opened by other team members that have the appropriate permissions. When another team member attempts to open a document that is already open and locked, a message appears. The message details which team member has opened and locked the document, and asks if they would like to open the document in Read-Only mode.

◆ Right-click on a document name in the Sessions folder to lock that document.

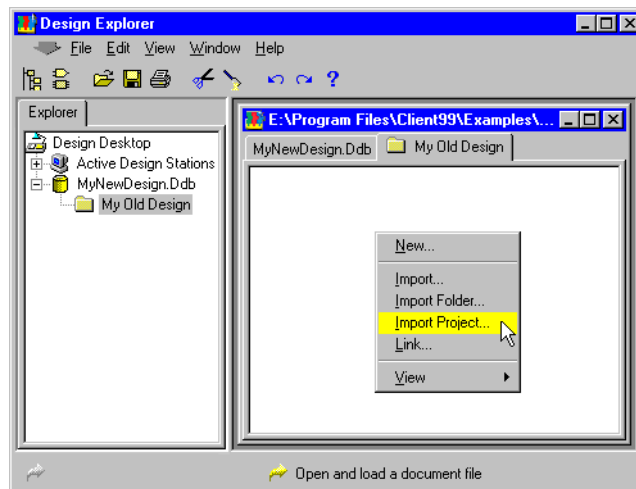
Working with your Existing Protel Designs

In Protel 99 SE, any document that you want to open in the Design Explorer must be stored inside a Design Database (or linked to it). You cannot open an existing schematic or PCB document, edit it and then save the changes back to the hard drive. If you open an existing document it will automatically be imported into a new design Database, and then opened for editing.

You can choose **File » Open** from the menus, set the File Type to All Files (*.*), then select an existing Protel document on your hard drive and click Open. When you do, you will immediately be prompted to save the document as a Design Database. After entering the Database File Name and clicking OK, the original document that you selected to open is imported into the Design Database, then opened for editing. The document that is displayed ready for editing is the copy in the Design Database, not the original copy on the hard disk. If you wanted to refresh the external copy on the hard drive you would need to export the edited document out of the Design Database.

Converting an Existing Design into a Design Database

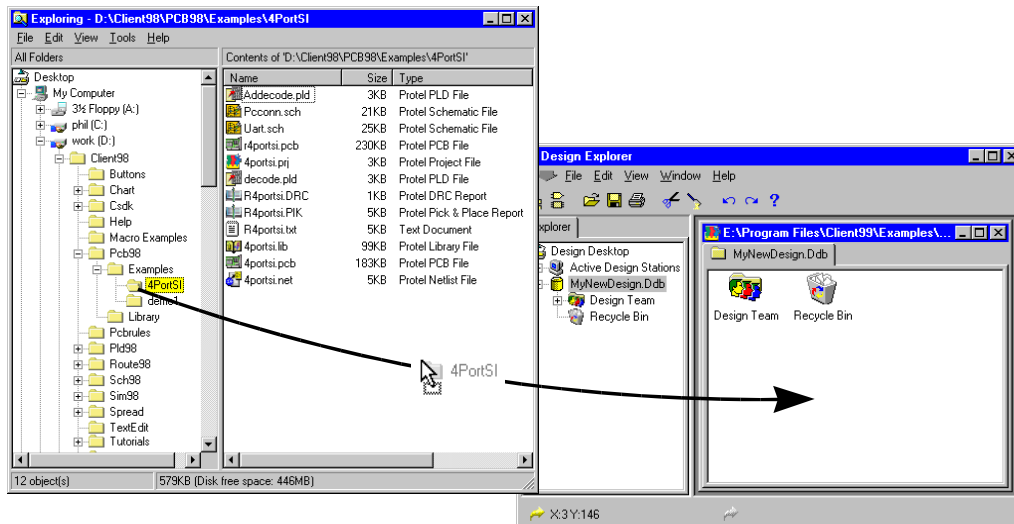
When you open an existing Protel document in the Design Explorer it is automatically imported into a new Design Database. If you would like more control over the process, then you can create an empty Design Database first, create the internal folder structure you would like, then right-click in the folder view to display the popup menu and select Import, or Import Project.



After creating a new Design Database right-click in the folder view to import your project

Using Drag and Drop to Import

You can also click, drag and drop from the Windows File Explorer directly into the Design Explorer. This is the most efficient way of bringing a large number of documents into a Design Database. If you click, drag and drop a folder into a Design Database all documents and sub-folders are imported. The folder hierarchy that existed on the hard disk is automatically recreated within the Design Database.



You can drag an entire design folder from the File Explorer, and drop it straight into a Design Database open in the Design Explorer

Linking to External Documents

Sometimes you may need to keep a document on the hard drive. In these circumstances it is better to link the document to a Design Database, rather than importing it. Documents that are linked to a Design Database are not stored inside the Database, they remain on the hard disk. This means that they can still be used by other designers, using older versions of the Protel design software.

Working with Existing Company Libraries

If you have existing libraries that you need to keep as .LIB format files, you can still work with these in Protel 99 SE. The easiest way to be able to use libraries in both Protel 99 SE and Protel98 is to create new Library Design Databases, and link the existing company libraries to them, rather than importing them.

Synchronizing an Existing Schematic Project and the PCB

In Protel 99 SE design information is passed between the schematic and the PCB using the Design Synchronizer. The Synchronizer makes it very easy to transfer design information from the schematic to the PCB, and back again.

When you run the Synchronizer you nominate which is the target document by selecting either **Update PCB**, or **Update Schematic**, from the menus. When you do this the Synchronizer will display the Update dialog, which you use to control the synchronization process.

Choosing the Synchronization Direction

When you run the Synchronizer you must choose which will be the reference, and which is the target. Information is extracted from the reference, and used to update the target. You do this by either updating the PCB from the schematic, or updating the schematic from the PCB.

◆ When it is initially matching, the Synchronizer relies on the component designator – it is important that the schematic designators correspond with the PCB designators.

How do you choose which way to initially run the synchronizer? Ask yourself which is more up-to-date – the schematics, or the PCB? Have you changed any designators, footprints or component values on the PCB since initially loading the netlist? Have you started doing design updates on the schematics? When you have decided which is more up-to-date, select either **Update PCB**, or **Update Schematic** from the appropriate **Design** menu.

◆ If you update from the Schematic to the PCB, both the component and the connectivity changes are transferred to the PCB – if you update from the PCB to the schematic, the component attribute changes are transferred, and new components and connectivity differences are listed in the Change Report.

Running the Initial Synchronization

The Design Synchronizer uses a matching identifier to match each schematic component, to its PCB component. When you synchronize an existing design these matching identifiers need to be assigned. This will happen the first time you synchronize the imported schematic and PCB.

When you select one of the Update options the Synchronizer's Update dialog appears, with the name of the target document in the title bar. As soon as you press the View Changes, or the Execute button, the Confirm Component Associations dialog will appear. This dialog will assign the matching component identifiers.

For a detailed description of how to use the Design Synchronizer, and how it works (including details of the Confirm Component Associations dialog), refer to the chapter *Transferring the Design Information to the PCB*, in the *Schematic Capture* section of the Handbook.

The Technology inside Protel 99 SE

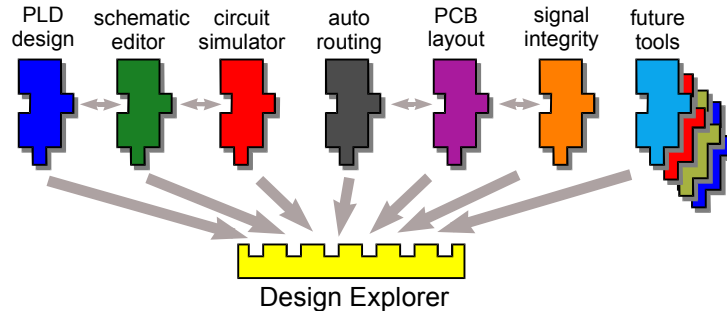
Protel 99 SE is built on sophisticated, state-of-the-art, smart-software technology; *SmartTeam*, *SmartDoc* and *SmartTool*. In Protel 99 SE these technologies bring together:

- a complete set of PCB design tools integrated into the one user environment,
- the ability to store and manage all the design documents in a single Design Database,
- teamwork collaboration features that give you total control over document access and locking.

SmartTool Technology

Protel 99 SE is built on SmartTool technology. SmartTool technology brings together Protel Editors and OLE compliant editors, accessible through a single user interface, the Design Explorer.

The foundations of the SmartTool technology is its client/server architecture, which separates the user interface (the client), from the tools, or editors (the servers).



The figure shows how each server plugs into the Design Explorer, and how a server can also communicate directly to another server

What is Client/Server Architecture?

Protel 99 SE's client/server architecture is a new approach to integrating design tools on a PC. Rather than each design tool being developed as a separate, stand-alone application, client/server architecture separates out the user interface (client), from the design tools (servers). In Protel 99 SE the Design Explorer is the client, and each of the tools is a server.

What is the Design Explorer?

The Design Explorer is the application, or executable, as it is often called. When you select Design Explorer from the Windows Start menu this is the executable that is started.

Externally, the Design Explorer presents all the features that the user interacts with – the navigation panel, the menus, the toolbars and the shortcut keys.

Internally, the Design Explorer is the platform that each server plugs in to. When a server is plugged in, it tells the Design Explorer what functions (or processes) it includes, and passes over a definition of all its menus, toolbars and shortcuts (resources). When a user selects a menu item the Design Explorer passes a message to the appropriate server, telling that server what process to run.

What is a Server?

In everyday computer-user language, a server is a module that plugs into the Design Explorer, to add new functionality to the environment. It can range from a complete document editor, such as the Schematic Editor, to a complex analysis engine, such as the Circuit Simulator, through to a simple utility that counts all the holes on a printed circuit board.

In software jargon, each Protel 99 SE server is a DLL (Dynamic Link Library). In Microsoft Windows a DLL is a library of functions and procedures, which can be used by any application, and other DLLs. Microsoft developed the EXE/DLL model to allow software to be reusable. Software functions that are used by more than one application are stored in these libraries, which can then be called when the application needs that function. Windows is structured so that using a function from a library (DLL) is as quick and easy as using a function that is internal to the application.

Protel 99 SE extends this model by making the functions and procedures inside each server DLL directly available to the user – through the menus, toolbars and shortcuts in the Design Explorer.

As well as exposing the functionality of a server to the user through the Design Explorer menus, toolbars and shortcuts, each server exposes its functionality to other servers through an open Application Programming Interface (API). The API is the definition of how all the functions in a DLL are used. An API is called “open” when this definition is published, so that the functions in this DLL can be accessed by other EXEs and DLLs.

As well as allowing programmatic access to the same functions that the user can access through the Design Explorer resources, the API also includes more powerful functions that support direct manipulation of information in the design document that the editor currently has open. An example of this is the PCB server – when it has a PCB document open for editing the autorouter can examine the contents of this PCB document, directly through the API. Using this mechanism it can extract information about the objects on the PCB, and pass back routing information (tracks and vias) into the PCB document.

Types of Servers

The Protel servers can be grouped into three categories:

Document Editor/Viewers – these servers present a document editing (or viewing) window. Examples include the Schematic Editor and the PCB Layout Editor.

Wizards – these servers pop up as a Wizard, where you step through a number of pages and answer questions. An example is the PCB Component Creation Wizard.

Utility Servers – these servers work with one of the Document Editor servers. Typically they add items to the Document Editor's menus to allow access to their features. An example is the PCB Autorouter, which adds the **AutoRoute** menu to the PCB menu bar. Another example is the Circuit Simulator, which adds the **Simulate** menu to the Schematic Editor's menu bar.

♦ For information on adding, removing and resetting a server refer to the topic, *Working with Servers*, in the chapter, *Customizing the Design Explorer*.

What is an OLE Server?

OLE (Object Linking and Embedding) is a high-level information exchange technology developed by Microsoft. OLE was developed to allow applications to exchange information and functionality. There are 2 distinct parts to the OLE model – OLE Servers, that expose their functionality to other applications, and OLE Controllers, that are able to call on the services of an OLE Server. The Design Explorer is an OLE Controller.

Applications that can run as OLE Servers include Microsoft Word and Excel, and the Visio graphical design and modeling software tools.

You do not need to install OLE Servers in the Design Explorer, the Design Explorer will automatically identify all the OLE Servers that are currently installed on your PC.

SmartDoc Technology

SmartDoc technology redefines document integration and document management, by bringing together all the design team documents into a single, integrated, Design Database. Any type of document can be stored in a Design Database, from the schematic sheets, the PCBs, the Gerber files – through to reports and spreadsheets prepared in Microsoft Word and Excel, mechanical drawings created in AutoCAD, and project drawings created in Visio.

Using the Design Explorer you can work with the documents in the Design Database just like you work in the Windows File Explorer – create folders, copy, paste and delete documents, drag and drop documents between folders, in fact you can even drag and drop from the Windows File Explorer straight into the Design Explorer.

SmartTeam Technology

SmartTeam technology allows design team collaboration in a way that has not been possible before. SmartTeam technology lets all your design team members work inside the same Design Database, at the same time. Once the team members are specified, you can control exactly what folder and document access rights each team member has.

Each team member can see what documents are currently open by other team members, and with a click of the mouse they can lock documents to prevent inadvertent overwriting.

All the Team functionality is specified and controlled at the Design Explorer level, making it both network and network-department independent.

Customizing the Design Explorer

The Design Explorer is completely user-customizable, you can reposition the toolbars and panels, as well as create new toolbars, menus and shortcuts.

If you would like to create a new toolbar, add a menu item, or define a new shortcut key, then you need to understand all about *resources* and *processes*.

Resources – Menus, Toolbars and Shortcut Keys

Within the Design Explorer environment you can perform operations such as opening and closing documents, editing these documents, and generating output based on the documents. All the operations that you perform in the Design Explorer are initiated via the menus, toolbars and keyboard shortcuts. Menus, toolbars and keyboard shortcut lists are known as *resources* in the Design Explorer environment.

All of the resources that are available in the Design Explorer are fully customizable. You can add buttons to a toolbar, reorganize the menus, and add your own shortcut keys.

Understanding Resources

All the ‘doing’ functions in the Design Explorer, such as placing a component, changing the zoom level, redrawing the screen and so on, are performed by processes. You access these processes through the resources. For more information on Processes refer to the next topic, *Understanding Processes*.

There are three types of resources; menus, toolbars and shortcut keys. During the initial installation process the default resource definitions for each tool are loaded into the Design Explorer and stored in a common resource definition file. Any modifications that you make are made in the common resource file. You can restore the resources of any tool back to the defaults at any time.

Menus

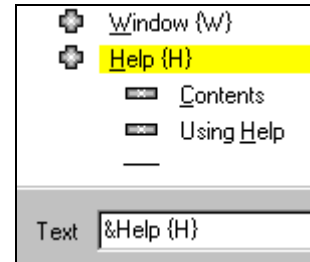
All the menus are organized so they are consistent with the Windows model. This means that standard operations, such as opening and saving files, printing, or using standard editing operations such as Cut or Paste, behave the same in each editor. This makes you more productive as you work in the different Editors.

◆ To edit a menu double-click on the menu bar.

Pop-up Menus

The Design Explorer supports special shortcut key assignments for direct keyboard access to menus in the graphical editors. For example, pressing F in the Schematic Editor will pop up the File menu, pressing T will pop up the Tools menu. This provides a convenient way to access menus (and sub-menus) directly from the keyboard.

The shortcut key is defined by the character in the {} braces, in the Text field where the menu text is entered.

**Toolbars**

The toolbars in the Design Explorer can be fixed to any side of the workspace, or be set to floating, where they can be repositioned anywhere in the environment.

New toolbars can be created and existing toolbars edited, linking any of the processes currently available in the Design Explorer, to any button. There is an example of how to create a new toolbar later in this chapter.

◆ To edit a toolbar double-click on the title region of the toolbar.

Keyboard Shortcut Keys

Each document editor includes at least one shortcut key table. These tables can be edited, and new shortcut key tables can be created.

Keyboard shortcuts can include key combinations, including CTRL, SHIFT and ALT, in combination with either one or two keys. To edit a shortcut table select **Client menu » Customize**.

◆ As well as being able to map keyboard shortcut keys directly to processes, keyboard keys can also be mapped to popup a menu. If a key has mistakenly been mapped to popup a menu and to directly launch a process, then the process mapping takes priority.

Default Menus, Toolbars and Shortcuts

The default resources for each server are defined in a resource file (*.RCS). This file contains the definitions of the default menus, toolbars and shortcut key tables. These resources are known as system resources and can not be removed from the Design Explorer environment, but they can be customized.

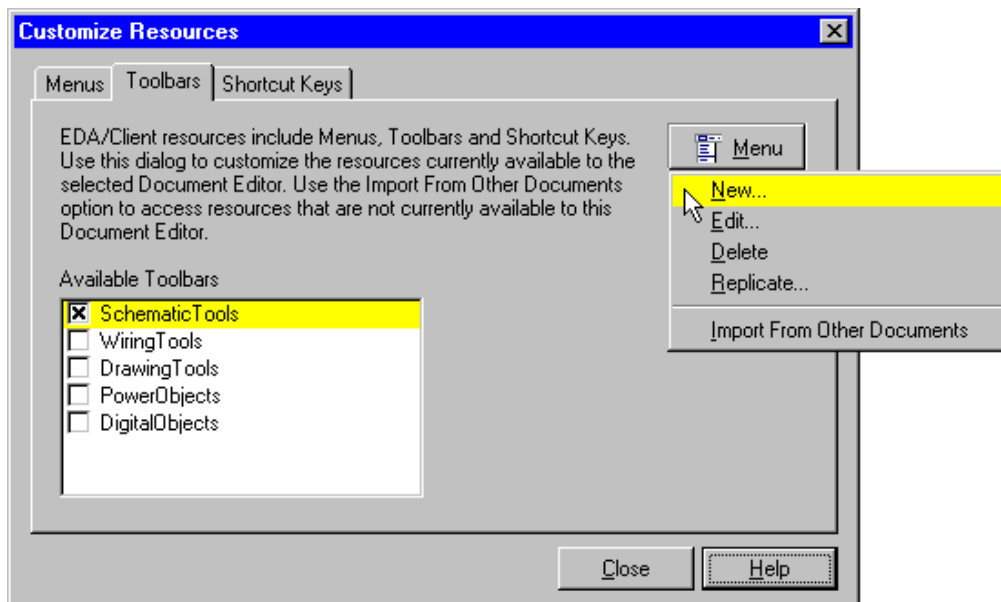
During the initial installation process the default resource definitions for each tool are loaded into the Design Explorer and stored in a common resource definition file. Any modifications that you make are made in the common resource file. You can restore the resources of any tool back to the defaults at any time.

Resetting the Resources back to their Defaults

You have total freedom in customizing the menus, toolbars and shortcut keys in the Design Explorer. If necessary you can restore the menus, toolbars and shortcut keys back to their original state at any time. To do this, select the **Client menu » Servers** menu item. Double-click on the required server's icon in the EDA Servers dialog, then press the Defaults button. The menus, toolbars and shortcut keys for the selected server will be returned to their default state.

- ◆ Sometimes when you reset the resources you may find that some menu items disappear. This happens because those missing menu items were plugged in by another server. For example, if you reset the resources for the Schematic Editor, the Simulate and PLD menus will disappear. To recover these you must also reset the resources for the Sim server and the Pld server.

Customizing Resources



When you would like to add a new shortcut key, change the menu to use your custom menu, or display a particular toolbar, you need to *customize* the resources.

To customize the resources *for the active document editor* select the **Client menu » Customize** menu item, to pop up the Customize Resources dialog box. If a schematic sheet is the active document then the Customize Resources dialog will give you access to the resources currently available to the Schematic Editor.

Click on the Tab at the top to select the resource that you wish to work on, then click on the Menu button and select the appropriate command.

By clicking on the Menu button you can:

Delete a Resource

Delete the selected resource. When you select this menu option you will be prompted to Confirm the deletion. You will also be asked if you want to “Remove from Global Resource Pool”. If you disable this option the resource is not removed from the Design Explorer environment, only the link from the resource to this editor is removed.

Replicate a Resource

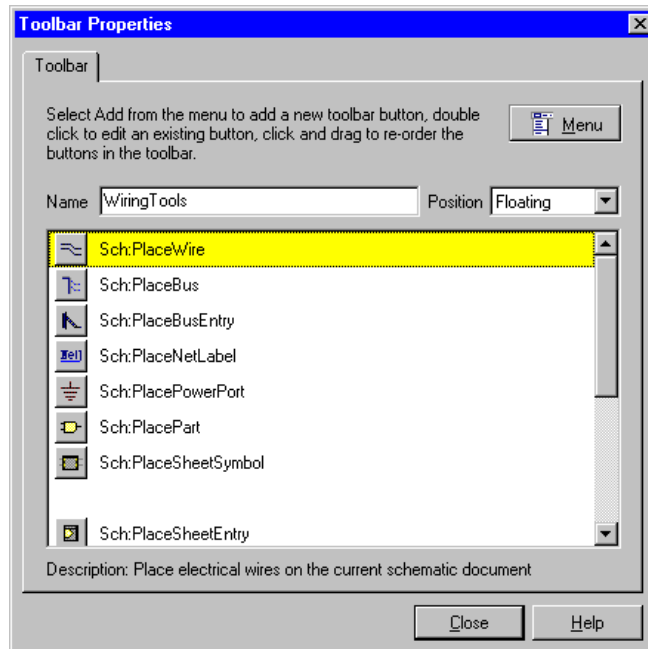
Select Replicate when you wish to create a new resource, but you do not wish to start from scratch. The selected resource will be replicated, and then opened, ready for editing.

Import From Other Documents

Use this option to access resources that belong to other servers, or custom resources that you have created.

Editing a Resource

Selecting Edit from the menu pops up a dialog which allows you to edit the selected toolbar, menu or shortcut key table.

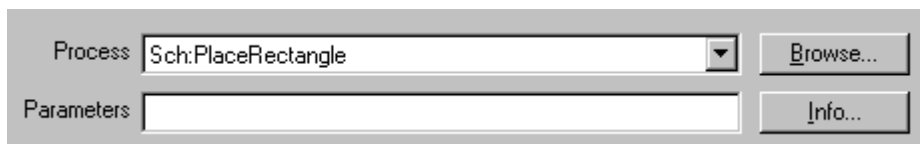


◆ You will find many shortcuts when working in the resource editing dialogs. Right-click to pop up the menu; use the INSERT and DELETE keys to add and remove items; click and drag to re-order; double-click to change the process that is mapped to a resource item.

Assigning a Process to a Menu Item, Shortcut Key or a Toolbar Button

There are essentially two steps to building resources:

1. Create the toolbar button, menu item or shortcut key.
2. Assign the process that will be launched when you select this resource item.



The image shows a dialog box with a light gray background. It contains two main sections. The top section has a label 'Process' followed by a text box containing 'Sch.PlaceRectangle' and a small downward arrow. To the right of this is a button labeled 'Browse...'. The bottom section has a label 'Parameters' followed by an empty text box. To the right of this is a button labeled 'Info...'.

Regardless of the resource, the way you map a process to a resource item is essentially the same. Each dialog that allows you to perform this function will have a region as shown above, where you specify which process is to be launched when you click on this button / menu item / shortcut key.

Finding the Process

The Process field specifies the process that is launched when you select this resource item. Use the Browse button to pop up the Process Browser – use the Browser filtering features to quickly locate the required process.

Need more Info about a Process?

Press the Info button to pop up On-line help on the selected process, complete with a description of any process parameters.

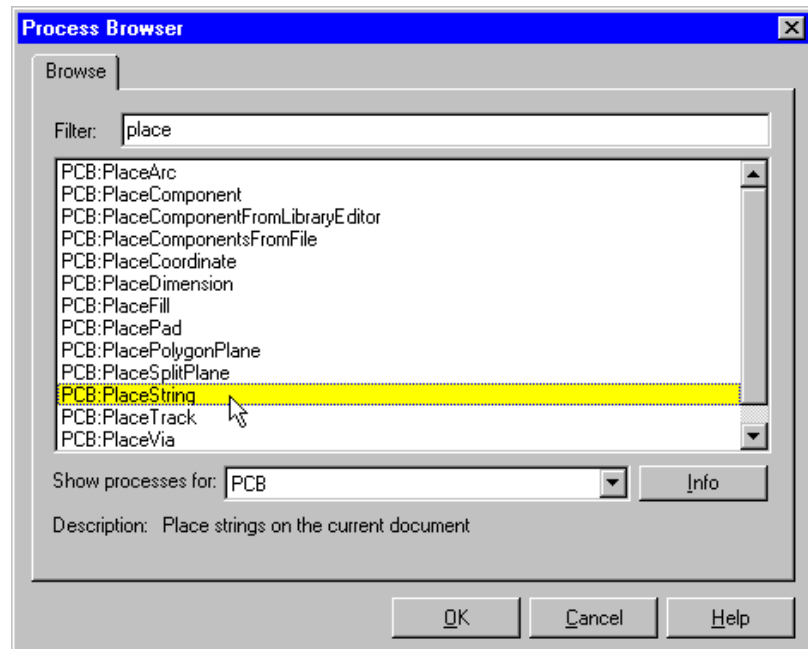
◆ For more information on Processes and Process Parameters, refer to the chapter, *Understanding Processes*.

Example: Creating a New Toolbar

There are two ways of creating a new toolbar. An existing toolbar can be replicated and then modified, or a new one can be built. This example shows how to build up a new toolbar for placing primitives in the PCB Editor.

1. With a PCB document as the active document Select the **Client menu » Customize** menu item.
2. Click on the Toolbar Tab and select **New** from the Menu to pop up the Toolbar Properties dialog.
3. Type the name “Primitives” into the Name field.
4. To add a button to the toolbar first click in the large empty white area, and press INSERT on the keyboard. A blank button will appear in the Toolbar window, now you need to map a process to this button.
5. Double-click on the new button to pop up the Button dialog. We will now use the Process Browser to locate the process.
6. Press the Browse button to pop up the Process Browser dialog.
7. Set the “Show Processes for” drop down list to PCB.

8. Type the word “place” in the Filter field.



The Process List will now display all the PCB processes that start with the string *place*. Note that the server name before the colon (:) is not considered by the filter. The * (any characters) and ? (any single character) wildcards can be used in the filter.

9. Select the PCB:PlaceString process and click the OK button (shortcut: double-click LEFT MOUSE).

You will return to the Button dialog. The next step is to assign the bitmap that you would like to use with this button.

10. We will use button bitmaps that already exist. To locate the button bitmaps click on the Browse button next to the Bitmap File field.

The Image File dialog will pop up. Button bitmaps are stored in the \Program Files\Design Explorer 99 SE\System\Buttons folder.

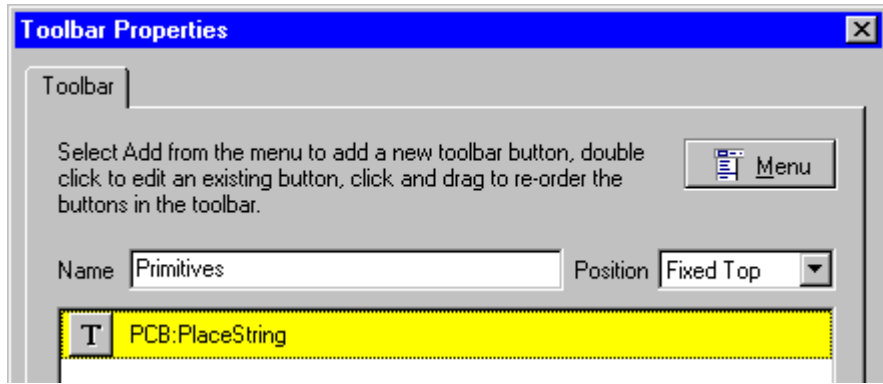
11. In the File Name field enter the string “t*.bmp” and press enter.

The dialog will now list all bitmap files starting with the letter “t”.

12. Double-click on the file text.bmp.

The Image File dialog will close, presenting the Button dialog again. Note that the text bitmap has been linked and the Preview button now shows the image of the letter T.

13. Click OK to close the Button dialog box.



The first button on the new Primitives toolbar

14. Repeat steps 4 to 13 to add the other five buttons using the following details.

Process Identifier	Bitmap
PCB:PlaceArc	arc.bmp
PCB:PlaceFill	rect.bmp
PCB:PlaceVia	via.bmp
PCB:PlacePad	pad.bmp
PCB:PlaceTrack	track.bmp

15. To make the toolbar appear as two rows of three buttons, highlight the forth button in the list, then select **Separator** from the dialog Menu. A blank space will appear above this button.
16. Click the Close button to close the Edit Toolbar dialog box.

The new toolbar has been created and will appear in the Available Toolbars List in the Customize Resources dialog.

17. Click OK to close the Customize Resources dialog.

The new Primitives toolbar now exists in the pool of resources available in the Design Explorer environment, and is automatically linked to the PCB Editor.



Understanding Processes

The functionality of each server that is installed in the Design Explorer is exposed by that server's *processes*. Once you come to terms with the concept of processes – how they work and launch them – you are well on the way to being able to customize the Design Explorer environment.

What is a Process?

A process can be thought of as the software executing a sequence of jobs. This job may be something simple, like refreshing the screen, or it may be more complex, like placing a polygon plane.

Any action or operation that is performed in the Design Explorer is carried out by a process. When you select **File » Save** from the menus, the *SaveDocument* process is launched. Selecting the **Place » Wire** menu item in Advanced Schematic launches the *PlaceWire* process, which you then interact with as you place the wire on the sheet. The menu items, toolbar buttons and shortcut keys launch the processes, and are called Process Launchers. The action, or job, is performed by the process.

Each process is identified by a unique *Process Identifier*. The process identifier includes the server name and the process name, separated by a colon. For the two process names mentioned above, the syntax is:

Client:SaveDocument

Sch:PlaceWire

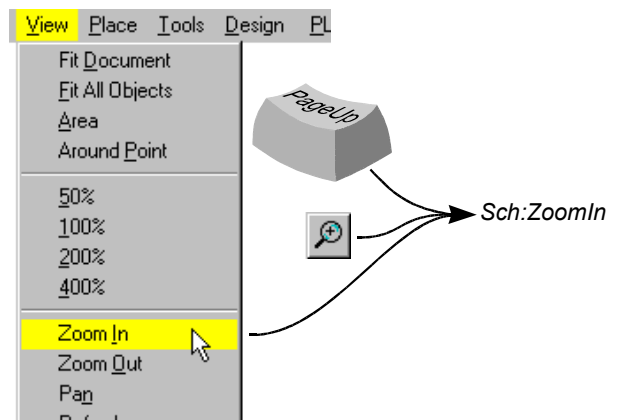
A definition of each process provided by the Design Explorer and each server can be found in the On-line Help. Each of the dialogs where you assign processes includes an Info button. Press this button to pop up the Help file with a description of that process.

Launching a Process

When you select a menu item or click on a toolbar button, you *launch* a process. Processes are launched by passing the process identifier to the appropriate server. The server then executes that process.

To pass the process identifier to the server, a *Process Launcher* is used. Process launchers include:

- menu items
- toolbar buttons
- keyboard shortcut keys



These 3 process launchers (toolbar button, shortcut key and menu item) all launch the same process, Sch:ZoomIn

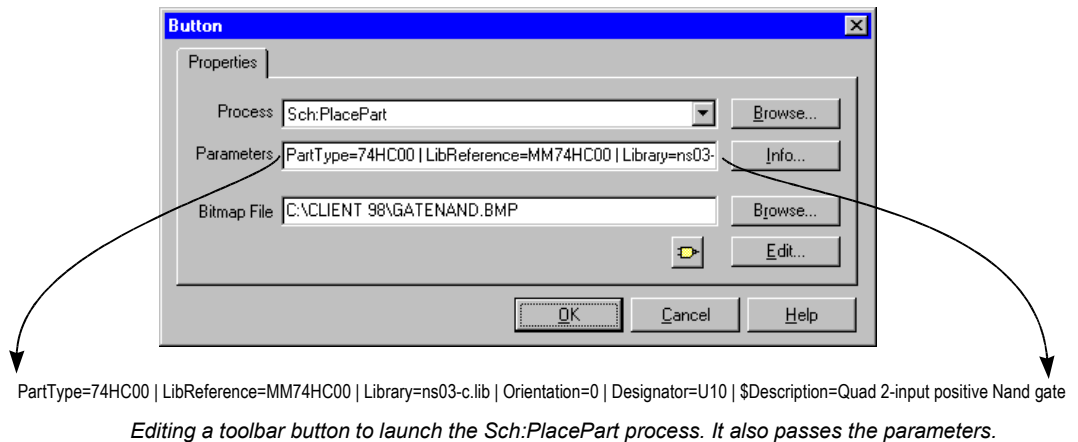
Process Parameters

Processes in the Design Explorer environment are *parametric* – that is, each process can have a set of *parameters*. What are process parameters? Remember how the process can be thought of as the software executing a sequence of jobs. You can think of the process parameters as the instructions on how you want the job to be carried out.

Consider the *Sch:PlacePart* process. When you launch this process (select **Place » Part**) a dialog pops up asking for the Library Reference of the part you would like. After entering this a second dialog pops up asking what Designator to use, then the part appears floating on the cursor. Instead of entering the Library Reference and the Designator in dialogs, you can pass this information when you launch the process.

To pass the process parameters when you press a toolbar button, enter the parameters in the Parameters text field of a process launcher editing dialog box (eg, the Button dialog).

◆ Process parameters can also be passed by a macro.



- Library = The name of the library that the component is in. If the library is not in the standard location, include the full path.
- LibReference = The name of the component in the library.
- Designator = The designator you want to give this component.
- PartType = The part type or comment you want to give this component.
- Orientation = The orientation of the component when it appears on the cursor.
- \$Description = This parameter allows you to create your own button tool tip.

When this button is pressed the 74HC00 will appear on the cursor with the designator specified, ready to be placed.

◆ Use the Info button to pop up the On-line Help file with a complete description of the process parameters.

Parameter Syntax

The syntax for passing process parameters in the Parameters text field is;

```
parameter1 = value1 | parameter2 = value2 | parameter3 = value3
```

Each parameter is separated by the vertical bar “|” (or pipe) symbol. It is not necessary to list parameters in any order, nor is it necessary to pass every possible parameter. Parameter text is not case sensitive, you can type in *LibReference* or *libreference*.

For instant access to information on the parameters of a process press the Info button in the Edit Menu, Edit Button or Edit Keyboard Shortcut dialogs.

◆ Refer to the *Macros* topic in the Online Help for information on passing parameters from macros.

Working with Servers

When you install the Design Explorer on your PC, all the servers are automatically installed into the Design Explorer. Generally you will not need to manually install or remove a server from the Design Explorer. As a server is not loaded into memory until you actually need to use that server, there is no reason to remove a server from the Design Explorer environment.

Types of Servers

The Design Explorer supports four kinds of Servers:

- Protel Document Editors – such as the Schematic Editor
- Protel Wizards – such as the PCB Board Creation Wizard
- Protel Utility Servers – such as the PCB Autorouter
- OLE Servers – such as Microsoft Word and Excel

Protel 99 SE includes an extensive set of default Document Editor Servers. These include:

- Schematic Editor – used to edit schematic sheets and components
- PCB Editor – used to edit PCB layouts and footprints
- Text Editor – used for editing text documents, such as PLD source files
- Spread – used for editing spreadsheets
- Chart – used for creating charts directly from the spreadsheets
- Macro – brings macro programming capabilities to the Design Explorer

Protel 99 SE also includes a large number of Utility Servers. These include:

- Circuit Simulator – analyses and simulates directly from the schematic sheet

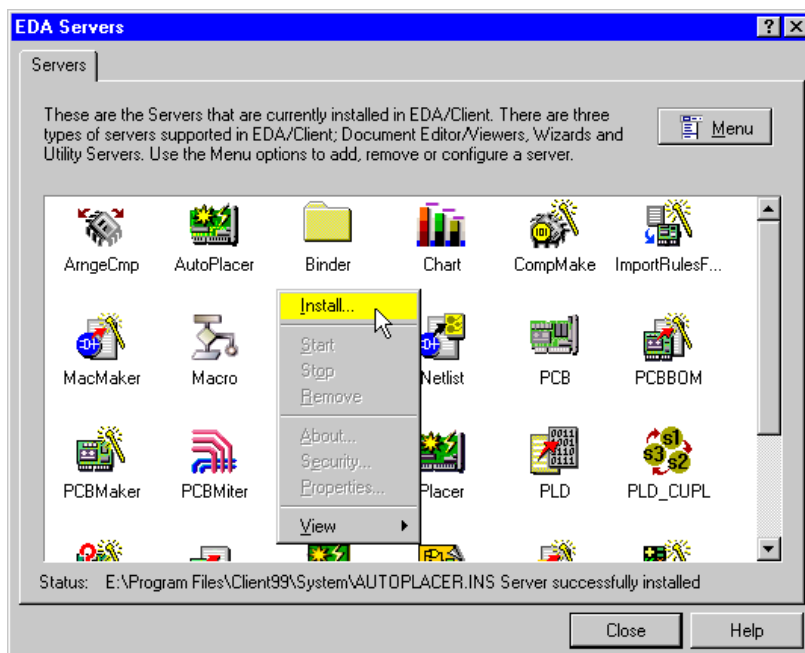
- Netlist – outputs a netlist from a schematic in a variety of formats
- Design Synchronizer – passes design data from schematic to PCB, and from PCB to schematic
- PLD compiler – compiles a PLD design written in CUPL-HDL or created as a schematic
- PCB autoplacer – places the PCB components
- PCB manual router – provides all the manual routing functionality in the PCB workspace, including the push and shove feature
- PCB autorouter – autoroutes the current PCB, directly in the PCB workspace
- Numerous other small utility servers, for creating BOMs, exporting to spreadsheets, creating project libraries, and so on.

Installing a Server

When you install Protel 99 SE from the CD all the servers are automatically installed. If you need to manually install a server:

♦ OLE servers do not need to be installed – the Design Explorer automatically identifies what OLE servers are available on your PC.

1. Select **Client menu » Servers** from the menu bar. The EDA Servers dialog will pop up, with a list of all currently installed servers.



1. Press the Menu button and select **Install**. A document opening dialog will pop up.
2. Locate and select the server installation file (eg, *advsch.INS*, *spread.INS*).
3. Click Open to install the server.

Check the Status at the bottom of the EDA Servers dialog to confirm that the server has been correctly installed.

Starting and Stopping a Server

When a server is first installed it has a Status of “Not Started”. When a server is Not Started it is not occupying any memory. If you do not start it now, it will automatically be started the first time you use it. Normally you do not need to manually stop a server, unless you wish to free up some memory.

Removing a Server

Generally you will not need to remove a server from the Design Explorer. Removing a server does not delete the server’s files from your hard disk, it merely removes the server from the Design Explorer environment.

To remove a server from the Design Explorer:

1. Select **Client menu » Servers** from the menu bar. The EDA Servers dialog will pop up, with a list of all currently installed servers.
2. Click on the icon to select the server that you wish to remove.
3. Press the Menu button and select Remove.
4. A Confirm dialog will pop up, click Yes to remove the server.

♦ A server must be stopped before it can be removed, and it cannot be stopped if there are any open documents using this server.

♦ When you remove a server from the Design Explorer environment all the resources associated with the server are removed. You will lose any customization you have performed on the default resources, such as toolbar buttons you have added. Custom resources that you have created will still be available.

The Text Editor

The Protel 99 SE includes a powerful Text Editor. Having an integrated text editor means there is no need to leave the Design Explorer environment to work with ASCII documents. Reports can be viewed, macro scripts and PLD source files can be written. All general text editing can be performed in the Text Editor.

The Text Editor includes the normal text editing facilities such as cutting, copying and pasting, search and replace. It also includes a feature known as Syntax Highlighting. Syntax highlighting allows you to highlight different elements in the document based on the syntax, where different word types, symbols and identifiers are assigned unique colors. This feature is an excellent document editing aid, particularly when working with documents with a repetitive, structured nature, such as macro scripts, or source code.

To broaden the usefulness of the syntax highlighting feature, the Text Editor allows the definition of multiple languages. Syntax highlighting can be uniquely configured for each of these languages.

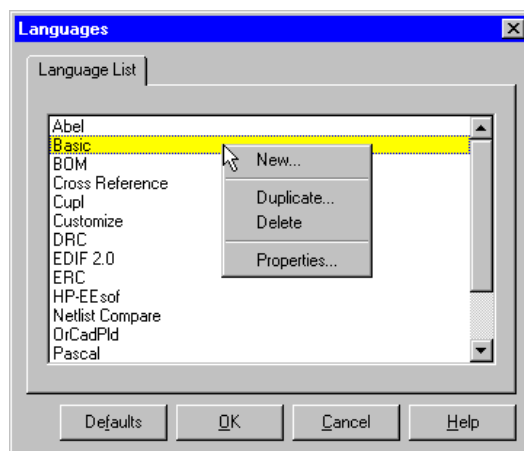
Within each language, there is a set of six types of syntax identifiers available; reserved word, symbol, string, number, comment and identifier. You can define the set of valid words or characters for each of these identifiers, and assign a unique color to each type of identifier.

Languages

The Text Editor includes a number of pre-defined languages, as well as the capability to create new languages. These languages are not the language of a country or culture like French or Mandarin, rather they are a language because each can have their own syntax highlighting definition.

Languages can be created, edited and deleted in the Languages dialog (select **Tools » Change Language** in the Text Editor, or the Change Language button on the panel). Click on a Language in the list and click the right mouse button to pop up the dialog Menu.

The language is associated to a document by the document's file extension. To associate a file extension with a language, double-click on the desired language in the

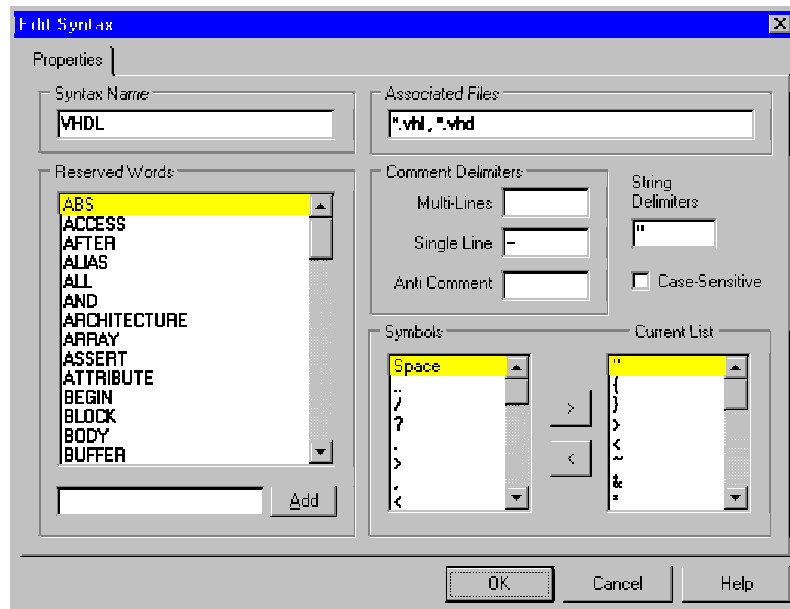


Languages dialog. The Edit Syntax dialog will pop up. In the Associated Files field enter the file extension. For multiple file extensions, separate each with a comma.

Syntax Highlighting

There are two distinct parts to Syntax Highlighting. The first is editing the syntax, the second is assigning the highlight colors to each type of syntax identifier.

To edit the syntax, select the **Options » Change Language** menu item. This pops up the Languages dialog. Select the language you wish to edit the syntax for and press the Edit button. In the Edit Syntax dialog you define the set of reserved words, how comments and strings are delimited, the valid set of symbols and any file extensions to be associated with this language.



Define the syntax for the selected language in the Edit Syntax dialog

Highlight colors are then assigned in the Text Editor Properties dialog (**Tools » Preferences** menu item).

Document Options

Select **Tools » Options** to pop up the Text Editor Properties dialog and set up the user preferences. The Colors used for syntax highlighting are set in the Text Colors Tab.

Macros

The Design Explorer includes a Macro server. The macro server supports the Client Basic scripting language, which is a subset of Visual Basic™.



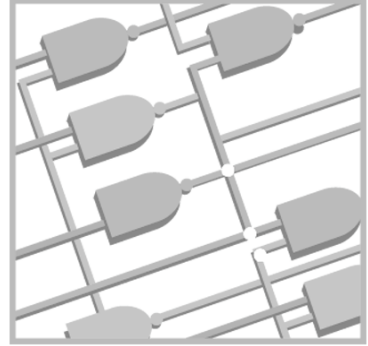
Macros provide a powerful mechanism to enhance your productivity in Protel 99 SE. The macro server supports all the processes available in the Design Explorer environment, and allows passing of parameters to those processes. Macros can be written to work with any server running in the Design Explorer.

Macros can perform anything from a repetitive sequence of processes, through to complex wizards which pop up dialog boxes and respond to user choices. The macro server also supports OLE automation, a feature where operations can be performed in other Windows applications (which also support OLE automation).

Client Basic is interpreted rather than compiled, so macros can be run as soon as they are written. Like all processes in the Design Explorer environment, macros can be launched from any process launcher. Client Basic includes a set of debugging tools which support breakpoints, variable watching, single step, and real-time animation, where the macro is run at a slow enough speed to watch the code being executed.

The macro server includes a comprehensive error flagging mechanism. When an error is encountered, the script file is opened in the Text Editor, the line in error is displayed and highlighted, and a dialog pops up with a description of the error condition.

◆ For comprehensive information on how to write macros in Client Basic, as well as lots of example code, refer to the On-line Help system.



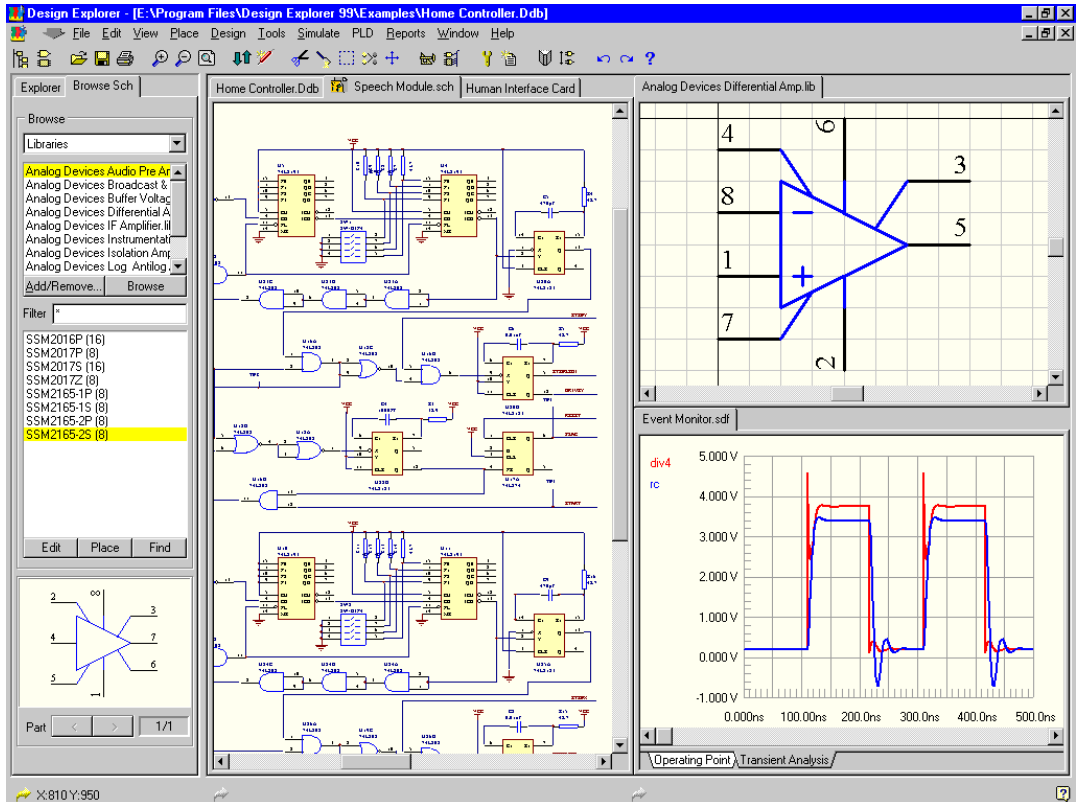
Section 3 _____

Schematic Capture

<i>Schematic Capture – Feature Highlights</i>	53
<i>Fundamentals of Schematic Capture</i>	57
<i>Setting up the Schematic Workspace</i>	64
<i>Working in the Schematic Editing Window</i>	72
<i>Schematic Editing Shortcuts</i>	95
<i>Schematic Design Objects</i>	99
<i>Schematic Components and Libraries</i>	106
<i>Multi-Sheet Design and Project Management</i>	122
<i>Design Verification</i>	136
<i>Preparing the Design for PCB Layout</i>	142
<i>Transferring the Design Information to the PCB</i>	146
<i>Printing Your Schematic Design</i>	154
<i>Schematic Reports</i>	158
<i>Linking to Databases</i>	159
<i>Interfacing to Third-Party Tools</i>	167

Schematic Capture

Schematic Capture – Feature Highlights



This section of the Protel Designer's Handbook guides you through the schematic capture phase of your electronic design project, using Protel 99 SE's Schematic Editor. As you read this section you will find all the information you need to get up and running with the system, and learn how to use the basic features required to design a circuit, perform electrical rule checks, generate a netlist and print out design documentation. Additional, detailed information about the schematic design objects can be found in the on-line Help system.

Schematic Sheet Editor

The Schematic Sheet Editor allows you to create, edit, check and print the schematic sheets that make up a design project. All the tools and utilities needed to perform electrical rule checks, generate reports and create presentation quality schematic drawings are available in the Schematic Sheet Editor.

Schematic Library Editor

The Schematic Library Editor is used to create, edit and manage libraries of component parts. In the Schematic Editor, the term *part* refers to the symbolic entity that represents one part of a multi-part component (for example, one gate in a 7400), or a generic component type (such as a resistor). The Schematic Library Editor shares many common features with the Schematic Sheet Editor, plus specialized tools and features for component part creation and library management tasks.

Design Capabilities

The Schematic Editor is a comprehensive design entry tool. When combined with the Protel Circuit Simulator, PLD design and board layout tools, the Schematic Editor becomes the front-end of a fully automated, integrated, end-to-end design system.

The Schematic Editor can generate single sheet, multiple sheet and fully hierarchical designs of virtually any size, limited only by the available memory and storage capacity of your PC. Sheet sizes include A, B, C, D, E and metric sizes A4-A0, plus user-defined sheets. You can also create custom sheet borders and title blocks, and save custom templates for re-use.

Component Support

The Schematic Editor's standard component libraries include over 65,000 components. The Altium Library Development Center is constantly developing new components, which can be downloaded from the Protel Web site. Visit www.protel.com to download the latest libraries, or to submit a suggestion for a future library.

Hierarchical and Multi-Sheet Support

The Schematic Editor supports single sheet, multiple sheet and fully hierarchical designs including complex hierarchy, where multiple instances of a single sheet can be used in a project. Projects can be navigated visually using the Design Explorer. The Design Explorer displays all the sheets that make up the design in a hierarchical tree structure. You can click on sheet icons to move from sheet to sheet.

Guided Wiring

Special automation features speed the connection of electrical items in the schematic sheets. An electrical grid provides true “snap to” wiring of all electrical items: ports, sheet entries, buses, bus entries, net identifiers, wires and parts. When this feature is active the cursor will jump to the nearest electrical “hot spot” within the range of the electrical grid, and then change shape to indicate the connection point. You need only click (or release LEFT MOUSE) to complete the connection, a junction is automatically added.

Connections between electrical objects are maintained as the objects are “dragged” to a new location on the sheet. The system will automatically add or remove wire segments to maintain orthogonal routing during complex moves.

Flexible Selection

Groups of items can be selected by sheet, by physical connectivity or by designating an area of the drawing. Individual items can be added to or removed from the selection. Selections can be manipulated using standard commands – such as Cut, Copy, Paste or Clear. They can also be moved and rotated. Selections copied to the clipboard can be pasted into other Windows applications. There is also an option to add the template to the clipboard, allowing you to Copy the entire sheet, including the border and title block.

Powerful Editing Options

Design objects (parts, wires, graphical entities, etc) can be edited by double-clicking directly on the item to open its editing dialog, which displays every editable object attribute. Changes to these attributes can be globally applied across a sheet, or across an entire multi-sheet project using specific conditions to define the targets. For example, when editing wires you can change the color or wire size or both attributes. These changes can be globally applied to other wires on the sheet, or to other open sheets. Similar global options are provided for components and other objects.

Text editing is supported by powerful Find and Replace functions, which allow you to define the scope of the edit across multi-sheet projects.

Library System

The Schematic Editor includes comprehensive tools for managing component libraries. Any number of libraries can be opened and accessed without leaving the sheet editor.

Components can also be browsed and placed directly from the Schematic Library Editor. A wide range of standard manufacturers’ libraries is included. Simultaneous multi-user library access is supported for network installations. Parts placed in sheets can be globally updated to reflect library-level changes.

Components include eight read-only (library) text fields and 16 sheet-level text fields of up to 255 characters that can be edited for each instance of a part. You can pre-

define these field names for a component type in the library editor, for convenient reference.

Special Strings

Special purpose pre-defined strings allow you to place date, sheet name, filename, component count and other information to be interpreted at plot time. For example, placing the string “.DATE,” on the sheet, places the current system date on the plot. Special strings can be incorporated in sheet templates.

Font Support

Windows TrueType fonts are fully supported. A system font can be assigned for component pins, port and power object names and sheet reference text. Default fonts can be defined for all other objects that include strings.

Array Placement Options

Linear array placement allows automated step-and-repeat placement of objects in the sheet. This can include individual objects or complex selections of objects. You can specify the number of repeats and set pre-defined x and y offsets, and text increments.

Alignment Tools

Objects can be aligned by their left/right/top/bottom sides, distributed horizontally or vertically or moved to the placement grid.

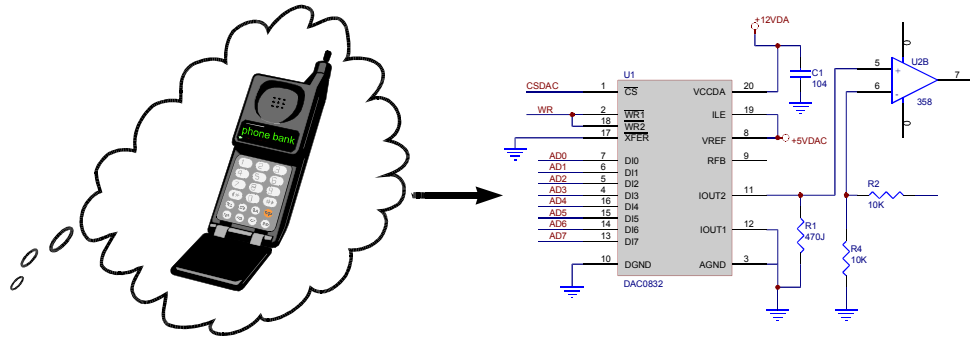
Design Verification Tools

Electrical Rule Check allows quick verification of large or complex drawings. ERC checks are performed in accordance with user-specified physical and logical properties. Options include flagging and reporting a wide range of physical/logical violations including unconnected net labels, unconnected power objects and floating input pins.

Windows Support for Printing and Plotting

Dot matrix and laser printing, color printing, pen plotting and PostScript output are all controlled from a common Print menu item. Any device supported by Windows can be selected. The Schematic Editor allows the production of presentation quality artwork.

Fundamentals of Schematic Capture



Schematic capture is the process of capturing your design concept as a ‘schema’, or diagram, in a computer aided design environment. Defining the circuit at the schematic level allows you to work with a logical model of the design, while maintaining the integrity of the physical model that will become the finished product.

The Computer Model of a Circuit

Outwardly, the use of schematic capture is similar to the traditional drafting process, where graphical symbols for circuit elements (parts, wires, etc) are rendered in drawings, which become a record of the design. Using computer-aided design provides many benefits, by automating the drafting process and allowing easy editing of the design layout.

However, it’s the *capture* part of the schematic capture process that provides the main benefit – creating an integral link between conceptual design of a circuit and its physical expression. By capturing the “logic” of a circuit, this approach allows the integration of simulation and physical layout into the design process. The schematic design process then becomes the design entry point for a number of technologies, from IC design to FPGA and PLD programming, circuit simulation, to PCB design.

The Schematic Editor incorporates many data management facilities that exploit the capabilities of computerized design. For example, each schematic sheet is an independent design document. An automated system links these sheets when they are used together in a hierarchical project. By recognizing these links, it is possible for the designer to globally edit and perform design verification across project sheets, or generate a bill of materials for an entire project in a single operation.

The Schematic Component Model

The schematic components are organized into families of libraries that correspond to manufacturer's data books.

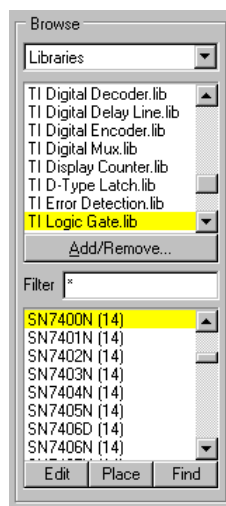
A component stored in the library is comprised of one or more component part descriptions that become the representation of the component on the schematic sheet.

Because components can have multiple parts (e.g., the individual gates in TTL logic components), it is extremely convenient to be able to work with each part of a multi-part component while laying out the design.

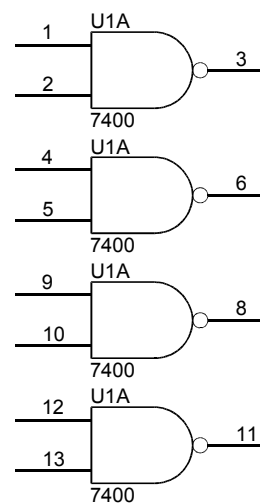
Libraries of Component Models

While an image of the component part is "placed" into the schematic, the component information is always stored in the library. Component creation and editing are always performed at the library level, not on the sheet. This approach maintains library integrity and allows library changes to be used to globally update components in existing designs.

As parts are placed in a sheet, an image of each component is placed in a special cache. The cache acts as a memory resident working library. When a sheet is saved, all the parts used on that sheet are copied from the cache and attached to the sheet file in a 'mini' read only library. The sheet-level library allows you to generate a permanent project library for the design. When the sheet is re-opened, the component descriptions are read from the sheet-level library back into the cache. The component images stored in the sheet-level library (and the cache) can be updated from the source libraries at any time while the sheet is loaded in the Schematic Editor. When they are updated a report is generated listing all parts updated in the cache.



component names are displayed in the panel



The four parts of a SN7400

A single library component can include multiple graphical part descriptions, as illustrated in this example of a TTL device. The individual parts can be placed independently, anywhere in the project sheets

Connectivity

An additional major feature of the Schematic Editor is the system's use of *connectivity*. Connectivity is the ability of the software to recognize the physical links between objects inside the sheet and the ability to associate the logical connections that exist between various sheets in a multi-sheet design. Connectivity is also used to anchor certain objects together. For example, you can drag connected electrical items (parts, buses, wires, ports, etc) without breaking existing connections. More importantly, connectivity allows the schematic to generate connectivity lists, used for passing design information to the PCB Editor, and for performing electrical rule checks.

The process of placing electrical objects in the sheet is often referred to as *wiring*. This is because the connectivity features allow you to work with electrical objects as though you were physically hooking-up the circuit.

Basic electrical objects used in wiring your schematic include: special connective lines that carry signals or power between components, called *wires*; *buses*, that graphically represent grouped nets; *bus entries* that graphically attach wires to buses; *junctions* that connect tangent or crossed wires; and *parts* that represent the component devices and their pins.

Two other special classes of electrical objects are provided as well. The first is *Directives*, which are used to indicate unconnected pins (*No ERC*), and PCB layout attributes on individual nets. The second class is *net identifiers*, which are used to indicate electrical connections that are not physically wired together, for example, connections that continue from one schematic sheet to another in a multi-sheet design.

Net Identifiers

As mentioned above, net identifier objects can create connections that are not physically joined by wires. These objects include: *net labels* that identify common nets on a sheet (or globally, across multiple sheets if you specify); *ports* that identify net connections between two sheets; *sheet entries* that identify net connections into a sub-sheet (referenced by a sheet symbol); and *power ports* which are special symbols placed to represent a global power (or another user-specified) net.

Hidden pins on parts are the fifth type of net identifier. Hidden pins function similarly to power ports. Each hidden pin is connected to all other hidden pins with the same name, and also connected to a net of the same name, if present. If “un-hidden” these pins are not automatically connected and must be manually wired.

Using Connectivity

Methods

Connectivity is derived from the placement of certain electrical objects in the sheet, and from the placement of net identifiers. However, not all electrical objects use placement to define connective behavior. Some objects use their physical geometry to establish connections. Other objects include logical connectivity in their behavior.

Physical Connectivity

Physical connectivity is derived by placing the “hot spots” of any two electrical objects so they are in physical contact. In the simplest example, a wire touching a component pin is deemed to be connected to that pin, and the Schematic Editor can extract that logical connection from the physical contact between the two items.

In general terms, when the hot spots of any two connective objects “touch” they are deemed to be connected. However, there are some special rules that apply to certain classes of connections.

Logical Connectivity

Logical connectivity depends upon the presence of net identifiers (net labels, ports, sheet entries, power ports and hidden pins) on the sheet. Logical connectivity does not require special placement or physical contact but relies on the matching of the net names that associate these objects within a single sheet or across multiple sheets in a project. Refer to the chapters *Multi Sheet Designs and Project Management* and *Creating a Netlist* for further information about using logical connectivity.

General Rule for Connectivity

Electrical objects are connected when their electrical “hot spots” are touching. Special cases of connectivity are described below. When the electrical grid is enabled (**Design » Options**) the cursor will jump to the nearest hot spot and change into a “dot” shape. When placing or moving electrical objects, clicking LEFT MOUSE (or releasing LEFT MOUSE when dragging) will establish a connection when the hot spot is displayed.

Special Rules for Connectivity

Wire to Wire

Wires whose ends touch at any angle, butt end-to-end or have co-linear (overlapping) terminations are deemed to be connected. Co-linear wires that terminate elsewhere on the sheet are not deemed to be connected. Wires that cross or terminate perpendicularly are not deemed to be connected unless a junction is placed at their intersection.

Wire to Bus

Buses are *graphical* representations of grouped signals, and do not have any special connective properties for netlisting. Although buses display hot spots when wiring and

maintain connections during drags, they do not simulate electrical connections. Wires are graphically connected to buses using bus entry symbols. Net labels must be used to indicate logical connectivity on either side of the bus connection.

Wire to Pin

Pins that touch the ends of wires at any angle are deemed to be connected. Pins that intersect perpendicular wires must be connected by placing a junction at that location. Junctions will be automatically inserted where wires cross pins perpendicularly when this option is active (**Tools » Preferences** menu item).

Wire to Port

A Wire touching the end of a port is deemed to be connected.

Wire to Sheet Entry

A Wire touching the end of a sheet entry is deemed to be connected.

Bus to Object

Buses are graphical representations of grouped nets only and have no special physical properties for netlisting. Logical connectivity (use of net identifier, e.g. net label and port) is used in these cases to indicate connections on either side of a bus (See “Net label to bus, below). Note however, if a bus is connecting to a port the bus line must end on the end of the port.

Net Label to Wire

Net labels associate a wire with single net. To achieve this association, the net label must be placed on the same grid point as the wire, either vertically or horizontally. Labels can only be placed on horizontal or vertical lines or at line vertices.

Net Label to Bus

Buses are graphical entities and do not provide physical connectivity for netlisting. Logical connectivity for buses can be assigned by placing a net label on the bus. Generally, this net label will include all bus signals, e.g. HA[0..19] represents nets named HA0, HA1, HA2, etc, to HA19. Buses and bus entries do not highlight when the **Edit » Select » Net** process launcher is used.

Pin to Object

Pins connect directly to other pins, wires, net labels, sheet entries or ports. Hidden pins can be assigned directly to nets in the Schematic Library Editor. Un-hidden pins can also connect directly to other sheets, when that sheet is named in the part’s Sheet Path field.

Pin to Pin

Pins are deemed to be connected if they are in contact at any angle.

No ERC

No ERC objects are deemed to be connected to pins or wires if they are in contact.

Design verification

Design verification is a general term for validating the physical (or electrical) and logical connections in your design. A number of tools are provided that allow you to perform design verification from within the Schematic Editor, by generating reports and by running the Electrical Rule Check (ERC) feature.

Electrical Rule Check report

The Electrical Rule Check (ERC) report is a listing of electrical (and certain logical) violations and warnings for the current active project. A wide variety of basic electrical errors are reported. Errors that can be reported include instances of open input pins on parts or “shorts” between two different nets. You can select the specific rules for a project. “Suppress ERC” symbols can be placed on intentionally unused pins or at other locations where you prefer violations to be ignored by the ERC system.

You can specify either an “error” or “warning” using a graphical matrix of pin, port and sheet entry conditions.

The way net identifiers are used in the design can impact the electrical validity of connections. For example, *global* net labels (Net Labels and Ports Global option) will join all nets with the same label across multiple sheets, where *local* net labels (Only Ports Global option) will only join electrical items within a single sheet. You can specify the “scope” of these net identifiers for electrical rule checks, the same way that they are specified for generating a valid netlist.

Special symbols are overlaid on the sheet, indicating the location of the reported conditions when you specify Add Error Markers. These symbols are cleared from the sheet when the error condition is corrected.

♦ Running an ERC is integral part of the schematic capture phase. Carefully check and resolve all reported errors prior to passing the design information to the PCB Editor.

Checking Sheets and Projects

While creating a schematic, a number of useful design verification features are available directly from the sheet workspace. The **Edit » Select » Net** and **Edit » Select » Connection** menu items can be used to highlight all objects associated with a net or a single connection (component pins do not select in these cases). These process launchers are useful in verifying connections. Once you have used either of these, select **Reports » Selected Pins** from the menus to display a list of all pins connected to the current selection.

Linking from the Schematic to the PCB Layout

Schematic-to-PCB Synchronization

Protel 99 SE has a powerful design synchronization feature, which can automatically match the schematic design data with the PCB design data. Previously, schematic design changes were carried forward using a technique called *forward annotation*, and PCB re-annotation information was taken back to the schematic using *backward annotation*. Both of these functions are carried out by the synchronizer.

When you run the synchronizer it separately examines the component and connectivity information in both the schematic sheets, and the PCB, and then updates one to match the other. When you run it you select either **Update PCB from Schematic**, or **Update Schematic from PCB** from the menus, nominating which is the source, and which is the target.

Including PCB Layout Information on the Schematic

The PCB Layout directive allows you to specify routing topology, priority, width and via size on a net-by-net basis on the schematic. This information is automatically transferred to the PCB if the synchronizer is configured to create Design Rules from Layout Directives (select **Design » Synchronizer Options**).

Cross Probing between the Schematic and PCB

If the PCB design is open while you are working on the schematic, you can move back-and-forth between matching objects in the designs by *cross probing*. Cross probing is bi-directional.

For example, after clicking on the cross probe button you can select a part in the schematic, and the PCB Editor will display the corresponding component on the PCB. Schematic pin-to-PCB pad cross probing and net label-to-physical net cross probing are also supported.

Setting up the Schematic Workspace

Environment Preferences

The Schematic Editor environment preferences are configured by selecting **Tools » Preferences** from the menus to pop up the Preferences dialog. The Preferences dialog is divided into three Tabs.

Schematic Tab

Pin Options

The Pin Options allow the pin name and number to be moved. The number specified is the distance from the end of the pin (the end closest to the component body). The units are hundredths of an inch.

Auto-Junction

Auto-Junction can be enabled and disabled here. Auto junction will automatically place a junction when you terminate a wire onto another wire.

Drag Orthogonal

When you drag components, the wiring is keep orthogonal (corners at 90 degrees). Turning Drag Orthogonal off allows the wires to move at any angle.

Multipart Suffix

Multipart components can use either a numeric or alpha part identifier suffix, for example U1:1, U1:2, etc, or U1A, U1B, etc. Note that this is an environment setting, it applies to all currently open sheets.

Default Power Object Names

These fields can be used to pre-assign a net name to these 3 styles of Power Port.

Orcad® Load Options

The Copy Footprint From / To is used to map the Orcad part field that contains the footprint to the schematic footprint field. The Orcad ports option prevents ports from being resized in the Schematic Editor, important if the design has to go back to Orcad (which does not support resizing ports).

Default Template File

Select a Default Template File to specify what sheet template is used when you create a new schematic sheet.

Graphical Editing Tab

Clipboard Reference

If this option is on when you do an **Edit » Copy** or an **Edit » Cut** you will be asked to select a reference point. This is useful when copying a section of circuit which is to be pasted back into a schematic sheet. This reference point will be the point where the section of circuit will be held when pasting.

Add Template to Clipboard

The sheet template is also copied to the clipboard when you Copy or Cut.

Convert Special Strings

Enable this option to see the contents of the special strings on screen, as they will be printed. For more information on using special strings refer to the *Sheet Templates* topic in the *Setting up the Schematic Editor* chapter.

Display Printer Fonts

Not all fonts are supported on all output devices (and Windows will automatically substitute). To see what the text is going to look like on the printout enable this option.

Center of Object

Hold the object being moved or dragged by its reference point (for objects that have one, such as library components or ports), or its center (for objects which do not have a reference point such as a rectangle).

Object's Electrical Hot Spot

Hold the object being moved or dragged by the nearest electrical hot spot (eg, the end of a pin) when moving or dragging.

Auto Zoom

Automatically zoom in when jumping to a component. Zoom level remains as it was if this option is not enabled.

Single '\ Negation

If this option is enabled a net name can be negated by typing a backslash character before the first letter in the net name. This applies to ports, net labels and sheet entries.

Undo Stack Size

You can undo this many edits. Set as required (remember, these undo-able edits are being held in memory).

Default Primitives Tab

The default attributes for every object that can be placed in the Schematic Sheet Editor can be defined here. Typical changes that you might make to the Default Primitives include: changing fonts, such as Designators, Part Types and so on; or changing the color of objects such as Wires and Text. User defined default values are held in the file ADVSCH.DFT. User DFT files can be created and loaded. Use the permanent check box to prevent these defaults being altered.

Schematic Document Options

The Schematic Editor gives you extensive control over the schematic sheet size and style, including customized sheets and re-usable sheet templates. These sheet controls allow you to easily create and use sheets customized for specific organizations, or purposes.

To set the current schematic sheet options select **Design » Options**. This will pop up the Document Options dialog.

Sheet Options

The Document Options dialog provides control over the attributes of a sheet including; the sheet style and sheet options, the grids, organization information and access to the system font.

Setting the Schematic Sheet Size

You can choose from 10 standard imperial and metric sheet sizes, or to define a custom sheet size. The maximum custom sheet size is 65 inches by 65 inches.

Schematic sheets are conventionally displayed and printed in “landscape” (wide) rather than “portrait” (tall) orientation. The Schematic Editor allows you to display and print your drawings in either orientation. Standard sheet sizes include:

<i>SIZE</i>	<i>WIDTH X HEIGHT (IN)</i>		<i>WIDTH X HEIGHT (MM)</i>	
A	11.00	8.50	279	216
B	17.00	11.00	432	279
C	22.00	17.00	559	432
D	34.00	22.00	864	559
E	44.00	34.00	1078	864
A4	11.69	8.27	297	210
A3	16.54	11.69	420	297
A2	23.39	16.54	594	420
A1	33.07	23.39	840	594
A0	46.80	33.07	1188	840
ORCAD A	9.90	7.90	251	200
ORCAD B	15.40	9.90	391	251
ORCAD C	20.60	15.60	523	396
ORCAD D	32.60	20.60	828	523
ORCAD E	42.80	32.80	1087	833
LETTER	11.00	8.50	279	216
LEGAL	14.00	8.50	356	216
TABLOID	17.00	11.00	432	279

The maximum available work area in a sheet (with the border hidden) will depend upon the output device. Many printers and plotters cannot print to the edge of the sheet, so some trial and error may be necessary to determine the exact available work area. Because of this, the standard ANSI and ISO border specifications cannot be applied

when targeting these devices. The Schematic Editor can compensate for this by allowing you to scale the output during printing or plotting.

The available work area will be smaller when sheet borders are displayed. The default sheet border removes 0.2 to 0.4 inch (approximately 5 to 10 mm) from the working area, depending upon the sheet size selected.

Template

Name of the template that this sheet is based on. Refer to the *Sheet Templates* topic later in this chapter for more information on creating and using sheet templates.

Borders

When defining sheet borders, you should be aware that not all devices can print all the way to the edge of the page. For example, laser printers typically reserve a margin of about 0.15 inches (4.0 mm) outside the printable area. This can make it impossible to include all of the standard border when printing at 100% scale using standard sheet sizes, such as “A” or “A4.” You can change the print scale to accommodate the maximum printable area of your printer.

Title block

Protel provides two pre-defined title block formats. Choose the default title block or an ANSI standard title block that is somewhat larger. Some of the information in title blocks is provided automatically, e.g., the sheet size, file name and creation date. Turn the title block off if you wish to draw your own.

Snap Grid

The snap grid is the grid that the cursor is locked to when placing or manipulating objects on the sheet. This grid should be left on at all times except when specifically placing or moving objects that need to be off grid, like text.

Visible Grid

The visible grid is the grid you see on the sheet, which acts as a visual reference. Typically it is set to be the same as or a multiple of the snap grid.

Electrical Grid

The electrical grid supports the Schematic Editor’s *Guided Wiring* feature. Imagine you are moving an electrical object in the workspace. When it falls within the electrical grid range of another electrical object that you could connect to, the object you are moving will snap to the fixed object, and a *Hot Spot* or highlight dot will appear. This dot *guides* you as to where a valid connection can be made. The electrical grid should be set slightly lower than the current snap grid or else it becomes difficult to position electrical objects one snap grid apart.

Organization

Click on the Organization Tab in the Document Options dialog to enter the organization details. Each field here is linked to a *Special String*. Refer to the *Sheet*

Templates topic later in this chapter for an explanation and example of how to use special strings.

Sheet Units

Both the Schematic Sheet Editor and Library Editor have a resolution of 0.01", or one hundredth of an inch. The units displayed on the left of the status bar are always hundredths of an inch, regardless of the sheet style.

Changing the System Font

The System font includes; border text, system title blocks, pin names, pin numbers, ports, power ports and sheet entries. For more information refer to the *Fonts* topic in the *Schematic Design Objects* chapter.

Sheet Templates

The sheet border, title block and included graphics make up what is referred to as the sheet template. The Schematic Editor is supplied with a number of sheet templates, one for each size of sheet available.

User defined sheet templates can be created. They are created in the same way you create a normal schematic sheet. After adding all the objects to the sheet select **File » Save Copy As** to save the sheet as a *.DOT template file. Once created, these pre-defined templates can be applied to new or existing projects.

As well as including custom title blocks and graphics, sheet templates can include special strings to automatically add document text when printing or plotting. Refer to the Activity on the following page for information on how to create a template.

Special Strings

Special strings are text strings which are recognized by the Schematic Editor and interpreted when the sheet is printed or plotted. Each special string either links to a field in the Organization Tab of the Document Options dialog, such as .TITLE, or provides current information, such as .DATE. Special text strings can be placed either on a sheet template, or directly on a schematic sheet.

◆ Use special strings with your templates to provide quick and consistent document text.

By placing the special strings on your sheet template you will not need to accurately place text each time you do a new design. You simply go to the Document Options dialog for that sheet and fill in the fields. When the sheet is printed, each special string will be replaced by the text you entered into the appropriate field of the Document Options dialog. The string will be placed at the location of the "." (the dot, or full stop).

If you wish to see the text that was entered in the Document Options dialog on the screen, rather than waiting till it is printed, check the Convert Special Strings check box in the Graphical Edit Tab of the Preferences dialog (**Tools » Preferences**).

The Special string that are linked to the Document Options dialog are:

.ORGANIZATION	Lists Organization field text.
.ADDRESS1	Lists text from first Address field.
.ADDRESS2	Lists text from second Address field.
.ADDRESS3	Lists text from third Address field.
.ADDRESS4	Lists text from fourth Address field.
.SHEETNUMBER	Lists text from Sheet No. field.
.SHEETTOTAL	Lists text from Sheet Total field.
.TITLE	Lists the Document Title text.
.DOCUMENTNUMBER	Lists the Document No. text.
.REVISION	List the Document Revision text.

The following special strings automatically insert the current information at the time the document is printed.

.DOC_FILE_NAME	The name of the schematic sheet file.
.DOC_FILE_NAME_NO_PATH	Name of sheet, without the path.
.TIME	The current time.
.DATE	The current date.

To place a special string select **Place » Annotation** from the menus and type in the special string, including the dot.

Activity - Creating a Custom Template

Protel templates are stored in the \Program Files\Design Explorer 99 SE\System\Templates.ddb database. You can add your template to this database, create your own templates database, or include the template in the design.

Follow these steps to define a custom schematic template:

1. After opening the database and the folder that the template will be stored in, select **File » New** to create an empty schematic sheet, then double-click to open this sheet.
2. Choose **Design » Options** from the menus, then click on the Sheet Options Tab.
3. Select the sheet size in the Standard Styles pull down list.
4. Un-check the Title Block option (to remove the standard title block) and click OK.

Notice that the standard title block no longer appears on the page. Zoom-in to the bottom right corner of the page to start a custom title block (zoom in shortcut: position the cursor where you wish to zoom and press PAGE UP) .

You are ready to draw a new title block.

5. Choose the graphical line tool from the Drawing Tools toolbar, or select the **Place » Drawing Tools » Line** menu item. A cross-hair will appear on the cursor.
6. Before starting the line, press the TAB key to set the attributes of the line. The Line dialog will open.
7. In the Line dialog click in the Color box to open the Color Selector, then scroll up to color number 4 (black). Click OK to close the Color Selector dialog .
8. You will be back in the Line dialog. Set the Line Width to Smallest.
9. Click OK in the Line dialog to accept these changes.
10. Now, position the cursor in the sheet workspace and click to begin the first title block line segment.
11. Move the cursor to the point where you wish to define the first corner. Click to define this corner, then move the cursor to the location of the second corner. Continue to do this until you need to start drawing a new line.
12. Click RIGHT MOUSE once (or the ESC key) when you want to end this multi-segment line. You can then move the cursor to a new location to start a new line. Click RIGHT MOUSE a second time (or the ESC key again) to exit the Place Line command.
13. You are ready to place text in the title block.
14. Choose **Place » Annotation** from the menus.
15. Before placing the text, press the TAB key to change the text attributes.
16. Press the Font Change button.
17. In the Size field, type *16* and Click OK.
18. In the Text field type .TITLE and then click OK to close the dialog. Note the full stop character immediately in front of the word TITLE, this must be included.
19. Position the cursor in the appropriate region of your new title block, then click LEFT MOUSE. To temporarily disable the snap grid hold the CTRL key as you position the text.

The .TITLE special string is mapped to the Title field in the Document Options dialog. Refer to the *Special Strings* topic earlier in this section for an explanation of special strings and a list of all the special strings. Continue to define your custom title block by adding the appropriate special strings as follows:
20. Press TAB to pop up the Annotation dialog again. Enter the .DOCUMENTNUMBER special string and set the font to an appropriate size. Position the cursor where you wish to place this special string and click LEFT MOUSE. Continue to place the special strings in the appropriate regions of your title block.
21. Press ESC to exit the place text annotation command.

22. Graphics can be included in the template. Select the **Place » Drawing Tools » Graphic** menu item to add a graphic. When the Image file dialog appears select the graphic image file and click OK. When the cursor appears you will need to click once to define the top left of the image location, then a second time to define the bottom right of the image location.

You are now ready to save this sheet as a template:

23. Choose **File » Save As** from the menus.
24. Type a name for the template in the File Name field, and set the Save As Type option to Schematic template binary (*.dot).
25. The extension .DOT defines this file as a sheet template. This template can now be used for new or existing designs. Close the template file when you have finished.

◆ When you place a graphic the image is not saved with the document, only a pointer to the graphic file. If you move the design to another PC the graphic file will need to move too.

Setting a Preferred Template to be Automatically Used

You can specify a template to be used automatically whenever a new file is created. To do this select the **Tools » Preferences** menu item. In the Preferences dialog, at the bottom of the Schematic Tab, press the Browse button to display the Select dialog. Select the database that contains your template in the drop-down list at the top of the dialog, if the database is not available in the list click the Add button and browse to locate it. Locate and select your template in the database that you selected.

When you click OK and return to the Preferences dialog, your template name will appear in the Default Template File field. Click OK to close the dialog. When you select **File » New**, the new sheet will use your template. Note that the objects that make up the template cannot be edited now, any changes have to be made to the template itself.

Updating Existing Templates

Templates can also be applied to the active sheet, or all open sheets, at any time. The Design menu has three menu items for working with templates:

- **Update Current Template** - use this if you have modified a template and need to “re-fresh” the sheets which use it. You will be asked if you wish to update the template for all currently open files. Click No if you wish to only re-fresh the active sheet.
- **Set Template File Name** - this removes the existing template and uses the one you choose.
- **Remove Template** - removes the template (but retains the sheet size from the old template).

Working in the Schematic Editing Window

Protel 99 SE's schematic server includes two document editors, the Schematic Sheet Editor and the Schematic Library Editor. Working in either of these document editors is quite similar, you build your design from the set of objects provided, placing these objects on a sheet. The strategies used for placing objects, editing their attributes, positioning and deleting them on the sheet, and so on, are common to both editors. In essence, the way you work in either editor is the same, what you do in each editor is different.

In the Schematic Sheet Editor you create, edit and validate schematics. In the Library Editor you create, edit and validate components and component libraries.

Changing Your View of the Sheet

Each sheet you open will appear on its own Tab in the integrated Design Window. You look “through” this window to view your sheet. You can bring the sheet closer to you (zoom in), or move the sheet away (zoom out). The View menu provides a number of ways of changing your view of the document, including a Fit All Objects option, where the sheet will be zoomed to fit all the placed objects, and a Fit Document, where the entire sheet will be displayed. Shortcut keys to change your view include:

- **PageUp** to zoom in
- **PageDown** to zoom out
- **Home** to re-center the screen at the current cursor position
- **End** to refresh the screen

Moving Around the Schematic

Panning the Schematic

If the sheet is zoomed such that you can not see the entire sheet, scroll bars will appear, allowing you to scroll around the sheet. These scroll bars have a sliding button, which you can click and drag to scroll up and down, or left and right across your sheet. The position of the sliding buttons gives an indication of what portion of the sheet you are currently viewing. Click above or below the sliding button to scroll across the sheet in large steps, or click on the arrows at each end of the scroll bars to scroll across in small steps. When the sheet is zoomed such that you can see the entire sheet, the scroll bars are removed.

Scroll bars provide one way of moving around the sheet. The other way to move around the sheet is autpanning. Autpanning is enabled whenever you have a cross-

hair cursor. You have a cross-hair cursor whenever you perform an “edit” type operation – like placing, selecting, moving or deleting objects.

This cursor can be moved either by moving the mouse, or pressing the arrow keys on the keyboard. If the cursor is moved such that it hits the window frame, you will pan across the sheet. To autopan at higher speed hold the SHIFT key while panning. The speed at which you autopan can be altered by changing the Speed setting in the Graphical Editing Tab in the Preferences dialog (**Tools » Preferences**). Here you can also change the style of autopanning as well, choosing either a Fixed Size Jump, where the sheet moves over by the current step size, or Re-Center where the sheet shifts in half screen increments, and re-centers the cursor.

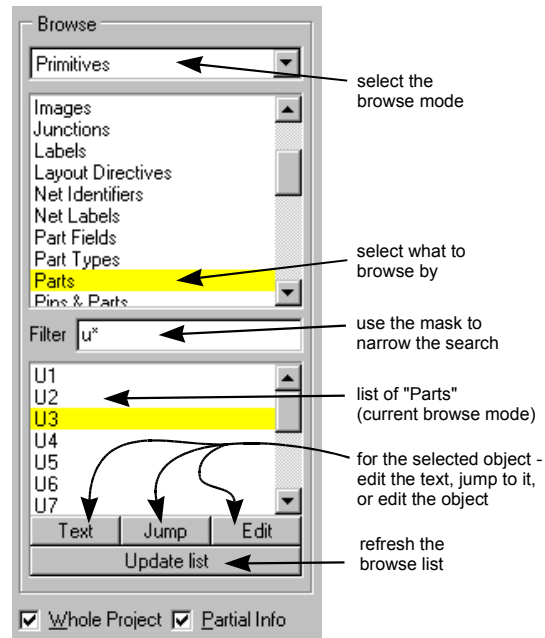
Browsing the Schematic

When your design has many sheets it can become difficult and time consuming to locate objects on the sheets. To simplify this process the Schematic Editor includes powerful browsing features.

Set the Browse mode at the top of the Editor Panel to Primitives, to browse through objects on the current sheet, or the entire project.

You can browse by any of the object types available in the Schematic Editor.

The bottom of the browser has three buttons, Text, Jump and Edit. Select an item from the browse list before using these buttons.



- Press Jump to jump to the selected object, centering it in the window.
- Press Text to jump to the selected object and pop up a text editing dialog if the object has a text field.
- Press Edit to jump to the selected object and pop up the object’s attribute dialog, allowing you to edit any of the objects attributes.
- Use the Update List button to refresh the browse list prior to browsing.
- When you select an item in the list, information about that item will appear on the Status Bar.

Jumping around the Schematic

Another way to move quickly around the current schematic sheet is to Jump. Supported Jump options include; the origin, a specific location, one of ten markers, or an Error Marker. This can save you from having to constantly zoom in and out to navigate around your design, and is particularly useful for large or complex layouts.

Jump » Origin

Jumps to the absolute (0,0) coordinate. In the Schematic Sheet Editor this is the lower-left corner of the sheet. In the Library Editor it is the center of the sheet.

New Location

This option allows you to type in the desired coordinates for the jump.

Using Location Markers

The Schematic Editor includes ten user-definable location markers, allowing you to “mark” positions on a sheet, which you can then quickly jump to with just two key strokes. These markers can be placed anywhere on a sheet by selecting **Edit » Set Location Marks**.

Jumping to Location Markers

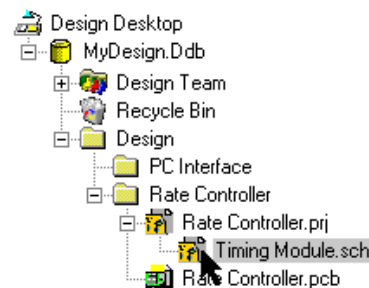
To jump to any of the pre-defined location markers on the current sheet select **Edit » Jump » Location Mark X** (shortcut: **J, 1** or **J, 2** etc).

Using the Navigation Panel to Move through the Project

The Design Explorer Navigation Panel displays all the documents in your Design Database.

As well as displaying how the various documents are organized in the Design Database, it also shows any logical parent-child relationship between schematic sheets.

Use the Navigation Panel to navigate through the sheets of your project. Click on the small “+” sign to display the sheet(s) below a sheet, click on the small “-” sign to hide the sheet(s) below. Click once on a sheet to open it, and make it the active document in the Design Window.



Another technique that can be used to navigate through a project is the Up/Down Hierarchy button on the Main toolbar.



For a complete explanation of how to use the Up/Down Hierarchy button refer to the topic *Tools for Working with Hierarchy* in the chapter *Multi-sheet Design and Project Management*.

Placing Schematic Objects

You will find the way you place and manipulate objects in the Schematic Editor to be quite simple and straight forward. The placement techniques are consistent with other Windows graphical editing tools; select a tool, click the mouse to define a location, move the mouse to define the next location, and so on.

As you experiment with each of the object types refer to the *Primitives* topics in the *Schematic Design Objects* chapter for tips on placing each object.

Each object can be placed using menus, toolbars or shortcut keys. The approach used to place objects is consistent for all objects. An example of placing wires is given below. Placing components is covered in the *Components and Libraries* chapter.

Activity - Placing a Wire

To place a wire in the current sheet window:

1. Zoom in on the sheet (press **PageUp** until you can clearly see the grid).
2. Select the **Place » Wire** menu item (shortcut keys: P, W, or click the Wire button on the WiringTools toolbar).

Note how a cross-hair appears on the end of the pointer-style cursor. The cross-hair is the editing cursor, it is used each time you perform any editing type operation (select **Tools » Preferences** to change the cursor style).

3. Click LEFT MOUSE (or press ENTER) to define a start point for the wire.

As you move the cursor, note that its position is constrained by a grid and that the cursor jumps to the nearest grid point. The cursor is snapping to the Snap Grid. Select **Design » Options** to change the Snap Grid.

4. Drag the wire segment in any direction. Click LEFT MOUSE (or press ENTER) to end this first segment of the wire.
5. Move the cursor to continue with a new wire segment, extending from the existing segment. Click LEFT MOUSE or press ENTER again to define the second segment. Continue like this until the wire is finished.
6. When the wire is finished click RIGHT MOUSE to end the segmented wire.

◆ Press the BACKSPACE key during wire placement to remove the last corner (vertex).

Note that cursor is still a cross-hair, indicating that you are still in the place wire mode. This allows you to end one wire and then begin a new series of wire segments elsewhere in the workspace, without having to select **Place » Wire** again.

◆ Refer to the topic, *Editing Schematic Objects*, later in this chapter for tips on how to modify a placed wire.

7. To exit wire placement, press ESC (or click RIGHT MOUSE a second time). Note that the cross-hair disappears from the end of the cursor.

Manual and Auto Wire Placement Modes

The Schematic Editor provides six wire and bus placement modes. Press the SPACEBAR as you place a wire or bus to toggle between the different modes. Options include two orthogonal modes, an any angle mode, and an Auto Wire mode.

Any Angle Mode

Allows the wire to be placed at any angle.

90/90 Line Mode

Constrains the wire placement to horizontal or vertical orientation. There are two modes; one keeps the shorter of the two segments attached to the cursor, the other keeps the longer of the two segments attached to the cursor.

45/90 Line Mode

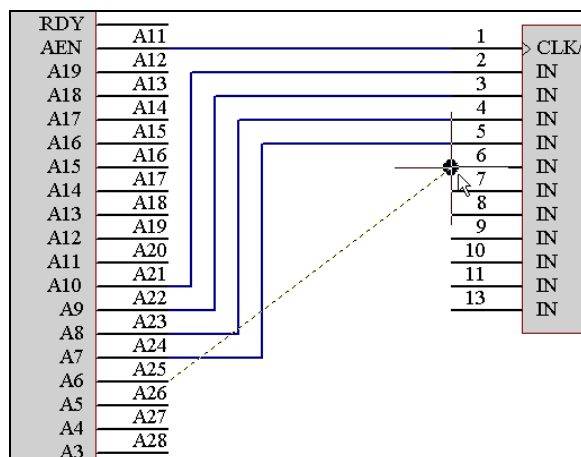
Constrains wire placement to 0, 45, 90, 135, 180, 225, 270 or 315 degree orientation. There are two modes; one keeps the straight segment with the cursor, the other mode keeps the 45 degree line with the cursor.

Auto Wire Mode

The Auto Wire mode will attempt to route a wire, in an orthogonal manner, from the first point you click, to the next point you click. These click points are not restricted to electrical hot spots (such as pins), they can be anywhere on the sheet. To modify the Auto Wire parameters press the TAB key during wire placement.

◆ To make it easy to tell when you are in the Auto Wire mode, the connective line is shown as a dotted line.

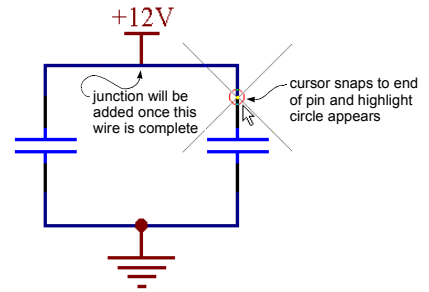
After placing the first wire from pin A21 to pin 2, the remaining seven wires in the bus can be quickly placed using the Auto Wire mode. Simply click on the start pin (pin A25 in the figure), click on the end pin (pin 6), then right mouse click to finish this wire.



Guided Wiring and Auto-Junction

To speed the process of wiring the Schematic Editor includes two powerful productivity features; guided wiring and automatic junctions.

As you are placing a wire try moving it towards an existing electrical object, such as a wire or a component pin. When the wire falls within the *electrical grid* range of the other electrical object, the cursor will snap to the fixed object and a *Hot Spot* (highlight circle) will appear. The *Hot Spot guides* you as to where a valid connection can be made.



Guided wiring speeds the process of getting the cursor to the correct location to terminate the wire – allowing you to wire quickly, and at much lower zoom levels. Once the cursor is in the correct location you simply click to terminate the wire.

There is no need to worry if a junction is required when you click, the Auto-Junction feature will automatically add one if it is required. Auto-Junction places a junction when two wires are connected in a T-type fashion, or a wire connects orthogonally to a pin or power port.

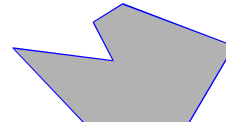
♦ Always set the electrical grid slightly lower than the current snap grid, otherwise you will find it difficult to position electrical objects one snap grid apart.

Creating 2-Dimensional Graphical Objects

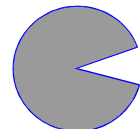
2-Dimensional objects, such as rectangles and polygons are created and placed similarly to wires, buses and lines. Each object may require a different number of mouse clicks to define it, refer to the *Primitives* topics in the *Schematic Design Objects chapter* for more information.



rectangle



polygon



pie



rounded rectangle



ellipse



graphical image

Note that graphical images are not stored in the schematic file

- only a link to the file is saved with the sheet. If you move your design to a different directory you will need to change the link (double-click on the image to edit the link).

Editing Schematic Objects

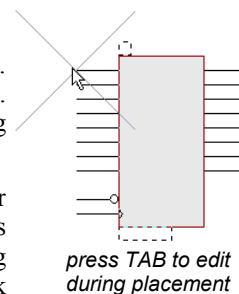
There are two approaches to editing an object in the Schematic Editor, either by changing its attributes in its Change dialog box, or graphically modifying the object. Certain operations (such as changing the color of a primitive) can only be performed by editing the attributes, others (such as re-sizing a rectangle) can be done through the dialog box or by graphically modifying the object.

To edit the look of an object in the workspace it is generally easier to do it graphically. To do this you must first bring the object into *focus*. To edit the object through its Change dialog box select the Edit-Change menu item and then click on the object.

Editing While Placing

You can edit the attributes of an object while it is being placed. While the object is floating on the cursor, press the TAB key. This pops up the object's dialog. The advantages of editing during placement are:

- Changes made during placement can become the defaults for that type of object. These changes are stored in the defaults file (ADVSCHE.DFT). Note that this method of setting defaults depends on the setting of the "Permanent" check box in the Preferences dialog box (**Tools » Preferences**). If this option is on the changes being made will not become the defaults, they will only apply to the object currently floating on the cursor.
- Objects that have a numeric identifier, such as a designator, will auto-increment.
- There is no need to edit the object after it has been placed, speeding the design process.



Editing an Object on the Sheet

To change any placed object select **Edit » Change** from the menus, move the cursor over the item and click LEFT MOUSE (shortcut: double-click LEFT MOUSE). Each object has its own range of editable attributes. You can change just this object, or extend changes across your entire design using powerful global editing options.

◆ Refer to the *Global Editing* topic, in the *Working in the Schematic Editor* chapter, for tips on Global editing.

On-Sheet Text Editing

Text strings can be edited directly on the schematic sheet. Click once to focus a text string, click a second time to edit the string directly on the schematic sheet. This behavior can be disabled by turning off the Enable In-place Editing option in the Preferences dialog.

Editing Graphically - Focus and Selection

One of the advantages of a graphical based editing environment is the ability to make changes directly to objects displayed on the screen. To be able to graphically edit an object, or a group of objects, you must first identify the objects. This is done through *focus*, or *selection*.

In other Windows applications selection is a single concept, the process of choosing objects before being able to modify them. A typical example would be selecting one or more objects to be *copied* to the clipboard and then *pasted* to another location. Often selected objects can be modified directly. For example, selected objects can be moved or re-shaped in most graphical applications.

Unlike other Windows applications, the Schematic Editor uses two independent methods for accomplishing selection oriented tasks. These methods, *focus* and *selection*, are used repeatedly when creating or editing your Schematic. Breaking selection into these two independent processes allows the Schematic Editor to perform complex modifications of objects which would be either difficult or impossible using the simple selection method described above.

Focus

When you position the cursor over a design object in the Schematic Editor and click LEFT MOUSE this object then “has the focus”, and the way it is displayed changes. This is similar to the way you can change the focus in Windows by clicking on an open window to make it active.

♦ To move the focus to another object click on that object. To have nothing in focus click in a clear area of the workspace.

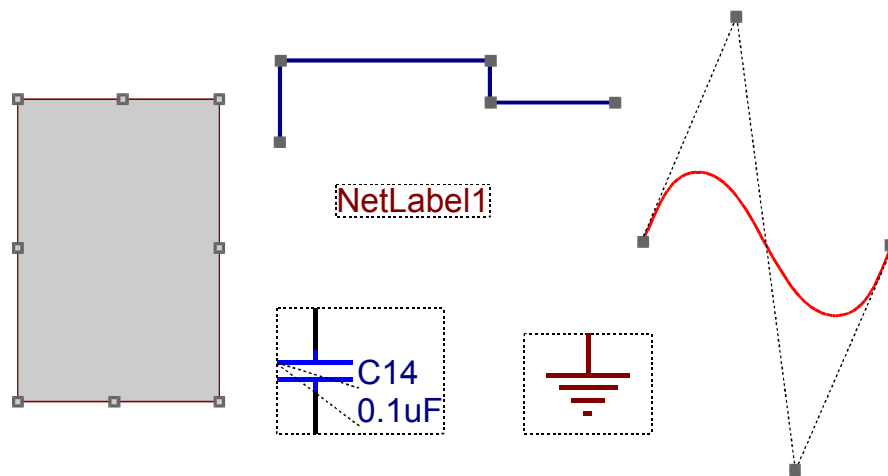
Only one object can be in focus at a time. You can tell which object is currently in focus because its graphical editing handles (or a focus box) are displayed. For example, if you click LEFT MOUSE on a rectangle a re-sizing handle will appear at each corner and along each side. If you click LEFT MOUSE on a net label a focus box will appear.

Graphically Editing the Focused Object

When an object is in focus, you can move the object, or edit its graphical characteristics.

To graphically change a focused object click once on an editing handle. That point of the object will then become attached to the cursor – simply move the mouse to a new location and click to place.

Click anywhere on a focused object to move it, press the DELETE key to delete it. For strategies about moving or dragging objects refer to the *Moving and Dragging* topic later in this chapter.



Objects in focus, displaying their focus handles or focus box.

Editing Handles and Polyline Behavior

Polyline objects have special graphical editing characteristics. For example, when you place wires, buses or graphical lines, you define a vertex each time the wire, bus or line changes directions. These vertices are displayed as editing handles when the object is in focus. These objects can have complex shapes (hence “polyline”), but can be manipulated (moved, cut, copied, pasted, cleared or deleted) as a single object.

A special feature of polyline objects is the ability to add or delete vertices from a placed object.

Activity - Adding a Vertex or Control Point to a Polyline Object

To add a vertex or control point:

1. Click on the polyline object to place it in focus.
2. Position the cursor over the line segment at the point where you would like to add the new vertex. Click and hold LEFT MOUSE and press the INSERT key. The segment will now “bend” at the cursor.
3. Move the new vertex to the desired location and release LEFT MOUSE to place.

If you find that the cursor jumps to the nearest handle rather than remaining where you clicked, disable the Object’s Electrical Hot Spot option in the Preferences dialog.

Activity - Removing a Vertex or Control Point

Sometimes you will want to remove a vertex (or control point) when reshaping a polyline object. To remove a vertex or control point:

1. Click on the polyline object to place it in focus.

2. Position the cursor over the handle that you wish to delete. Click and hold LEFT MOUSE to grab the handle.
3. Press DELETE to remove the vertex (or control point).

The cursor will jump to the nearest remaining vertex.

Focus Summary

As illustrated in these examples, you must *focus* an object before you can modify it graphically. Note that you cannot use the clipboard menu items: **Edit » Copy**, **Cut**, **Paste** or **Clear** with the focused object. These Clipboard features work only on a *selection*.

Selection

Selection provides a second, distinct method of manipulating objects. Unlike focus, selection can be used with both individual objects and with a group of objects.

Selection does not display an object's graphical editing handles or a focus box. Instead, the object is outlined in the *selection color* (to change the color select **Tools » Preferences**).

◆ You must *select* objects before you can use the clipboard functions: **Edit » Copy**, **Cut**, **Paste** and **Clear**.

Once selected, objects can be moved, cut, copied, pasted into another window or location in the current window, or cleared. Special **Edit » Move** options allow you to move or rotate selections in a single operation. Selection also works with the Schematic Editor's global editing feature, which can limit global changes to selected or un-selected objects.

A key feature of the Schematic Editor's complex selection model is the ability to click LEFT MOUSE without *de-selecting* objects that were previously added to the current selection. This allows you to perform a wide variety of operations without effecting the current selection.

Selections are made in the following ways:

- Direct selection, using SHIFT+LEFT MOUSE to add (or remove) individual items to the current selection.
- The click-and-drag-a-window-around shortcut.
- Use the **Edit » Select** and **Edit » DeSelect** sub-menus to define a selection.

- By using the Selection check box in the object's dialog. This option allows you to use the Schematic Editor's global editing feature to apply selection status changes to other objects in the design. Refer to the *Global Editing* topic later in this chapter for tips on global editing.

If something unexpected happens, select **Edit » Undo from the** menus to restore back to a previous state.

Activity - Directly Selecting an Individual Object

Direct selection is the most flexible way to select an individual object. To select one object at a time:

1. Hold down SHIFT and click LEFT MOUSE with the cursor positioned over the object.

The item will be redrawn, outlined in the Selection color (select **Tools » Preferences** to change the color). You can do this repeatedly, each time adding another item to the current selection.

If you hear a “beep”, or nothing appears to be selected, try zooming in closer (press PAGEUP) and make sure that the cursor is directly over the item you wish to select. To select a component, position the cursor within the component body. Components, especially complex components can take a moment to select.

To add another item to the current selection:

2. Hold down SHIFT and click LEFT MOUSE over another item.

To release individual items from the selection:

3. Hold down SHIFT and click on the selected item.

When an object is de-selected it is redrawn in its original colors. Other selected items remain selected until they are either individually de-selected (SHIFT+LEFT MOUSE) or until an **Edit » DeSelect** is executed (shortcut: X, A).

Activity - Direct Selection of an Area

Direct selection can also be performed on an area. To select all objects within an area:

1. Position the mouse where there are no objects under the cursor.
2. Click and hold the LEFT MOUSE button.

The Status Bar will prompt “Choose Second Corner”.

3. Drag the mouse diagonally away. Define the area with the selection rectangle and release the LEFT MOUSE button.

Only objects that fall entirely with the rectangle will be selected. The selected objects will highlight in the current selection color.

4. Repeat the process to extend this selection.

◆ Objects remain selected until you deselect them – it is good practice to always clear the current selection prior to making a new selection (**Edit » DeSelect All** or shortcut: X, A)

Using the Menu Options to make a Selection

The **Edit » Select** menu items allows you to select all objects inside or outside of an area, or all objects. You can also select by Net (selects all wires and net identifiers for the chosen net on the current sheet), or Connection (selects all wires and net identifiers that are “physically” connected).

Edit » De-Select provides the same options, less the Net and Connection options. Shortcut: press X to pop up the De-Select menu.

The **Edit » Toggle Selection** menu item allows you to toggle the selection state of individual objects, behaving the same as the SHIFT+LEFT MOUSE “direct” selection technique.

Working with a Selection

Objects that are selected can be cut or copied to the clipboard, and from there pasted into other schematic sheets, or into any Windows application that supports the Windows clipboard. Selections can also be deleted by selecting the **Edit » Clear** menu item or the CTRL+DEL shortcut keys.

Use the clipboard in the Schematic Editor the same as you would in any Windows application. The sequence is; select the objects to perform the operation on, cut or copy the selection to the clipboard, then paste the clipboard contents to the desired location.

Notes on Using the Clipboard

- The Schematic Editor includes a Clipboard Reference Location option. A reference location is a coordinate you nominate when you copy or cut the items. When you paste the selection, you will be “holding” it by this reference location, allowing you to accurately position the pasted objects. If this option is enabled you will be prompted to “Choose Clipboard Reference Location” when performing a Copy or Cut. Select **Tools » Preferences** to set this option.
- **Edit » Copy** copies the current selection to the clipboard.
- **Edit » Cut** clears the current selection from the workspace and copies it to the clipboard.
- Select **Edit » Paste** to paste the selection back into any open sheet.
- Make sure that the selection includes only those items you wish to copy or cut. To ensure that nothing is selected before making a new selection use the deselect all shortcut: x, a.
- Use the shortcut SHIFT+left mouse to add or remove items from the current selection.
- The clipboard holds the last selection only, each time you select Cut or Copy you overwrite the clipboard contents.
- Select **Edit » Clear** to delete the current selection from the workspace without copying it to the clipboard (shortcut: CTRL+DELETE).

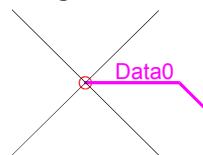
- The Schematic Editor includes an option to include the sheet template when copying to the clipboard. Select **Tools » Preferences** to set this.

Pasting an Array

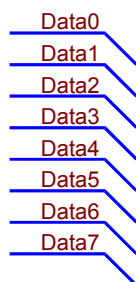
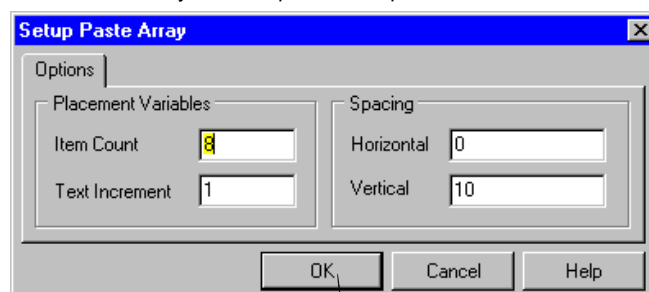
When you use **Edit » Cut** (or **Copy**) you are placing a copy of the current selection in the clipboard. **Edit » Paste Array** allows you to place multiple copies of the clipboard contents back into the workspace. The following figure shows an example of using the array feature to create a set of eight data lines. It is easier to create and place the array if the Clipboard Reference feature is enabled (refer to the previous topic).

Creating a set of eight data lines using the Paste Array feature.

After placing the wire, bus entry and net label, select them, then cut the selection to the clipboard.



Select Paste Array and setup for the required number of data lines.



Click OK to close the dialog, then click to place the array. The array will then appear.

Item Count

The number of repeat placements to be performed. For example, typing 4 will place 4 of the current clipboard contents.

Text Increment

This option is used for text which you want to auto-increment, such as net labels. Setting this to 1 will increment the component designators, for example U1, U2, U3 etc.

Horizontal

The horizontal spacing between each pasted item. Setting this to 10 would place each object on the next standard visible grid point (0.1 inches apart).

Vertical

The vertical spacing between each pasted item. Setting this to 10 would place each object on the next standard visible grid point (0.1 inches apart).

Global Editing

As well as being able to edit the attributes of a single object, you can also apply these edits to other objects of the same type on the current document, or if you wish, across the entire project .

Additionally, you can further define conditions that either extend or restrict global changes. For example, changes can be applied to all objects that are selected or all objects that are not currently selected, or the change can be applied without regard to the object's selection status. If desired, you can create a complex set of conditions for applying changes.

Virtually every one of these editable object attributes can be globally applied. A simple example would be changing the color assigned to all wire segments associated with a specific net. In another instance you may wish to change the font of all net labels. These options (and more) are possible with global editing. The possible applications for global changes are limited only by the imagination of the designer.

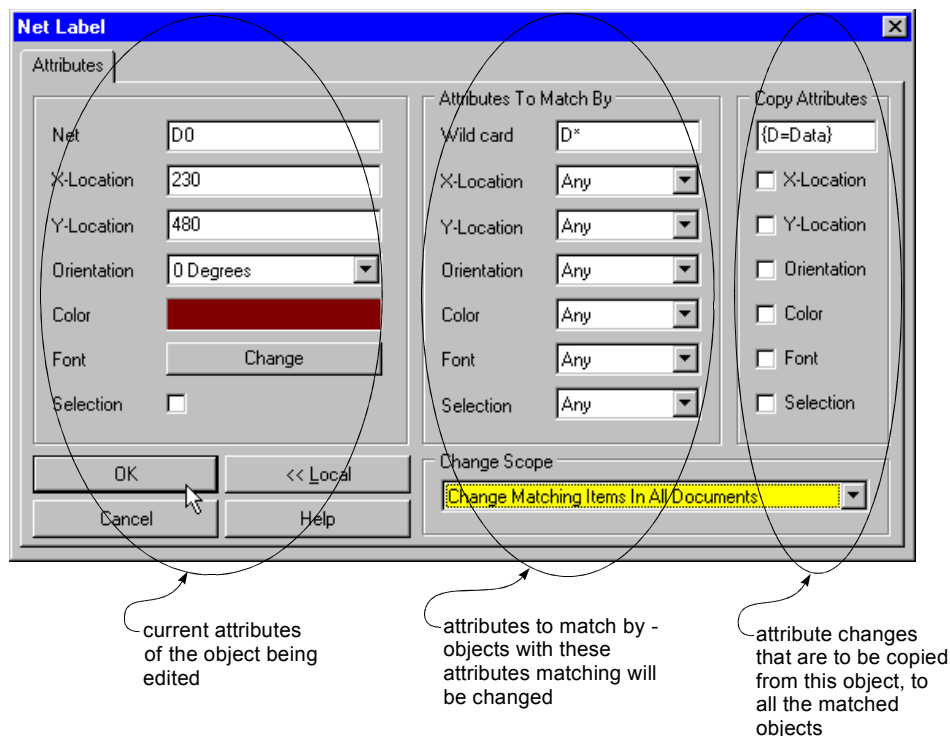
The large number of global change options may make this feature appear somewhat complex at first. However, the principles of applying global changes are reasonably simple once understood. When mastered, this feature can be an important productivity tool that can save a great deal of manual editing of a schematic.

Cross-Project Global Editing

Global editing of schematic objects throughout a multi-sheet, hierarchical project is supported. This feature allows you to change items located in various parts of a project, or impose style changes throughout all open sheets in the current project. Set the Change Scope to control which sheets are affected by the global edit.

Global Editing Strategies

While the presentation of global change options may appear differently in the various object dialogs, the strategy used is always the same. This description will outline the approach to global editing.



Current Attributes

When you double-click on an object, you are presented with the dialog for that type of object. This dialog contains the *current* values or settings of the attributes of that object. Change the attributes you would like to alter.

In the dialog shown above we are going to change all the data net labels from D1, D2, etc to Data1, Data2, and so on. As this is a string substitution we do not need to change anything in the Current Attributes.

Attributes to Match By

After changing the attributes press the Global button. The dialog will expand to look like the one shown in the picture above. In the center of the dialog there be a column titled Attributes To Match By. In the Attributes To Match By column you define *how to identify the other objects that this change is to apply to*.

The Attributes To Match By column will contain either a choice field for each attribute or a text field which you can type in.

The choice field has three options: Same (apply global changes if this object attribute is matched in the target object); Different (apply global changes if this attribute is not a

match in the target object) and Any (the default) which applies the change irrespective of whether the attribute has the same value in both objects.

If the Match By attributes are all set to *any* and the text fields contain the wildcard symbol (*), then the global change will apply to all objects of this type.

Use combinations of Match By attributes to define a particular set of objects to apply the change to. For this example we set the Net Label to D*, to target all net labels starting with the letter “D”.

Copy Attributes

The third column in the dialog is titled Copy Attributes. This column will contain either a check box for each attribute, or a text field which you can type in.

In this column you specify which of the attributes you want to copy the changes to, and if the attribute has a text field what new text value to copy to all matched objects.

◆ To perform a complete string replacement simply remove the braces “{}” and type in the new string.

In the dialog shown we enter the string substitution that we wish to perform {D=Data}. The syntax of String Substitution is described below.

Change Scope

The last parameter to set is the change scope. This will be; the current item, all matching items in this document, or all matching items in the documents that make up the project. Other open documents which are not part of the project will not be affected.

When you press the OK button all Net Labels which start with the letter “D” will have the “D” replaced with the string “Data”, so D1 would become Data1, D2 becomes Data2, and so on. Note that if the design included a net label with a value of “DR” it would become DataR.

◆ If you are unsure if your global edit will target the right objects, only enable the Selection option in the Copy Attributes column. When you click OK to perform the change the set of target objects will be selected, but not changed. You can then do the global change again, this time using selection as the Attribute to Match By.

Using Wildcards when Globally Editing Text

Many objects include text fields. These text fields allow you to use wildcards to define changes. This applies to parts, net labels, annotations (single line text), sheet symbols, sheet entries, ports and power ports.

Both the “?” and “*” wildcard characters can be used to extend the definition of target strings. For example, S* will limit the fields to strings beginning with S, etc. Wildcards are case in-sensitive.

◆ Separate **Edit » Find Text** and **Edit » Replace Text** features allow text replacement across different object types. They also support the wildcard search and string substitution syntax.

Syntax for String Substitutions

The Copy field for text strings is used to define the changes to be made to the string. This field can be used in two ways.

If you wish to replace the entire contents of this field with a new value, remove the braces, “{” and “}” and enter the new value.

The braces are used when you wish to perform selective string substitutions, using the syntax {oldtext=newtext}. This means you can change a portion of the string "oldtext" to "newtext". In the example shown earlier in this section we changed the letter “D” for the string “Data”.

You can use multiple sets of brackets to define complex replacements. In this case the leftmost replacement is made, then the next on, etc. This is very powerful, you must take care because the first change can effect subsequent replacements, possibly generating an unexpected result.

You can further limit the replacement by typing {!Text=text} to make the changes case sensitive. In this case, "Text" becomes "text". Otherwise replacement is case insensitive by default. Use Undo to recover from any mistakes.

Summary

With care and planning you can experience significant productivity benefits from this powerful feature. However, the very power of these options can contribute to some unanticipated results – particularly when complex selections are globally edited. When in doubt, it’s always safest to **Edit » DeSelect All** (shortcut: X, A), then create a fresh selection. Remember, the Undo/Redo features allow you to recover several operations, if required.

Quick-Copying an Object

The Schematic Editor has a powerful feature for copying the attributes of one object into a second object *of the same type*. With this feature you can “morph” the object currently floating on the cursor into an object already placed on the schematic.

For example, lets say you have a capacitor floating on the cursor, when you would really like to place a resistor. Rather than pressing ESC to get rid of the capacitor, then browsing through the library to find a resistor, you can use the quick-copy feature to turn the capacitor into a resistor.

Position the cursor (with the floating capacitor) over the “source” object (an existing resistor on the schematic), and press the INSERT key. There will now be a resistor floating on the cursor. You can also edit the attributes of the object before you place it by pressing the TAB key.

All attributes of the placed object will be copied to the floating object. If the floating object did not “inherit” the attributes of the placed object, the cursor cross-hair may not have been inside the body of the placed object when pressing INSERT. Place the object in the desired location.

The quick attribute copy feature (also known as “morphing”) can be used to “clone” the attributes of all schematic object types, except sheet symbols. This feature works whether placing a new object, or moving an already-placed item. It does not work when *dragging* connected electrical objects.

Activity - Finding and Replacing Text

The Schematic Editor allows you to find and replace text anywhere on a sheet, or even across a multi-sheet project.

For example, you may wish to rename all the data bus nets from D1, D2, D3, etc, to Data1, Data2, Data3, across the entire multi-sheet project. This could be done using global editing, but it would have to be done separately for net labels, ports and sheet entries.

To search and replace the text:

1. Select **Edit » Replace Text** from the menus.

The Text Find And Replace Text dialog will pop up.

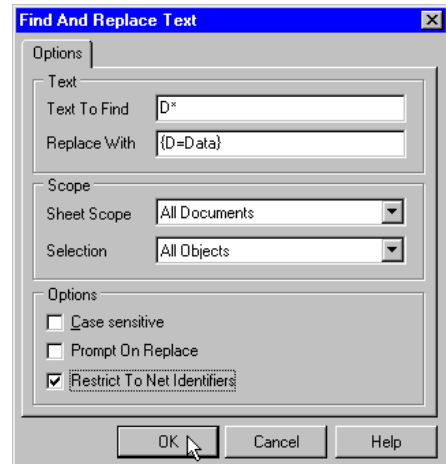
2. Type the Text To Find string, for this example D*.

The Find Text and Replace Text process launchers support the use of both the “?” (any single character) and the “*” (any characters) wildcards.

3. In the Replace With field type the replacement text.

In this example we want to replace the “D” from each string, without effecting the numeric part of the net label. If we just entered the string “Data” then every net identifier would be changed to “Data”. By using the string substitution feature `{oldstring=newstring}`, you can specify a partial string replacement (see below).

4. Set the Scope and the Options. These are described below.



5. Click OK to perform the text replacement.

Scope

Changes can be applied to the Current Document Only or to All Open Documents. Objects with text to be changed can be restricted to selected or un-selected items.

Case Sensitive

Changes can be made on a Case Sensitive basis (upper and lower case must match exactly when searching). Replacement text always matches the case used when typing text into the New Text field.

Prompt on Replace

Always prompt before replacing the found text.

Restricting changes to net identifiers

Restrict find-and-replace text changes to net identifier objects, including; net labels, power ports, ports and sheet entries.

Conditional String Substitutions

Both the “?” and “*” wildcard characters can be used to extend the definition of target strings. For example, S* will limit the fields to strings beginning with S, etc. Wildcards are case in-sensitive.

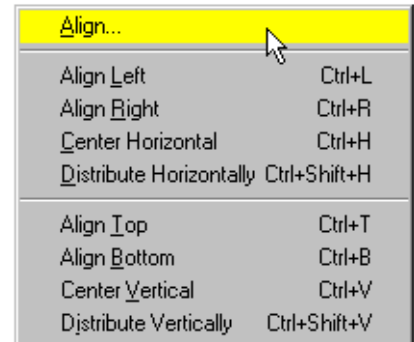
Selective string substitutions can be performed, using the {oldtext=newtext} syntax. This means you can change a portion of the string "oldtext" to "newtext". In our example we changed the letter “D” for the string “Data”.

You can use multiple sets of brackets to define complex replacements. In this case the leftmost replacement is made, then the next on, etc. This is very powerful, you must take care because the first change can effect subsequent replacements, possibly generating an unexpected result.

You can further limit the replacement by typing {!Text=text} to make the changes case sensitive. In this case, "Text" becomes "text". Otherwise replacement is case in-sensitive by default. Use Undo to recover from any mistakes.

Aligning Objects

Two methods of alignment are provided. You can align a group of selected objects on both axes, by choosing **Edit » Align » Align**. Alternatively you can align objects on one axis, by choosing the other Align sub-menu items.



*Align objects by selecting them and then choosing **Edit » Align » Align** to pop up the dialog, or by choosing **Edit » Align » Align Left**, **Align Right**, etc.*

Notes on Aligning Objects

- Ensure that only the required objects are selected – first clear the current selection (shortcut: x, a).
- Select all the items to be aligned.
- Use the **Edit » Align » Align** menu item to pop the dialog if you want to align both horizontally and vertically, for one direction you can just pick the required option.
- Use the Move primitives to Grid option to constrain alignment to the nearest grid point.
- If you are not happy with the alignment use **Edit » Undo** to remove the change.

Moving and Dragging

In the Schematic Editor moving and dragging are different operations; to *Move* an object is to change its position on the sheet and not maintain connectivity, to *Drag* an object is to change its position on the sheet whilst maintaining connectivity.

Moving and dragging can be performed on single objects, or on selected groups of objects.

The Move Object Shortcut

To move a single object, position the cursor over the object, click and hold on the object, then move it to the desired location. To move a group of objects they must be selected first. Once they are selected, click and hold on any of the selected objects, then move the selection to the desired location. Both of these operations can also be initiated via the **Edit » Move** sub-menu.

Moving Objects in a Stack of Objects

Graphical objects can be placed in a sheet so that they overlap. When you place a new item, it is always placed at the “front” of other items. When you move items, they retain their position in the display, relative to other overlapping items. The Schematic Editor includes special move features for changing the “stacking” order of items in the sheet. Press the M shortcut key to pop up the **Move** sub-menu and select one of the following:

Move To Front

Moves an object to the front of other items in the stack and allows you to re-position it. When you select this menu item you are prompted to choose the item to be moved. When you click on the item, it floats on the cursor. Click a second time to place the object once it has been re-positioned.

Bring To Front

Moves an object to the front of other items in the stack. When you select this menu item you are prompted to choose the item to be moved. When you click on the item it moves to the front of the stack without changing its x or y coordinates.

Send To Back

Sends an object to the back of other items in the stack. When you select this menu item you are prompted to choose the item to be moved. When you click on the item it moves to the back of the stack without changing its x or y coordinates.

Bring To Front Of

Moves an object to the front of a second item. When you select this menu item you are prompted to choose the item to be moved. When you click on the item, you are then

prompted to choose the “target” item. The item to be moved will be re-located in front of the “target” without changing its x or y coordinates.

Send To Back Of

Moves an object behind a second item. When you select this menu item you are prompted to choose the item to be moved. When you click on the item, you are then prompted to choose the “target” item. The item to be moved will be re-located behind the “target” without changing its x or y coordinates.

Dragging Objects

Often you will need to rearrange the objects on the schematic sheet, perhaps as you add to the circuit, but you want to maintain the connectivity. To do this you need to *Drag* the object(s).

Dragging a Single Object

To drag a single object, position the cursor over the object, hold the CTRL key down and click and hold on the object. Release the CTRL key, then drag the object to the desired location.

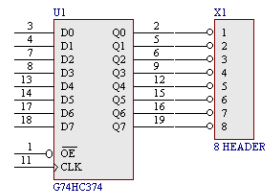
Dragging a Group of Objects

To drag a group of objects they must be selected first. Once they are selected, select the **Edit » Move » Drag Selection** menu item, click to choose the reference location, then drag the selection to the desired location.

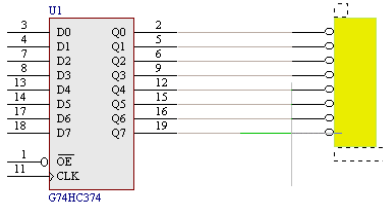
Tips on Dragging

- If the wiring is dense you can find that wires land on top of other wires as you drag a selection, which are then connected by the auto-junction feature. Select **Tools » Preferences** to disable Auto-Junction.
- Another technique that helps when dragging in a dense design is to disable the Drag Orthogonal feature (select **Tools » Preferences**). This allows wires to move at any angle, and will avoid wires landing on top of other wires as you drag. You can then re-shape each wire as required; click to focus, then click and move the vertices as required. New vertices can be added and existing vertices removed, refer to the Focus topic earlier in this chapter for more information.
- If two component pins are touching a wire will automatically be added when you drag them apart. This also works for an orthogonal wire across the end of component pins.
- Press CTRL + SPACEBAR to rotate the objects being dragged.
- Press the SPACEBAR to toggle the orthogonal wire mode while dragging.

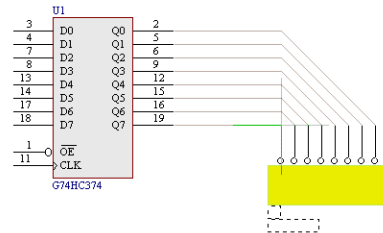
Dragging Objects - a Graphical Example



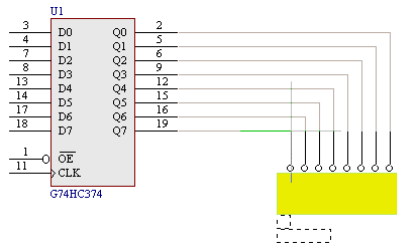
original component placement, note that the component pins are touching



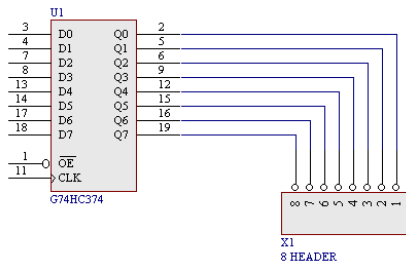
dragging X1 (CTRL+Click and hold, release the CTRL then drag X1), note how the wires are added automatically



press SHIFT+SPACEBAR to rotate X1



press the SPACEBAR to change the wiring mode



click to place X1

Using the Schematic Editors interactive editing features to change the layout of you schematic

Schematic Editing Shortcuts

Re-entrant Editing

The Schematic Editor includes a powerful feature which allows you to perform a second operation, without having to quit from the operation you are currently carrying out. This facility is known as re-entrant editing.

Re-entrant editing allows you to work more flexibly and intuitively. For example, you start placing a wire then remember that it needs to be connected to a port. There is no need to drop out of Place Wire mode, simply press the Place Port shortcut keys (P, R), place the port, press ESC to drop out of the Place Port process, and then connect the wire to the port.

A large number of processes can be completed within another process. The number of times another process can be launched before the current process is complete depends on the demands each of these incomplete processes is placing on the software.

♦ If you are already executing a command you can only execute a second command by using the shortcut keys.

Canceling a Screen Redraw

Whenever you change the size and/or position of your view of the screen, the contents of the workspace will be redrawn to reflect the change. You can terminate the redrawing process at anytime by pressing the SPACEBAR. This saves time whenever you wish to immediately scroll or zoom again, without waiting for the redraw to complete.

Mouse Shortcuts

As you read through this guide, you will notice several mouse and keyboard shortcuts that are used to speed-up or simplify frequently performed operations. For example, pressing P, N allows you to place a net label without having to go to the Place menu and choose the Net Label menu item. Using the left mouse button for ENTER and the right mouse button for ESC will allow you to perform many operations without using the keyboard. The opposite can also be done, press ENTER or ESC on the keyboard rather than clicking OK or CANCEL in a dialog box.

Sometimes keyboard actions provide the only practical way of performing an operation when you do not wish to move the mouse in the workspace, such as toggling the Snap Grid off when you are placing a text string (V, G), or changing the zoom level while moving a selection.

Standard Windows shortcuts, such as pressing ALT+F4 to close the application window or CTRL+TAB to toggle through document windows are supported. Other shortcut keys are specific to the Schematic Editor. You can also create your own custom shortcut keys. Refer to the *Design Explorer* section for clues on creating your own shortcut keys.

Windows allows you to assign operations to specific key combinations by using the Recorder feature. See your *Microsoft Windows Users Guide* for details.

Keyboard Shortcuts

There are two ways of creating shortcuts invoked through the keyboard. The first is through the Keyboard Shortcut Editor (**Client menu » Customize**). These are known as Keyboard Shortcuts – these shortcuts launch a process directly. For example, pressing CTRL+T will align the selected objects along their top edge.

Processes can also be launched through the keyboard via the menu keyboard shortcuts. Underlined menu items which lead to a sub-menu will pop up that sub-menu, underlined menu items which do not pop up a sub-menu will launch the process tied to that menu item. For example, press P to pop up the Place menu, then press N to present the current Net Label on the cursor, ready for placing. Press T to pop up the Tools menu, followed by N to pop up the Netlist Creation dialog.

If the same key has been assigned to a keyboard shortcut and a menu shortcut, then the keyboard shortcut will take precedence.

Menu Shortcuts include:

A	Edit » Align sub-menu
B	View » Toolbars sub-menu
E	Edit menu
F	File menu
H	Help menu
J	Edit » Jump sub-menu
L	Edit » Set Location Marks sub-menu
M	Edit » Move sub-menu
D	Design menu
P	Place menu
R	Reports menu
S	Edit » Select sub-menu
T	Tools menu
V	View menu
W	Window menu
X	Edit » DeSelect menu
Z	Zoom pop-up menu

Keyboard Shortcuts include:

CTRL+BACKSPACE	Redo
ALT+BACKSPACE	Undo
PGUP	Zoom In
CTRL+PGDN	Zoom All
PGDN	Zoom Out
END	Redraw
CTRL+HOME	Jump to Origin
HOME	Pan
SHIFT+LEFT ARROW	Cursor Left (10x snap grid)
LEFT ARROW	Cursor Left (on snap grid)
SHIFT+UP ARROW	Cursor Up (10x snap grid)
UP ARROW	Cursor Up (on snap grid)
SHIFT+RIGHT ARROW	Cursor Right (10x snap grid)
RIGHT ARROW	Cursor Right (on snap grid)
SHIFT+DOWN ARROW	Cursor Down (10x snap grid)
DOWN ARROW	Cursor Down (on snap grid)
SHIFT+INSERT	Paste
CTRL+INSERT	Copy
SHIFT+DELETE	Cut
CTRL+DELETE	Clear
DELETE	Delete focused object
CTRL	Temporarily disable the snap grid
CTRL+1	Zoom 100
CTRL+2	Zoom 200
CTRL+4	Zoom 400
CTRL+5	Zoom 050
CTRL+F	Find Text
CTRL+G	Find And Replace Text
CTRL+V	Align objects horizontally, through their centers
CTRL+B	Align objects horizontally, along their bottom edge
CTRL+T	Align objects horizontally, along their top edge
CTRL+SHIFT+H	Space objects equally horizontally
CTRL+H	Align objects vertically, through their centers
CTRL+L	Align objects vertically, along their left edge
CTRL+R	Align objects vertically, along their right edge
CTRL+SHIFT+V	Space objects equally vertically
F1	Run Online Help
F3	Find next text
SHIFT+F4	Tile all open documents
SHIFT+F5	Cascade all open documents
SHIFT+CTRL+Left-Click	Move single object
SHIFT+Left-Click	Toggle single object selection
CTRL+Left-Click	Drag single object

Left-Click	Focus object
Left-Double-Click	Change object
CTRL+Left-Hold-Down	Drag single object
ALT	Constrain object movement to Y direction
ALT+SHIFT	Constrain object movement to X direction
Left-Hold-Down	Move object / move selection

Frequently Used Shortcut Keys

- SPACEBAR to abort screen re-draws
- X, A to de-select all
- V, D to zoom to fit the sheet
- V, F to zoom to fit all placed objects
- PAGE UP to zoom in (zooms in around the cursor, so position the cursor first)
- PAGE DOWN to zoom out
- HOME to re-center the screen at the current cursor position
- END to refresh the screen
- TAB while an object is floating on the cursor to edit its attributes prior to placement.
- SPACEBAR whilst placing an object to rotate it by 90 degrees
- X whilst placing an object to flip it along the X axis
- Y whilst placing an object to flip it along the Y axis
- BACKSPACE whilst laying a wire/bus/line/polygon to delete the last vertex
- SPACEBAR while laying a wire/bus/line to step through the placement modes
- ESC to escape from what you are doing when you change your mind
- CTRL to temporarily disable the snap grid while moving/placing an object
- CTRL+TAB to switch between open documents in the Design Explorer
- ALT+TAB to switch between open applications in Windows
- F1 when you have a cross-hair cursor to pop up a list of shortcut keys

Undo and Redo

The Schematic Editor includes a full multi-level Undo and Redo facility. Each procedure is stored in a stack-like arrangement. When Undo is chosen, the last operation is undone. Choosing Undo again will undo the next-to-last operation, and so on.

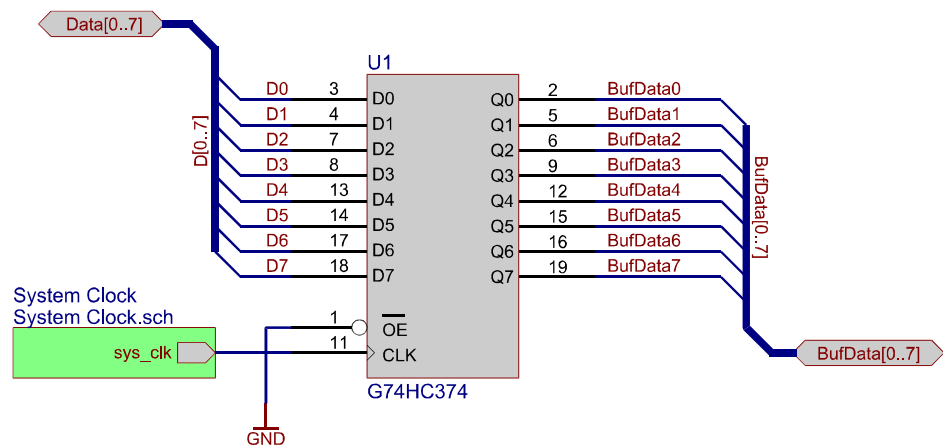
The Schematic Editor has been optimized to ensure that Undo and Redo operations are not memory intensive, however you can clear the Undo stack by temporarily setting the Stack Size to zero in the Preferences dialog.

Schematic Design Objects

The schematic environment, whether creating and editing circuit diagrams, or working in the library editor, consists of two basic features: *objects* that are placed in the workspace to build up the design; and *processes*, which are used by the system or the user to create, modify, save and report on the objects.

There are two types of objects in the Schematic Editor, *Primitive Objects*, and *Group Objects*. Primitive objects are the most basic schematic elements and include the electrical objects; such as wires, junctions, power ports, net labels, and sheet entries – as well as the drawing objects; such as text and the various drawing primitives. Components are a group object, being constructed from electrical and drawing primitives.

Each object has its own set of attributes, which can be defined for each instance of the object. The next section of this chapter gives a brief description of each of the primitives.



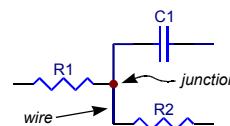
Your design is built up by placing the various design objects to create the circuit.

Electrical Schematic Primitives



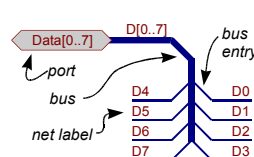
Wire

Wires are straight line segments which are placed on the sheet to create the electrical connections.



Junction

Junctions are small circular objects used to logically join intersecting wires on the schematic.



Bus Entry

A Bus Entry is a special wire at an angle of 45 degrees, that is used to connect a wire to a bus line. A Bus Entry allows you to connect two different nets to the same point on a Bus. If this was done using wires the two nets would short.



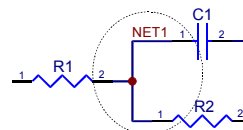
Bus

Buses are special graphical objects that represent a common pathway for multiple signals on the sheet. Buses have no electrical properties, they must be correctly identified by Net Labels and Ports, as shown in the figure above. The Bus net label can be either ascending (eg D[0..7]), or descending (eg D[7..0]).



Net Label

When you wire from one component pin, to a second pin, then to a third pin, you are creating a *Net*. When you create a netlist each net is given a unique identifier.



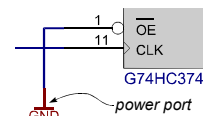
Rather than letting the Schematic Editor assign a system generated net identifier, you can identify any net with a *Net Label*. Net Labels can be placed horizontally or vertically. The bottom left corner of the Net Label must be adjacent to the wire or bus for them to be associated. NET1 in the figure above has three pins (or nodes); R1-2, R2-1 and C1-1.



Power Port

Power Ports are special symbols that represent a power supply net. Power Ports allow you to conveniently indicate a power net at any location in the project, which can then be connected to pins or wires.

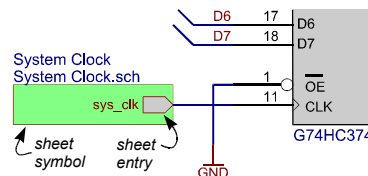
A Power Port is always identified by its Net name, not its graphic – ensure that the graphic is appropriate for the net name. A net name can be pre-assigned to the 3 styles of power ports that do not display their name, this is done in the Preferences dialog.





Port

A Port is used to connect a net on one sheet, to Ports with the same name on other sheets. Ports can also connect from a child sheet, to Sheet Entries in the appropriate sheet symbol on the parent sheet. Press the SPACEBAR to rotate the port during placement.



Sheet Symbol

Sheet Symbols represent another schematic sheet (often referred to as a child sheet). The link between the Sheet Symbol and the other schematic sheet is the File Name attribute, which must be the same as the name of the child sheet.

◆ Refer to the chapter *Multi-Sheet Design and Project Management* for more information on how to use Ports, Sheet Symbols and Sheet Entries.



Sheet Entry

A Sheet Entry creates a connection between the net touching it on the parent sheet – to a Port with the same name on the child sheet. Drag the sheet entry to position it in the sheet symbol.



Probe Directive

A Probe is a special marker which is placed on the worksheet to identify nodes for digital simulation.

Test Vector Directive

Test Vectors are special symbols used to identify a node with a simulation test vector. The test vectors are referred to by a column number, which indicates the column of the test vector file to use when the simulation is run.



Stimulus Directive

A Stimulus is a special symbol which is used to identify a node or net to be stimulated when the digital simulation is run.



PCB Layout Directive

PCB Layouts are special symbols that allow you to attach board layout information to a specific net.



No ERC Directive

The No ERC directive is a special symbol that identifies a pin as one that you want the Electrical Rules Checker to ignore.



Pin

Pins are special objects that have electrical characteristics and are used to direct signals into and out of Parts. Refer to the *Library Editor* chapter for more information on pins.

Non-Electrical (Drawing) Primitives

The non-electrical drawing tools are used to add reference information to a sheet. Use them to build graphical symbols, create custom sheet borders or title blocks or adding notes and instructions.



Line

Lines are graphical objects with any number of joined segments.



Polygon

Polygons are multi-sided graphical objects. It takes a minimum of three clicks to define a polygon. Each vertex of the polygon can be moved independently.



Elliptical Arc

Use the Elliptical Arc tool to create open circular or elliptical curves. There are five clicks to define an Elliptical Arc; the center, the X radius, the Y radius, the first end point, then the other end point.



Bezier Curve

Use the Bezier tool to create curved line shapes (for example a section of a Sin wave, or a pulse). At least four clicks are required to define a Bezier curve. If you continue clicking after the fourth click you then commence a new Bezier curve, attached to the previous curve.



Text Annotation

An annotation is a single line of text used to place notes or comments on the sheet. Uses might include section headings, revision history, timing information or some other descriptive or instructive text.



Text Frame

Text frames hold multiple lines of free text. Text frames are used to place detailed notes or descriptive text on the worksheet.

Rectangle



Rectangles are filled or unfilled graphic elements. It takes two clicks to define a rectangle; the top left corner, and the bottom right corner.

Rounded Rectangle



Rounded Rectangles filled or unfilled graphic elements with rounded corners. It takes two clicks to define a rectangle; the top left corner, and the bottom right corner.

Ellipse



Ellipses are filled or unfilled graphic elements. It takes three clicks to define an Ellipse; the center, the X radius, and the Y radius.

Pie



Pies are filled or unfilled graphic elements. It takes four clicks to define a Pie; the center, the radius, the start point, and the end point.

Graphic Image



Graphic images can be included on your schematic. The following file formats are supported:

BMP, TIF, JPG, WMF

Note that a copy of the image is not stored inside the sheet, only a link to it. If the location of the image changes you will need to update the link (double-click on the image to do this).

Refer to the *Placing Schematic Objects* topic in the *Working in the Schematic Editor* chapter for examples of how to place schematic design objects.

Process Container

The Schematic Editor includes a method of storing a process identifier with a particular configuration of the its process parameters. This is done in a *Process Container*.

Netlist:RunElectricalRulesCheck ECL ERC setup
C:\Client 98\SCH4\EXAMPLES\Demo1.ERC

You can use a process container to hold the setup for a particular process with the design. As an example, your design may have a special ERC setup that you need to store with the design for future reference.

Configuring a Process Container

After placing an empty process container double-click on it to configure it.

Process Name

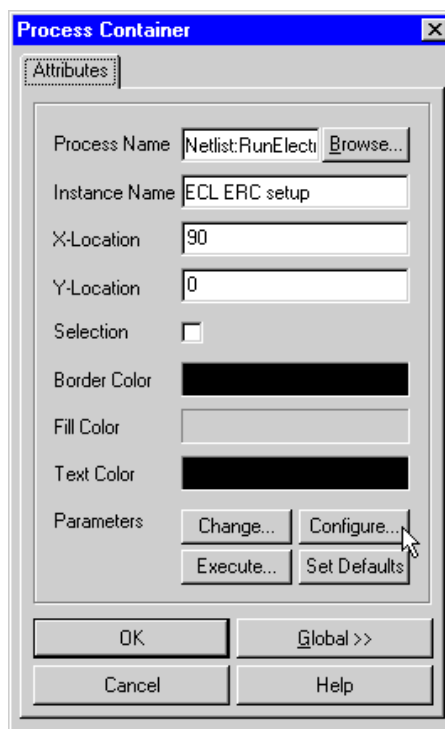
Use the Browse button to locate the process that you would like to run from the container.

The Configure Button

If the container is running one of the Netlist server processes, they can be configured by pressing the Configure button. This will pop up the appropriate dialog – set it up as required. When you click OK the process will not run, you will simply return to the Process Container dialog (the *Sch:Annotate* process can also be configured in this way).

The Change Button

Once you have run the container once, you can press the Change button to examine the process parameters. You can change the parameters here, or use the Configure button again.



Not all processes can be configured from the process container (press the Configure button, if the process runs immediately then it can not). To set up the parameters for these processes press the Change button and enter the process parameters, one per line.

Executing

To execute a process container, select **Tools » Process Containers » Run**. Alternatively, double-click on the process container and press the Execute button.

Files Produced by Process Containers

If the process specified in the process container generates a file, such as *Netlist>CreateNetlist*, the name of the netlist file can be specified in the Change Parameters dialog. When the container is run the file will be opened. It will also be linked as a child sheet of the schematic in the Project Manager.

Fonts

The Schematic Editor supports the TrueType fonts that are delivered with Windows, including bold and italic formats and display font scaling. PostScript scaleable fonts, and Windows non-scaleable raster fonts can be used when they are part of vector image files imported into schematic sheets.

Organization of Fonts

You can think of the text fonts as belonging in one of two groups; those that are individually editable, and those that use the *System Font*.

- Individually editable fonts include; component designator, part type, net label, text annotation, text frame, sheet symbol name and sheet symbol file name.
- Text that uses the system font include; border text, system title blocks, pin names, pin numbers, ports, power ports and sheet entries.

Changing Fonts

The easiest way to tell if a font is an individually editable font is to double-click on the text. The dialog for that object will pop up. If it has a Font Change button, then this is an individually editable font. Pressing this will pop up the Font dialog. If double-clicking produces no response, or if there is no Font Change button, this text uses the system font.

You can also specify default fonts for all objects that have an editable font. Select **Tools » Preferences** to display the Preferences dialog, then click on the Defaults Tab. Select the object that you wish to set the font for (for example Net Label) and press the Edit Values button. Select the required font in the normal way. From now on all new net labels will use this font. Note that it will not change the font for net labels that are already on the sheet, these can be changed using global editing.

Setting the System Font

To change the system font, select the **Design » Options** menu item. This will pop up the Document Options dialog. On the Sheet Options Tab there is a Change System Font button. Pressing this will pop up the Font dialog. Select the required font in the normal way. This will change all text that uses the system font.

Schematic Components and Libraries

Efficient use and management of schematic libraries requires an understanding of the relationship between *libraries*, *components*, and *parts*. Libraries are databases which store component descriptions. Each component then consists of one or more parts, such as a 74LS00 which contains four parts, a capacitor which has one part, or a relay which may be made up of a coil part and a contact part.

Components are created and modified in the Schematic Library Editor, an independent document editor that includes tools for managing and editing libraries. The Schematic Sheet Editor and the Schematic Library Editor can have documents open simultaneously within the Design Explorer environment, with special features to link between sheet and library operations. For example, you can move directly from a part symbol on the sheet, to editing its component information inside the source library.

What is a Schematic Library?

Protel 99 SE is delivered with extensive component libraries that include the industry standard symbolic representation, as well as ANSI-IEEE and DeMorgan equivalents.

The schematic libraries consist of component descriptions, made up of the individual parts that are placed in the schematic sheet. Components can have one or many parts (e.g., the gates that comprise a multi-part component like a 74LS00).

The Schematic Library Editor allows you to manage library data, create new components, and move and copy components between libraries. Refer to the *Schematic Library Editor* chapter for information on the general features and use of this editor.

♦ The Altium Library Development Center is constantly developing new libraries – check www.protel.com to download the latest libraries for all your Protel tools.

What is a Component and what is a Part?

The best way to think of a *component* is to equate it to the physical device that is placed on the actual printed circuit board; the resistor, the integrated circuit, or the connector for example. Many electronic components also comprise of separate *parts* within the one component. Examples of these would be; a 74LS00 which contains four parts, or a dual op amp which contains 2 op amps.

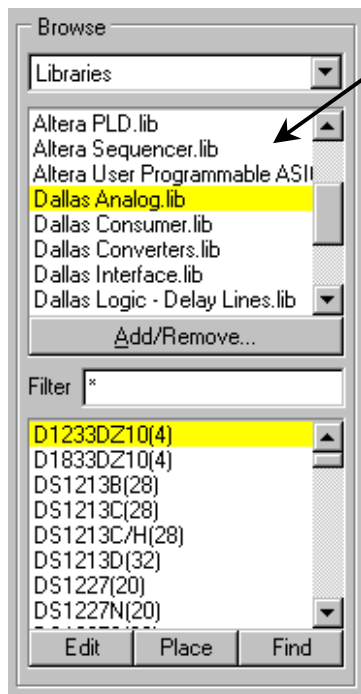
Where are the Schematic Component Libraries?

In Protel 99 SE, the schematic libraries are stored within a set of Library Design Databases. Most of these Schematic Library Databases are manufacturer specific – each database includes all the libraries that have been produced for that manufacturer. There are also a number of other special-purpose databases, for example, one for schematic-base PLD design, and another which includes all the simulation-ready symbols.

The Protel 99 SE Library Databases are in the \Program Files\Design Explorer 99 SE\Library\Sch folder.

- ◆ In Protel 99 SE you can still access the components in the older .LIB format libraries. These libraries can be added to the list of available libraries in the normal way.

Accessing the Components You Need for Your Design



- ◆ The list of libraries whose components are currently available in the Schematic Editor. Press the Add/Remove button to add and remove libraries from the list.

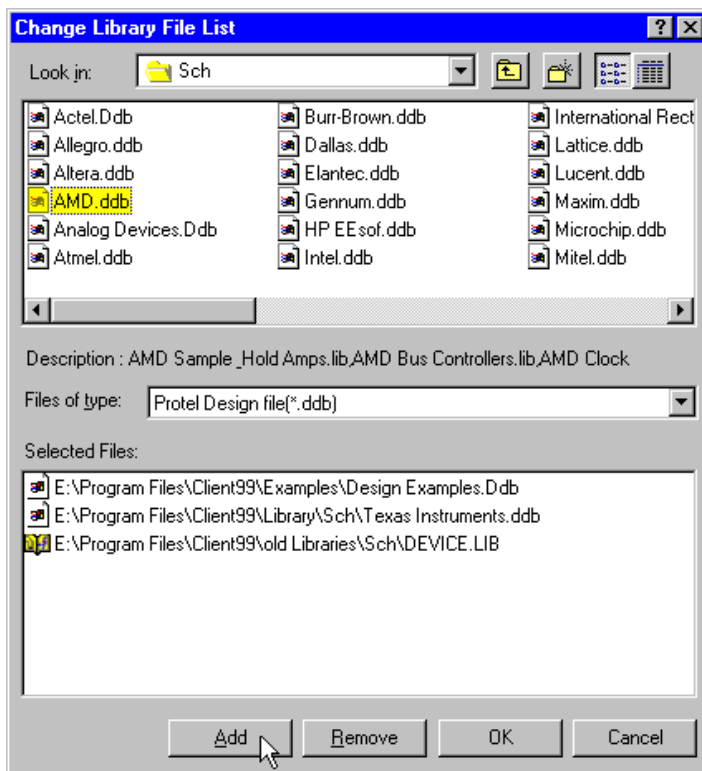
To access the components in the schematic libraries, the libraries must first be *added* to the list of available libraries in the Schematic Sheet Editor. The top of the Schematic Editor Panel includes a list of all the currently added libraries.

There are two ways to add or remove a library; either by pressing the Add/Remove button on the panel, or by selecting **Design » Add/Remove Library**. This pops up the Change Library List dialog where new libraries can be added, and added libraries removed from the Current File List. The only limit on the number of libraries that can be added is the memory available in your computer.

Once libraries have been added, parts from those libraries can be placed on the sheet.

Adding and Removing Libraries

Use the Change Library File List dialog to access components in the libraries. The Selected Files window at the bottom of the dialog lists all the currently added libraries.



After locating the required Library Database, double-click on it to add it to the list. Double-click on a Library Database in the Selected Files list to remove it.

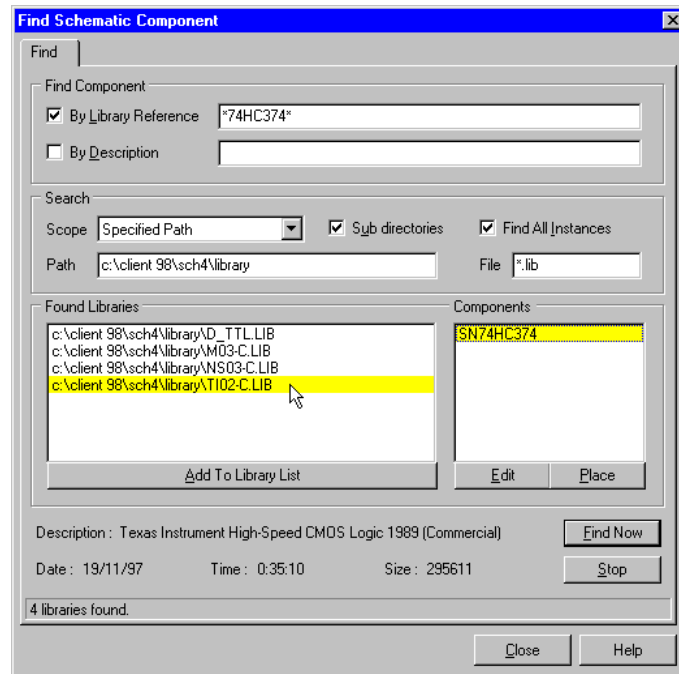
Use the Look in field at the top of the dialog to browse to the folder where the Library Databases are located. The Protel 99 SE Library Databases are stored in the \Program Files\Design Explorer 99 SE\Library\Sch folder.

- ◆ Libraries are stored in standard Protel 99 SE Design Databases, which means you can easily create your own Library Databases. These databases can even be a mix of schematic and PCB libraries.
- ◆ To access components in libraries that are stored inside a Project Design Database, simply add the Project Design Database to the Selected Files list in the Change Libraries File List dialog.

Finding a Component in the Libraries

Adding a library is easy if you know which library you want, and what Library Database it is in – but what do you do when you do not know which library the component is in? The Schematic Editor includes a powerful Find component feature.

To find a component press the Find button on the panel, or select **Tools » Find Component** to pop up the Find Schematic Component dialog.

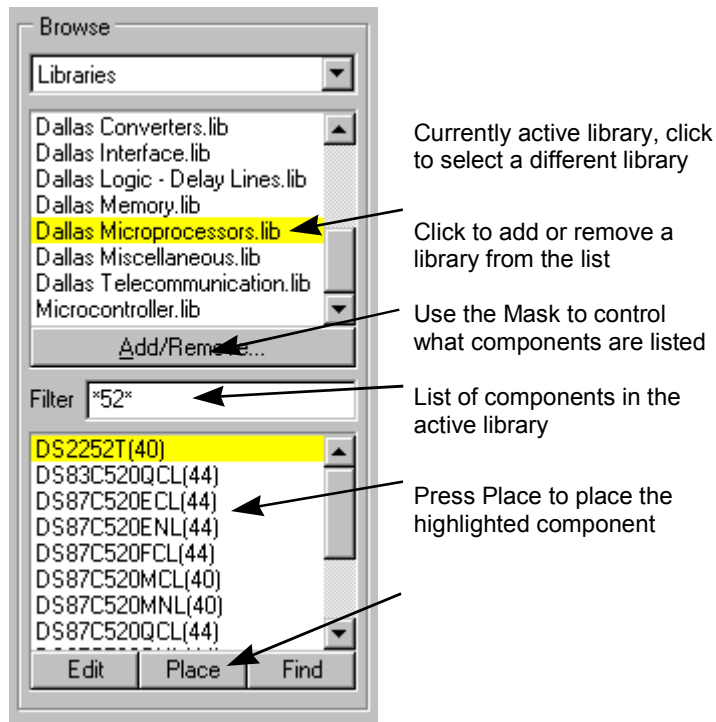


Tips on Finding Components

- Searching by Library Reference is generally faster.
- Include the * wildcard before and after the search string (as shown in the image), as different manufacturers use different prefix and suffixes.
- Use multiple search strings in the description field to improve the chance of a match when searching by Description for more than one word. Enclose each word with the wildcard character and separate with a space (eg – *barrel* *shifter*).
- If your search produces no results check that the Path is correctly specified. Also, try searching for a component that you know is in a library to check that everything is set correctly.
- Use the Add to Library List button once you locate the correct component. When you press this button the selected library is added to the list of available libraries in the Schematic Sheet Editor.

Placing Parts on the Schematic Sheet

Once you have located the components you require and you have added the libraries to the library list, you are ready to start placing the parts. To place a part select it in the Schematic Sheet Editor Panel, and press the Place button.



You can also place parts by selecting **Place » Part** in the menus, or by pressing the Place Part button on the Wiring Toolbar. This will pop up the Component Library Reference dialog. When you enter the exact Library Reference all the Libraries in the list will be searched for this part.

- ◆ You can edit the attributes of the component floating on the cursor before actually placing it on the sheet. To do this, press the TAB key while the component is “floating”, and you will be presented with the Part dialog. If you set the designator now and continue to place instances of the same part, the designator will auto-increment.
- ◆ While the component is floating on the cursor it can be rotated by pressing the SPACEBAR, and flipped along the X or Y axis by pressing the X or Y key.
- ◆ When placing parts from a multi-part component there is no need to specify which part it is (eg: U1:A), the alpha suffix is added automatically.

The Attributes of a Schematic Part

Like all objects in the Schematic Editor, parts have a set of attributes associated with them. Some of these attributes can only be defined in the Library Editor, and therefore apply to all instances of that part, some can only be defined in the Schematic Sheet Editor and are associated with only that instance of the part, and some can be defined in either.

To edit the attributes of a part double-click on the placed part, or select **Edit » Change** and click on the part. Each editable attribute can be globally edited, and the changes apply to some, or all of the parts on the sheet. Refer to the *Global Editing* topic in the *Working in the Schematic Editor* chapter for tips on global editing.

About Component Text

Parts have two sets of special text fields, eight *read only library Text Fields* whose values are entered in the library editor, and sixteen *Part Fields* whose values are entered at the sheet level once the part is placed.

Library Reference

The Library Reference is the name of the component in the Library. This field can be up to 255 characters long.

Footprint

The Footprint is the physical package that the component will be mapped to when you create a netlist and load the netlist into the PCB Editor. Each part *must* have a Footprint, and this Footprint *must* be available in a PCB Library when you load the netlist into the PCB Editor.

Up to four footprints can be pre-assigned in the Library Editor, or you can enter it once the part has been placed.

Designator

The Designator uniquely identifies each part in your design. If you do not enter a designator before you place a part then its designator will be the pre-assigned default, such as R? or C?.

♦ The default designator prefix for the part designator is assigned in the Library Editor.

To enter the designator before you place the part press the TAB key while the component is floating on the cursor. If you enter the designator while the component is floating on the cursor then the designator will increment automatically (R1, R2 etc) as further parts are placed.

If the component is a multi-part device it will automatically be assigned a part suffix, for example U3A, U3B, and so on. The suffix will automatically increment if the

designator is assigned before you place the part. The suffix is always an alpha character.

Select **Tools » Annotate** to automatically re-assign the designators. Refer to the *Passing Your Design into the PCB Editor* chapter for more information about annotation.

Part Type

The Part Type text field is for the part description, such as the value (220nF) or device type (74HC32). This field can be up to 255 characters long.

Sheet Path

Parts on the schematic sheet can be made to behave as sheet symbols rather than component parts. When they are in this mode, the nets connecting to their pins connect to matching ports on the sheet below. To get a part to behave as a sheet symbol, you specify the sheet that exists below in the Sheet Path field and enable the *descend into sheet parts* option in the Netlist Creation dialog. When a component is configured to behave as a sheet symbol it does not appear in the netlist. Refer to the *Multi-Sheet Design and Project Management* chapter for more information.

Part

This field indicates which part this particular instance is, in the multi-part component. 1 indicates part A, 2 indicates part B, and so on. The part suffix can be either an alpha character, or a number, depending on the setting of the Multi-part Suffix option in the Preferences dialog. Note that this is an environment setting, which applies to all currently open schematic sheets.

Hidden Pins

Component pins can be specified as “Hidden” or “Visible” in the Library Editor. Normally, hidden pins are used for component power pins.

Hidden pins are automatically connected to nets with the same name during netlisting. If hidden pins are displayed they then must be wired manually. If the hidden pins on a multi-part component are being displayed, then the hidden pins on all parts of that component (eg: U2:A, U2:B, U2:C, U2:D) must be displayed.

Hidden Fields

The 16 Part Fields are not displayed on the schematic sheet by default. Use this option to display the values of these fields. Double-click on an individual Part Field to hide it.

Field Names

The names of the 16 Part Fields are not displayed on the schematic sheet by default. Use this option to display the names of these fields. Double-click on an individual Part Field name to hide it. The Part Field names can be edited in the Library Editor.

Read Only Library Fields

Each library component has eight user-definable 255 character text fields. Library text fields cannot be edited in the schematic sheet, but can be viewed and included in the Bill of Materials.

Part Fields

In addition to the eight library component text fields, sixteen additional user-definable fields (255 characters) are available for each part, at the sheet level. These fields can be displayed or hidden (enable the Hidden Fields check box in the Attributes Tab), and are editable on the sheet, with user definable fonts, sizes and colors. These fields can be included in the Bill of Materials.

The names of part text fields can be defined in the Library Editor. Up to 255 characters can be used, but only the end of the character string will be displayed in the Part dialog if the length of the string exceeds the display area. There is room for 14 characters and/or spaces, when 12 point Helvetica is used as the dialog font.

Orientation

Sets the orientation of the part, relative to its original orientation in the library.

Mode

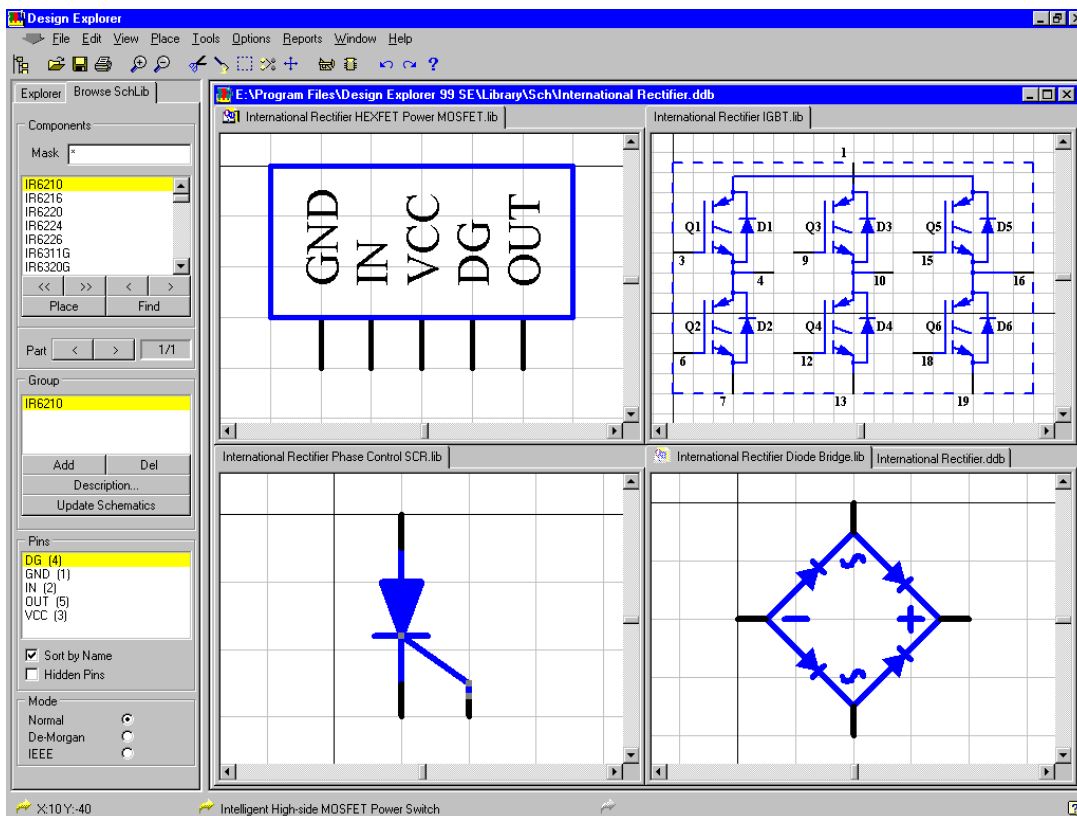
The Schematic Editor supports up to three “Modes”, or drawing styles, for each part – Normal, DeMorgan and IEEE. Use this option to select an alternative mode. Each mode is defined in the Library Editor, however only the Normal mode must be defined. If the drawing style does not change when you change this option it means that only the normal mode has been created.

Colors

Part outline (Line Color), Pin Color and Fill Colors (if part includes closed rectangles, filled arcs or polygon graphics) are pre-defined in the Library Editor. These can be changed locally on the schematic sheet by enabling the Local Colors option, and then setting the colors as required.

If the part graphics are un-filled (you will be able to see the sheet color and grid inside the part), changes made to the Fill Color field will be ignored.

The Schematic Library Editor



The Library Editor is the second document editor included with the Protel 99 SE's schematic server. Where the Schematic Sheet Editor is used for capturing the schematic, the Library Editor is used to create and modify the components used in those schematics.

In Protel 99 SE, the schematic libraries are stored within a set of Library Design Databases. Most of these Library Databases are manufacturer specific – each database includes all the libraries that have been produced for that manufacturer. There are also a number of other special-purpose databases, for example, one for schematic-based PD design, and another which includes all the simulation-ready symbols.

- ◆ The Altium Library Development Center is constantly developing new libraries – check www.protel.com to download the latest schematic libraries.

Working with Libraries

In Protel 99 SE libraries are stored within a Design Database, just like all the other design documents. In this Handbook they are referred to as Library Databases, but these databases are exactly the same as the Design Databases you store your design documents in.

The Protel 99 SE Library Databases are stored in the \Program Files\Design Explorer 99 SE\Library\Sch folder.

Opening an Existing Library

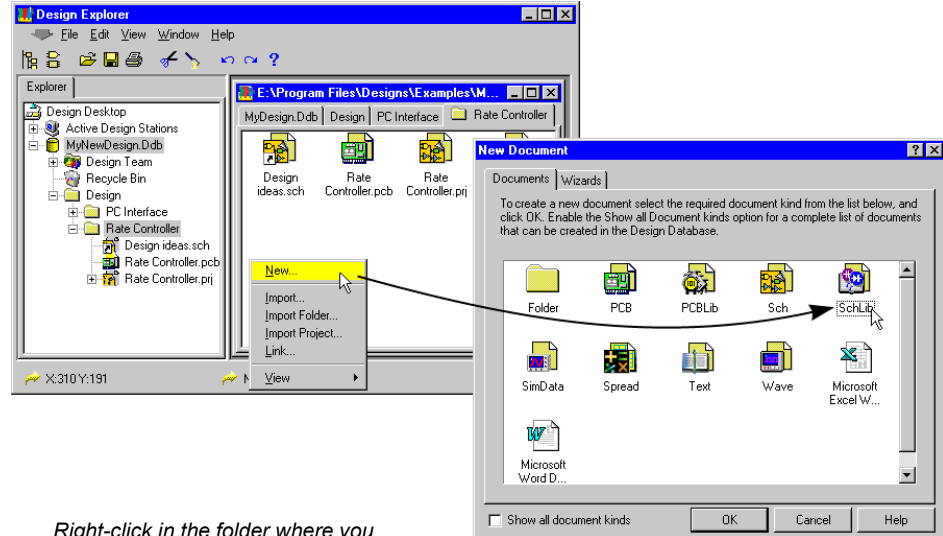
Libraries are opened in the Library Editor in the same way all documents are opened the Design Explorer, by first selecting **File » Open** to open the Library Design Database, then browsing through the database and opening the library for editing.

Each open library will appear on a separate Tab in the integrated Design Window.

Creating a New Library

Before you create a new library you must open the Design Database that you want to store the library in, then browse through the database and open the folder that you want to create the library in.

To create the new library right-click in the folder window, and select **New** from the floating menu that appears. The New Document dialog will pop up, double-click on the SchLib icon to create a new library.



Right-click in the folder where you want to create the new library, select New, then double-click on the SchLib icon.

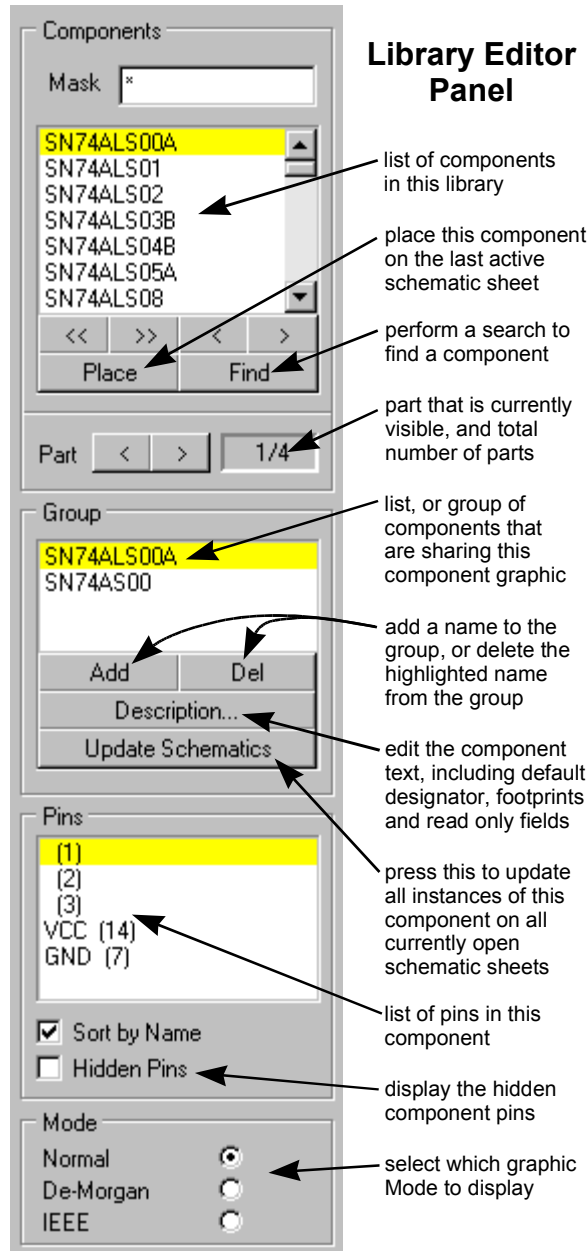
An icon for the new library will appear in the folder, with a name like Schlib1. To rename the library click once to select the icon, then press F2 to highlight the name, ready for typing. Type in the new name and press ENTER on the keyboard. Double-click on the icon to open the new library.

Components and Parts

A library is a set of components. Think of the component as the physical device. Each of these components can then be made up of one or more parts. A resistor component would have one part, whereas a resistor network might have eight parts.

The partitioning of components into parts is entirely up to you as the designer. You may wish to draw the relay coil as one part and the contacts as another, or it might be more appropriate to draw the whole relay as one part. A four pin connector could be drawn as one part, or it could be drawn as four parts. Each part of a component is drawn on a separate sheet. To add a new part to a component, select **Tools » New Part**.

As well being able to have more than one part, each part can have up to three graphical representations, or modes; *Normal*, *De-Morgan* and *IEEE*. Each mode is drawn on a separate sheet. The preferred mode is selected when the part is placed on the schematic sheet, with the default being normal. Only the Normal mode must be created, the other two are optional.



Creating and Editing a Component

All the tools you need to create and edit library components are in the **Tools** menu. To create a new library component:

1. Select **Tools » New Component**.
2. You will be presented with an empty component sheet, titled Component_1, with a cross-hair through the center (origin) of the sheet.
3. You are currently at full zoom, so press **PageUp** until the grid becomes visible. If necessary re-locate the origin of the sheet by selecting **Edit » Jump » Origin**.
4. The first step is to define the component body. If the Component has a rectangular shape select the Rectangle tool from the toolbar, or in the **Place** menu. Alternatively, use the line tool to draw the required shape.
5. Click at the sheet origin to define the top left of the component body. Use the Status line to confirm that you have the cursor at the origin.
6. Move the mouse down and to the right to size the rectangle. The size will depend on the number of pins the component will have. Use the grid to guide you as you size it, allowing one grid line per pin.



◆ If you accidentally moved the cursor outside the window and the view suddenly pans, use the **J, O** shortcut keys to jump back to the origin.

7. Click a second time to define the bottom right corner of the component body.

◆ You can re-size the component body at any time – simply click once inside the rectangle to *focus* it, then click and drag on the small handles that appear. Click outside to rectangle to release the focus when you are finished.

8. After defining the component body you can place the component pins. When you click on the Place Pin tool on the toolbar a pin will appear floating on the cursor. You are holding the pin by the non-electrical end, which goes against the component body. Press the **SPACEBAR** to rotate the pin while it is floating on the cursor.



◆ Only one end of a Pin is electrical – you must place the pins with this end out from the component body. You can easily identify the non-electrical end of the pin, it has the pin name next to it.

9. Before placing the pin press the **TAB** key to edit the pin attributes. Each attribute is described in the next topic. If you define the pin attributes before you place it, then the settings you define will become the defaults – this means that numeric pin names and numbers will auto-increment.
10. Continue to add the pins required to finish the component.

◆ Click and hold on a pin to re-position it after you place it, press the **SPACEBAR** to rotate it.

Component Pins

Component pins give a component its electrical properties. Pins have a number of attributes, which can be set in the Pin dialog. To set the attributes prior to placing the pin, press the TAB key while the pin is floating on the cursor. To set them after placing the pin, double-click on the pin, or click once on the pin in the pin list in the panel.

Pin Name

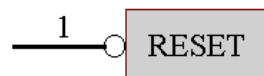
The pin name is optional, except when the pin is going to be hidden. A hidden pin is automatically connected to other hidden pins with the same *name*, and to nets with the same name, when the netlist is created.

Pin Number

Number of this component pin, each pin must have a unique number.

Dot symbol

Add a small circle (negation dot) to the end of the pin.



Clk Symbol

Add a small arrow (clock marker) to end of the pin.



Electrical Type

The Electrical Type attribute is used when you perform an Electrical Rule Check in the Schematic Sheet Editor. It is not used when creating a netlist. This attribute must be set correctly if you intend to use the Electrical Rule Check feature.

Hidden

Enable this option to define the pin as Hidden. A hidden pin is automatically connected to other hidden pins with the same *name*, and to nets with the same name, when the netlist is created. Power pins are often defined as Hidden. Once a pin is defined as hidden you must enable the Hidden Pins option in the panel to be able to see them (or select **View » Show Hidden Pins**).

Show Name

Show the pin name on the schematic sheet. It remains visible in the Library Editor.

Show Number

Show the pin number on the schematic sheet. It remains visible in the Library Editor.

Pin Length

Set the length of this pin, in hundreds of an inch.

Component Description

As well as its graphical definition, each component has a number of text fields associated with it. Select **Tools » Description** to pop up the Component Text Fields dialog.

Default Designator

A default prefix can be defined for the part designator. The default designator would typically take the form R?, C?, U? and so on.

Footprint

Four fields are provided for naming PCB footprints for the component. If these are not pre-assigned in the Library Editor a value can be entered when the part is placed on the sheet. Four fields allow you to nominate alternate patterns for SMD versions, etc.

Sheet Part Filename

Parts on the schematic sheet can be made to behave as sheet symbols rather than component parts. When they are in this mode, the nets connecting to their pins connect to matching ports on the sheet below. To get a part to behave as a sheet symbol, you specify the sheet that exists below in the Sheet Part Filename field and enable the *descend into sheet parts* option in the Netlist Creation dialog. The Sheet Part Filename can also be entered once the part is placed on the schematic sheet. Refer to the chapter *Multi-Sheet Design and Project Management* for more information.

Library Text Fields

Each library component has eight user-definable text fields. These fields can hold up to 255 characters each. Library text fields cannot be edited once the part has been placed in the schematic sheet, but can be viewed, and can also be included in the Bill of Materials.

Part Text Fields

The names of the sixteen part text fields can be defined in the Component Text Fields dialog. Up to 255 characters can be used, but only the end of the character string will be displayed in the Edit Part dialog if the length of the string exceeds the display area. There is room for 14 characters and/or spaces, when 12 point Helvetica is used as the dialog font.

Description

Use up to 255 characters for a text description of the component. Component searches can be done on the contents of the description field.

What is a Group of Components?

Many components share the same packaging – they have identical graphics and pin numbers, but exist as individual names in libraries. Perhaps these are identical devices from different manufacturers, or components that share the same packages but vary on some specification, such as a 120ns versus 80ns RAMs. While it is convenient to access these otherwise duplicate parts using either description, it would be inefficient to maintain a separate graphical version of each duplicate.

The Schematic Editor uses the concept of component *groups* to associate multiple component names with a single description stored in the library. This keeps libraries efficient and manageable. For example, while the TTL library contains nearly 1800

component names, the graphical and data descriptions that represent these components number only about six hundred.

Copying Components

Components can be copied within a library, or between libraries by selecting **Copy » Component**. In the Destination Library dialog you can select the same library, or another library. Once you have selected a library and clicked OK the component is copied. If there is only one library open this dialog does not pop up.

When you copy a component within a library you will notice that the component name appears twice in the list in the panel. You can then select **Tool » Rename Component** to give one of them a new name.

Components can also be copied between libraries, and from a library to a schematic using the right-click menu in the Schematic Library Editor panel. To copy between libraries select the component(s) in the Library Editor panel using the standard Windows selection keys (left-click, SHIFT+click and CTRL+click). Once the components have been selected click the right mouse button to pop up the floating menu and select **Copy**. Change to the target library, right-click in the Library Editor panel, and select **Paste** to add them to the target library.

Updating Your Schematic

After editing components in the Library Editor there are two ways of passing these updates across to your schematic sheets.

1. From the Library Editor – press the Update Schematics button in the panel (or select **Tools » Update Schematics** in the menus). This will update all instances of this component, on all open schematic sheets.
2. From the Schematic Sheet Editor – select **Design » Update Parts in Cache** to update all parts, on all open sheets, that are different from those in the libraries in the current Schematic Editor library list (not libraries open in the Library Editor).

Library Editor Reports

There are three reports that can be generated in the Library Editor.

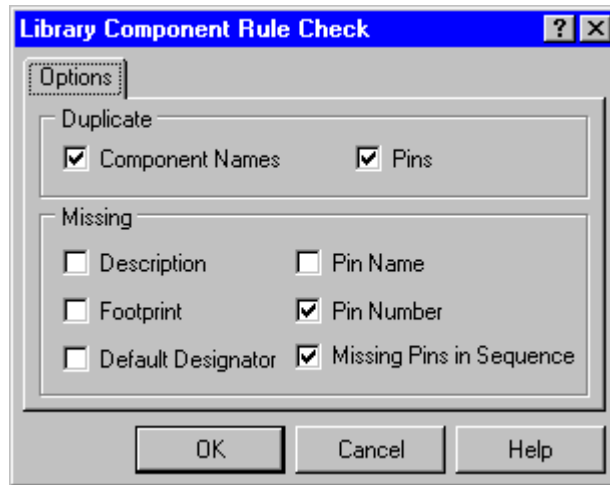
Component

This report lists all the information available for the active component. This includes; the number of parts, the group names and the pin details for each part in the component. This report has the file extension CMP.

Library

This report lists each component in the library and its description. This report has the file extension REP.

Component Rule Check



The Component Rule Check is used as an aid in component verification. Set the attributes you wish to test for, click OK and a report will be generated and opened in Text Expert. This report has the file extension ERR.

Multi-Sheet Design and Project Management

The Schematic Editor supports single sheet, multiple sheet and fully hierarchical designs including complex hierarchy, where multiple instances of a single sheet file can be used in a project.

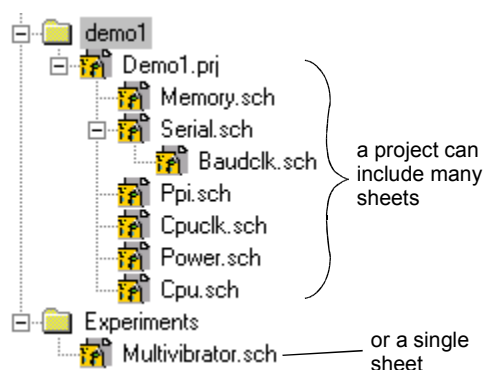
There is no limit on the number of sheets that you can have in a design, the limit will be defined by the amount of memory available in your PC.

Overview

Each schematic sheet is stored as an individual document. Any sheet can be opened and edited independently of all other sheets, simply click on the icon for the sheet in the Design Explorer Navigation Panel.

A schematic design can consist of a single drawing sheet, or multiple linked sheets.

Multiple sheet projects support large or complex designs that cannot be served by a single sheet. Even when the design is not particularly complex, there can be advantages in organizing the project across multiple sheets. For example, the design may include various modular elements. Maintaining these modules as individual documents allows several engineers to work on a project at the same time. Another reason to organize a project across multiple sheets is that this method allows you to use small format printing, such as laser printers.



When two or more sheet files are associated or linked in some way, we refer to this as a *multi-sheet* project. There are a number of methods for organizing multiple sheet projects. Choosing one approach or another is based upon the type, size and structure of the project.

Managing Multiple Sheet Projects

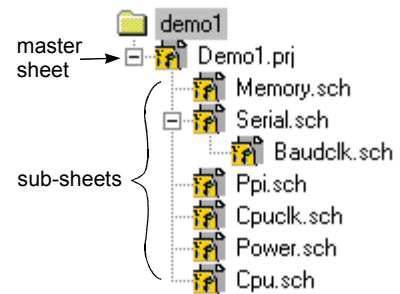
Project management is the process of defining and maintaining the links between individual sheets that comprise a project. These inter-sheet links establish the connectivity, as well as providing other project management benefits. For example, these links support navigating, viewing and accessing each individual sheet in a project, as well as multi-sheet netlist generation and Electrical Rules Checks (ERC).

Sheet-to-sheet links also allow you to open and save entire projects in a single operation, as well as perform global editing and text searches across multiple sheets.

- ◆ Many schematic features, such as re-annotation and printing, apply to the open sheets. When you perform these operations the entire project (includes the master sheet and all associated sheets) will automatically be opened.

Structure of a Multi Sheet Schematic

A major benefit of the Schematic Editor's multi-sheet environment is an intuitive system that makes management and navigation of even complex hierarchies easy. All multi-sheet projects include a special sheet file called the *master sheet*. The master sheet is the top, or first sheet in the design *hierarchy*. The term hierarchy refers to the relationship between master sheets and *sub-sheets* that make up the project.



A special class of hierarchy, called *complex hierarchy*, allows you to use multiple references to the one sub-sheet. This hierarchical structure can have a number of forms, defined by the method used to connect the sheets. Refer to the topic *The Different Methods of Structuring a Multi-Sheet Design* later in this chapter for more information on using complex hierarchy.

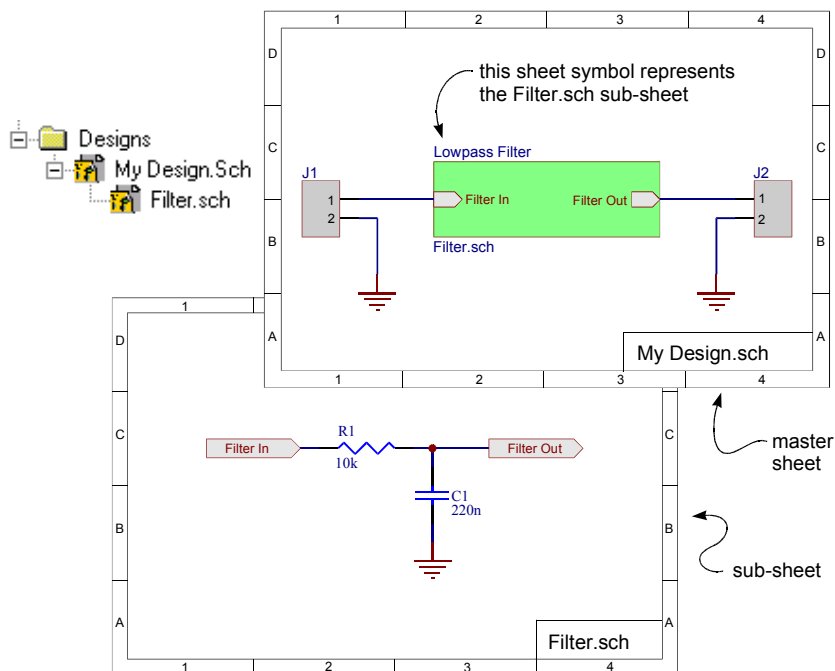
- ◆ Hierarchical organization supports a truly modular approach by allowing you to work with functional blocks. These blocks have a spatial relationship on the sheets which support both “top down” and “bottom up” design methodologies.

Master Sheets and Sub-Sheets

The relationship between sheets in a multi-sheet design is formed by a special symbol called a *sheet symbol*. The sheet symbol provides a graphical representation of the sheet that it represents, which is referred to as a sub-sheet.

Along with its graphical display attributes (color, size, location, etc) the sheet symbol has two additional fields: a *Name* field and a *Filename* field. The Name field is a text label, and is provided for reference only. The Filename refers to the specific sheet file that the sheet symbol represents. This field creates the link between the master sheet and its sub-sheet, and thereby defines a project.

This association of sheets resides on two levels. At the project level, the association is maintained by the presence of sheet symbols in the master sheet. Projects are also defined at the electrical (or connective) level by *net identifiers*. Net identifiers provide the links that connect circuits across multiple sheets.



Adding a Sub-Sheet to a Project

To add an open schematic sheet to a project place a sheet symbol in the master sheet. Double-click to edit the sheet symbol and enter the name of the sub-sheet in the Filename field (eg, *SubSheet1.sch*). The Project Manager will be updated to show the new sheet linked into the project.

How Connectivity is Created in a Multi-Sheet Design

The Schematic Editor provides a number of ways of creating connections between sheets in a multi-sheet design. The net-to-net connections are created by *Net Identifiers*, each of these is described below.

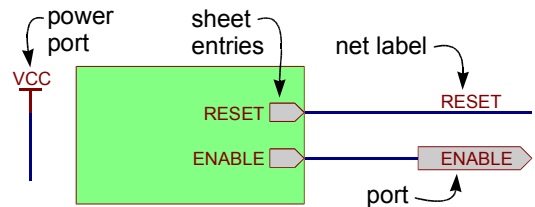
When you create a netlist you must instruct the Schematic Editor how you want these net identifiers to connect – you must specify the *Net Identifier Scope*. There is essentially two ways of connecting net identifiers: either *horizontally*, directly from one sub-sheet to another sub-sheet; or *vertically*, from a sub-sheet to its parent sheet. In horizontal connectivity the connections are from net label to net label, or port to port. In vertical connectivity the connections are from sheet entry to port.

Management of hierarchical multi-sheet schematics may seem complicated at first. However, once a few basic principles are understood it is relatively easy to use the power of hierarchy to organize complex projects.

Net Identifiers

Net identifiers provide the connective “glue” between nets in schematic sheets.

Net identifiers are placed to “connect” objects belonging to a common net, either inside a single schematic sheet or across two or more sheets in a hierarchical project. The connections can be physical (when the object is attached directly to another electrical object, e.g. “wired”) or logical (when the net identifier indicates a link to another net which has the same name).



Four of the five possible net identifier objects are illustrated. Hidden pins on schematic parts provide a fifth method for identifying nets.

Net identifiers include:

Net Label

Use a net label to uniquely identify a net. This net will connect to other nets of the same name on the same sheet, and can also connect to nets of the same name on different sheets. Net Labels are attached to individual wires, part pins and buses.

Port

Depending on the method of connectivity, a port can connect horizontally to other ports with the same name, or vertically to a sheet entry with the same name.

Sheet Entry

When the connectivity is vertical you can use a sheet entry to connect to a port of the same name, on the sheet below.

Power Port

All power ports with the same name are connected throughout the entire project.

Hidden Pin

Hidden pins behave like power ports, connecting globally to nets of the same name throughout the entire project.

◆ Bus connectivity is created by the correct combination of net labels and ports. Refer to the *Schematic Design Objects* chapter for details on connecting with buses.

Negating Net Identifiers

There are 2 methods that can be used to negate (include a bar over the top of) a net label, sheet entry or port. The first is to include a backslash character before each letter in the net name (eg “\n\e\t”). The second method is to enable the Single \ Negation option in the Preferences dialog, and include one backslash character at the start of the net name (eg “\net”).

The Scope of Net Identifiers

The connectivity that is created when you generate a netlist for your multi-sheet project depends on the *scope* of the net identifiers. The scope specifies how you want the net identifiers to connect: *local* – within the sheet; *global* – across all sheets; *sheet symbol/port connections* – connect vertically between a port and the matching sheet entry.

For example if net labels are local then a net labeled Clock1 on one sheet will not connect to a net named Clock1 on another sheet. In this case the scope of net labels is *local*.

Net labels can also be specified to be *global* net identifiers. If they are then all instances of the net label *Clock1* (on all sheets in the project) would be deemed to be part of the same net. Two special net identifier objects are always deemed to be global: power ports and hidden pins.

The *scope of net identifiers* is specified when you generate a netlist (**Design » Create Netlist**) or when you run an electrical rules check (**Tools » ERC**).

The scope of net identifiers must be determined at the beginning of the project. The different methods of structuring the project are described in the following topic.

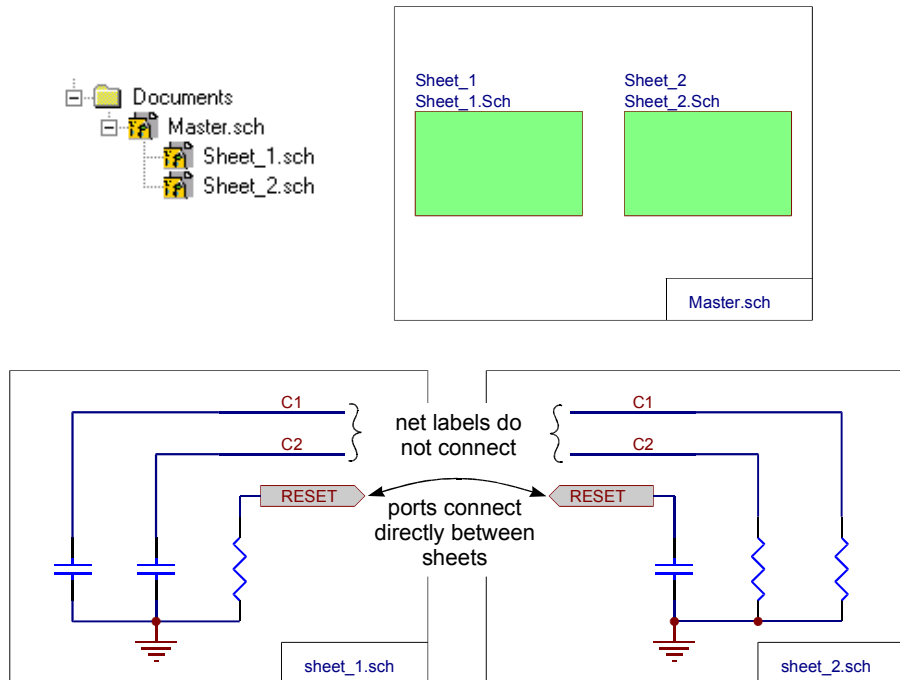
The Different Methods of Structuring a Multi-Sheet Design

The relationship between project sheets and net identifiers is best described by illustrating the five possible models of Schematic project organization.

It is important to remember that all multi-sheet schematic projects are organized hierarchically, even if the hierarchy is limited to two levels, where a single master sheet holds sheet symbols to all sub-sheets. Models 1 and 2 below show the “flat” model supported by Orcad (Model 1) and Protel DOS Schematic (Model 2).

- ◆ Every multi-sheet project must include a master sheet, with a sheet symbol for each sub-sheet. This allows identification of all sheets associated with a single project.

Model 1 - Global Ports Define Inter-Sheet Connections



This model of hierarchy represents a “flat” design. The master sheet and sheet symbols simply provide a means of identifying all project sheets. Ports are the only method used to link between sub-sheets, and this linking is “horizontal”. All circuit information is in the sub-sheets.

Model 1 is called a “flat” design because all the sub-sheets are on the same level in the project hierarchy. The top sheet includes the sheet symbols for all the sub-sheets, but it does not include any wiring or circuitry.

In this model all the inter-sheet connections are made *globally* through ports, where ports of the same name are connected throughout the project. Note that the net names in the two sub-sheets are *local*, meaning that the net name connects only within each sheet, not to other sheets in the project.

This model treats your design as though it were laid out on a single large sheet, that has been cut into individual pages. While this approach works well for designs that are not large, management of large designs can be awkward because it is difficult to trace a net from one sheet to another. To simplify tracing a net in this style of design you can include cross-sheet port references on the sheets.

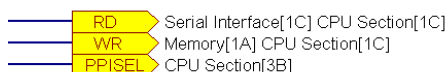
Cross-Sheet Port References

You can include sheet-to-sheet port referencing on your schematic, in one of two ways. The two options are in the Schematic Editor **Reports** menu and include:

Add Port References (Flat) – select this to include a string next to each port, detailing the sheet name and grid reference of all other ports of the same name in the project.

Add Port References (Hierarchical) – select this to include a string next to each port, detailing the sheet name and grid reference of the sheet entry that this port connects to.

Port references can be removed at any time by selecting **Reports » Remove Port References** from the menus. Port references are a calculated attribute of the port, they can not be edited and are not stored with the design. Their location is determined by the location of the port on the sheet and the position of the connecting wire.

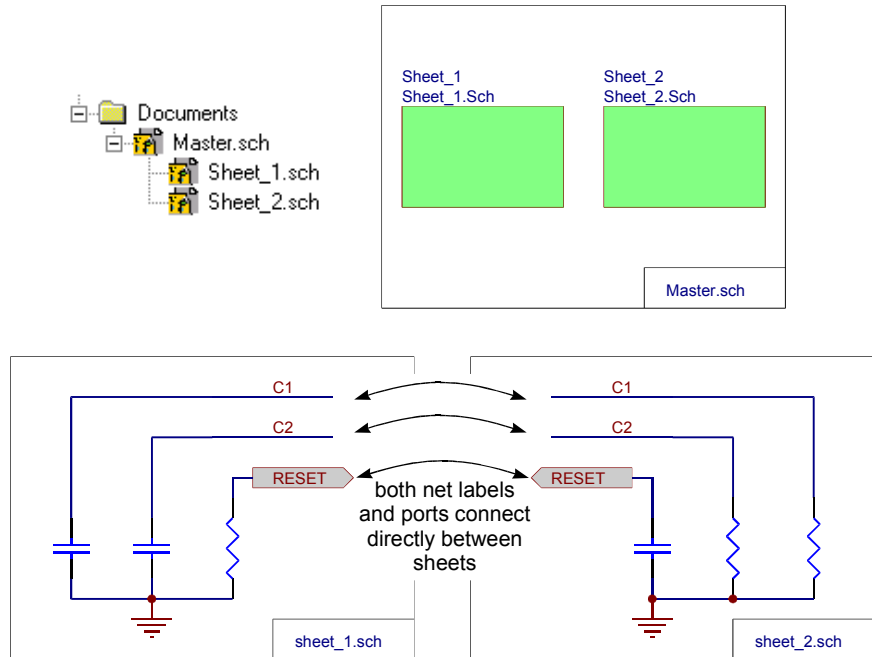


*Include port referencing to track
connections between sheets*

Setting the Net Identifier Scope

When performing an ERC or using the synchronizer to update the PCB set the Net Identifier Scope to *Only Ports Global* for this model.

Model 2 - Global Net Labels and Ports



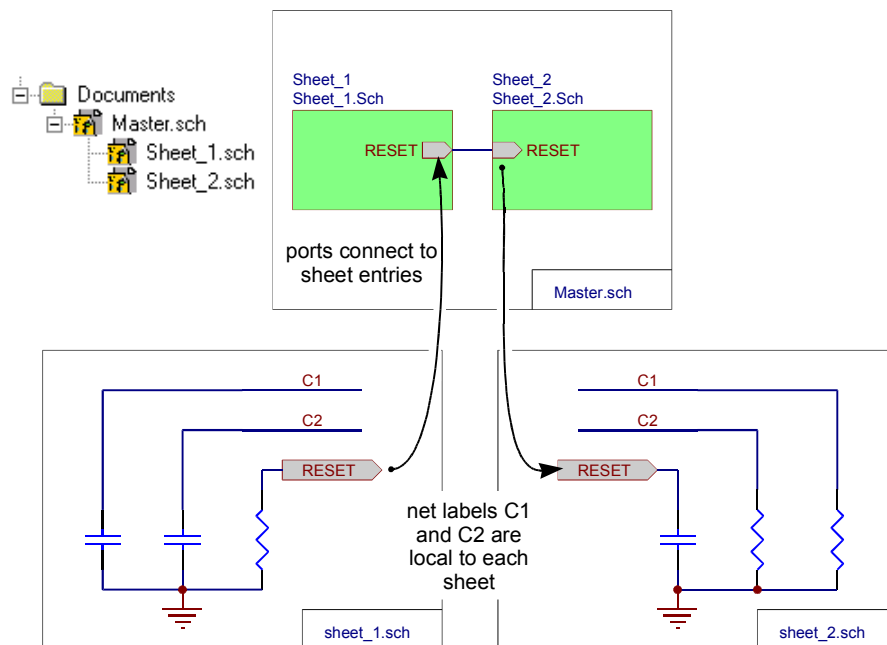
This model of hierarchy represents a “flat” design. The master sheet and sheet symbols simply provide a means of identifying all project sheets. In this model both ports and net labels can be used to link between sub-sheets, and this linking is “horizontal”. All circuit information is in the sub-sheets.

Model 2 is also called a “flat” design because all the sub-sheets are on the same level in the project hierarchy. The top sheet includes the sheet symbols for all the sub-sheets, but it does not include any wiring or circuitry. In model 2 the inter-sheet connections are created by both net labels and ports.

Setting the Net Identifier Scope

When performing an ERC or using the synchronizer to update the PCB set the Net Identifier Scope to *Net Labels and Ports Global* for this model.

Model 3 - Simple Hierarchy



This model of hierarchy represents “simple hierarchy” where each sheet appears once in a design. Hierarchy is implemented by each sheet entry connecting to a port of the same name on the sub-sheet below. Nets labels are local.

The third model is referred to as *simple* hierarchy. This model supports multi-level or *block* design, where the design hierarchy can be represented by a tree-like structure. In this model the sheet symbol represents a *child* sheet, which *descends* from the *parent*.

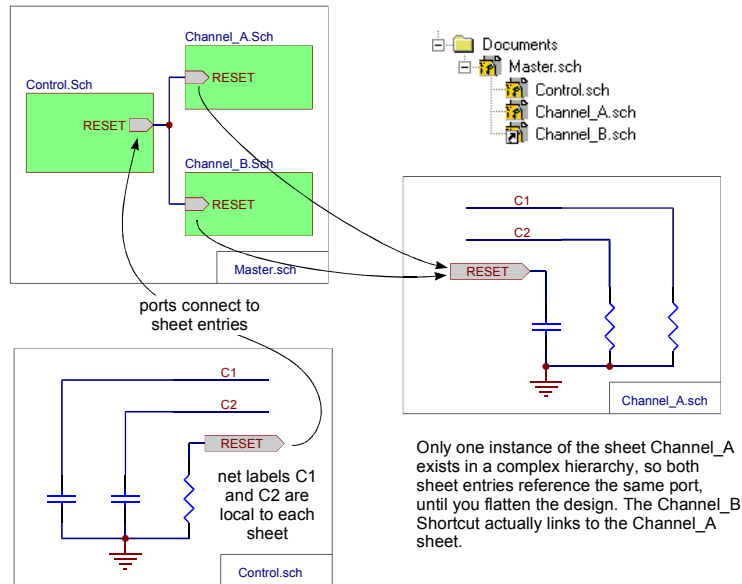
All the inter-sheet connections are vertical, the sheet entries in each sheet symbol are connected to similarly named ports on the respective sub-sheets. The sheet symbol-to-sheet symbol wiring is included on the parent sheet.

- ◆ Model 3 is a true hierarchy because the inter-sheet connections follow the hierarchy of the sheets themselves, and the design can be as many levels “deep” as you like. This is the recommended model to use for a multi-sheet design.

Setting the Net Identifier Scope

When performing an ERC or using the synchronizer to update the PCB set the Net Identifier Scope to *Sheet Symbols/Port Connections* for this model.

Model 4 - Complex Hierarchy



This model of hierarchy represents “complex hierarchy”, where a sub-sheet appears more than once in a project. Hierarchy is implemented by each sheet entry connecting to a port of the same name on the sub-sheet below. Nets labels are local.

The fourth model is referred to as *complex* hierarchy. In this model multiple sheet symbols reference the same sub-sheet.

To establish the complex hierarchy you must create a document shortcut for each duplicate sheet symbol. Consider the figure above. Channel_A.sch was created and referenced by the Channel_A sheet symbol. This sheet symbol was copied and pasted on the schematic sheet, and the schematic filename field for the copy was changed to Channel_B.sch.

The document icon for Channel_A.sch was then copied (right-click on the document icon in the folder that contains the Channel_A schematic and select **Copy** from the floating menu), and pasted as a shortcut (right-click in free space in the same folder and select **Paste Shortcut** from the floating menu).

After creating the document shortcut icon, the hierarchy can be rebuilt by selecting **View » Refresh** from the Folder menus.

This model fits projects which are highly modular. An example would be a stereo amplifier, where left and right channels are identical circuits.

Converting From Complex Hierarchy to Simple Hierarchy

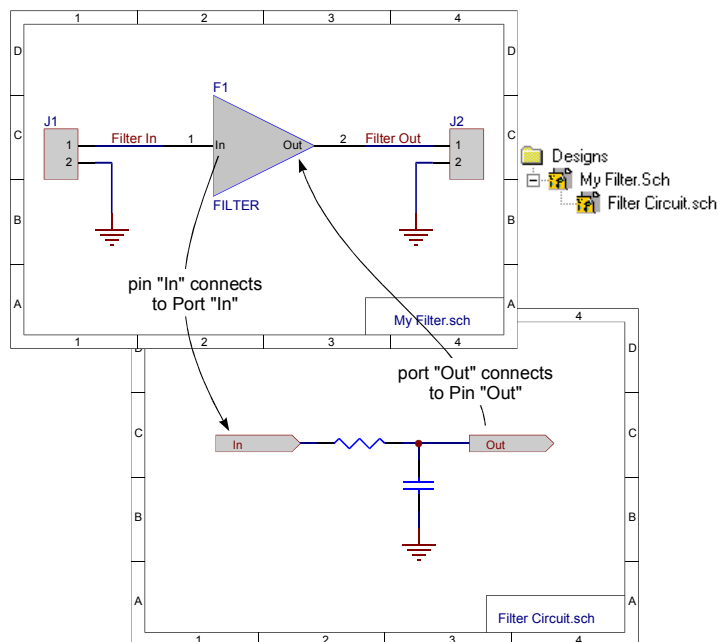
Complex hierarchy is used during the schematic capture phase. When you are ready to create the netlist you must “flatten” the design, converting it from complex to simple.

Select **Tools » Complex to Simple** to flatten the design. Each child sheet that is used more than once will be copied and renamed. You must also re-annotate the design to assign a unique designator to each part.

Setting the Net Identifier Scope

Model 4 also uses the *Sheet Symbols/Port Connections* Net Identifier Scope when performing an ERC or synchronizing (after converting the design from complex to simple).

Model 5 - Using Sheet Parts to Create Hierarchy



Model 5 allows you to use components to behave as sheet symbols

This is a special model of hierarchy where a component behaves as a sheet symbol, referencing circuitry below it. There are a number of situations where you may wish to use this feature: perhaps your design includes a small daughter board and you want to design and analyze the entire circuit; or a programmable device, where you want to include the circuit inside the device in the overall design; or perhaps you simply want to represent the circuitry below symbolically rather than with a rectangular sheet symbol.

At netlist creation time you have the option to either include the circuitry below in the netlist (descend into the sheet parts), where this special component is not included in

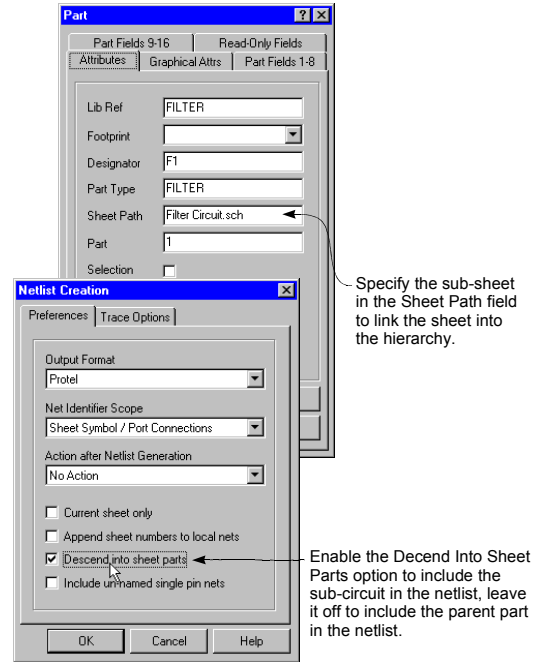
the netlist but the circuitry below it is; or to treat these parts as components, where they are included in the netlist, but the circuitry below is not.

There are two steps to using this model:

1. Enter the name of the sub-sheet in the Sheet Path field of the Part dialog on the parent sheet.
2. Enable the Descend into Sheet Parts option in the Netlist Creation dialog. All sheets must be in the same folder if you use this option.

The pins will behave as sheet entries and connect to ports of the same name on the sheet specified in the Sheet Path field.

You can also specify the sub-sheet filename in the library editor when you create the Part, select **Tools » Description** in the Library Editor to edit the Sheet Part Filename.



Setting the Net Identifier Scope

Model 5 also uses the *Sheet Symbols/Port Connections* Net Identifier scope when creating a netlist or performing an ERC. Remember to enable the *Descend Into Sheet Parts* option when creating a netlist or performing an ERC.

Summarizing Hierarchical Design

Models 3, 4 and 5 illustrate powerful ways to organize complex projects. In hierarchical designs, sheet symbols can represent functional blocks, with the sheet entries serving as connectors that tie circuitry on the sheet to the sub-sheet.

This hierarchical structure can be represented by thinking of the first sheet as the “parent” and the sheet represented by sheet symbols as the “child.” In the terminology of hierarchical design, we can say that the child is descended from the parent. Extending this model, the child can have its own “children,” – additional sheets that descend, in this top-down structure, to lower and lower levels.

As shown above, hierarchy can be either simple, where each sheet is unique, as in Model 3 or complex, where the same children (and its children) appear more than once in the design – a modular approach illustrated by Model 4.

Working With a Hierarchical Project

The Schematic Editor includes a number of tools to make it easy to work with multi-sheet designs. These include tools for navigating, tools for creating sub-sheets and sheet symbols and a tool to convert complex hierarchy to simple hierarchy.

Navigating Through a Project

To make hierarchical design practical, it is necessary to have some means of tracking and navigating the sometimes complicated relationships between the many sheets that can be incorporated into a single project. There are two tools provided for this purpose; the Design Explorer Navigation Panel, and the Up/Down Hierarchy button.

Using the Design Explorer Navigation Panel

Simply click on any sheet icon in the Navigation Panel to open that sheet and make it the active sheet.

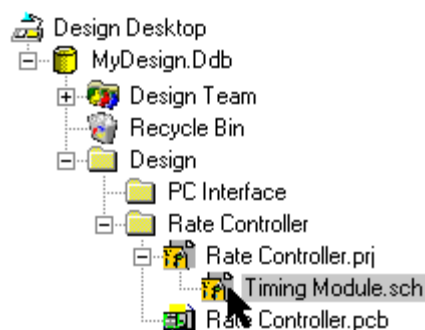


Using the Up/Down Hierarchy Button

There is also a button on the main toolbar for navigating up and down through the hierarchy.

When you click on the Up/Down Hierarchy button the Status line will prompt you to select a port, sheet entry, sheet symbol or sheet part. If you click on a sheet entry you will be presented with the matching port on the sub-sheet, if you click on a sheet symbol or a sheet part you will be presented with the entire sub-sheet.

To navigate up through the hierarchy, click a port to be presented with the matching sheet entry on the parent sheet.



Creating Sheet Symbols and Sub-Sheets – the Easy Way

The Schematic Editor includes features to “automate” the project building process.

Top-Down Design

If you are designing in a top-down fashion, where you start with the top sheet and lay the design out as functional blocks using sheet symbols, you can use the **Design » Create Sheet From Symbol** feature to automatically create the sub-sheets.

When you select this menu item you will be prompted to select a sheet symbol. After clicking on a sheet symbol the Schematic Editor will open a new schematic sheet with the correct file name. The new sub-sheet will include ports to match each of the sheet entries on the sheet symbol you selected.

Bottom-Up Design

If you are designing in a bottom-up fashion, where the sub-sheet already exists and you need to create a sheet symbol to represent it, select the **Design » Create Symbol From Sheet** menu item (the parent sheet must be the active sheet). The Choose Document to Place dialog will pop up.

Select the sheet you would like to base your sheet symbol on. You will then be asked if you want to Reverse Input/Output Directions. After you answer this, you will be presented with a sheet symbol floating on the cursor. This sheet symbol will have the correct Filename to link it to the sub-sheet, and will have sheet entries to match each of the ports on the sub-sheet.

To respond to the Reverse Input/Output Directions question - each of the ports on the sub-sheet has an I/O type. Say one of the ports has an I/O type of *output*. If you respond “yes” to the Reverse Input/Output Directions question then the sheet entry that matches this port will have an I/O type of *input* (the I/O direction has been reversed) and will be positioned on the left of the sheet symbol. If you respond “no” to the Reverse Input/Output Directions question then the sheet entry that matches this port will have an I/O type of *output* and will be positioned on the right of the sheet symbol.

Schematic Design Verification

Design verification is the process of ensuring that the schematic capture process has produced an accurate “snap shot” of your design, from which a valid netlist can be created. The Schematic Editor includes a tool specifically for this purpose, the Electrical Rule Checker (ERC). This process examines the schematic for both electrical inconsistencies, such as an output pin connected to an output pin, and drafting inconsistencies, such as unconnected net labels or duplicate designators.

Select the **Tools » ERC** menu item to pop up the Setup Electrical Rule Check dialog. This is used to set-up and execute the Electrical Rule Check. Running the ERC produces two results. First, a text report is generated, listing the electrical and logical violations either for the active sheet, or the entire project. Secondly, error markers are placed on the sheets at the site of each ERC violation as an aid in tracking and correcting reported problems.

Marking Points That You Do Not Want Flagged as Errors

There may be points in the design which you know will be flagged as ERC errors or warnings, which you do not want to be flagged. To suppress these, place a No ERC directive at each point (**Place » Directive » No ERC**). These directives can be excluded from the printout by disabling the Include on Printout option in the Schematic Printer Setup dialog.

Verification Options

A wide variety of basic electrical errors are reported. For example, floating input pins on parts and shorts between two differently named power nets.

Setup Electrical Rules Check

When the **Tools » ERC** menu item is selected, the Setup Electrical Rule Check dialog pops up. This dialog is used to define the options, scope and parameters of the electrical rules check. Options include:

Multiple Net Names On Net

Reports physical nets with multiple net identifiers with different names.

Unconnected Net Labels

Reports net labels that are not physically connected to at least one other electrical object in the sheet.

Unconnected Power Objects

Reports power objects that are not physically connected to at least one other electrical object.

Duplicate Sheet Numbers

Reports sheets that have been assigned the same sheet number (**Design » Options** dialog, Organization Tab).

Duplicate Component Designators

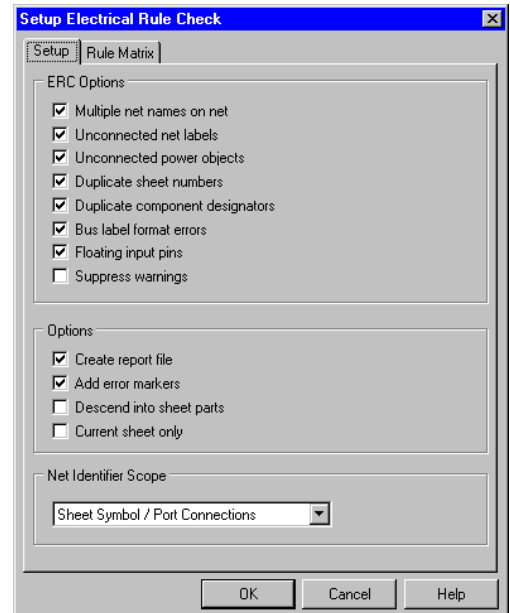
Reports parts that have identical designator labels. This condition can occur when the Annotate process launcher has not been used or when the **Tools » Complex to Simple** process launcher has not been used to “flatten” a complex hierarchical project (project with duplicate sheets).

Bus Label Format Errors

Reports net labels attached to buses which are not legally formatted to reflect a range of signals. Logical connectivity for buses can be assigned by placing a net label on the bus. Generally, this net label will include all bus signals, e.g. HA[0..19] represents nets named HA0, HA1, HA2, etc, to HA19.

Floating Input Pins

Reports unconnected pins whose Electrical Type is set to Input.



Suppress Warnings

This option generates a report and error markers for error conditions only. Warning conditions (see Connected Pin / Sheet Entry / Port Rule Matrix below) are ignored. This allows you to perform a quick ERC for all error-level problems.

Other Options

Create Report File

This option generates a text report listing all ERC report information.

Add Error Markers

This option places special error markers on the sheets, at the site of each reported warning or error. Special facilities in the sheet editor allow you to jump from error marker to error marker.

Descend Into Sheet Parts

This options treats *sheet parts* as hierarchical sheet symbols. A sheet part is a part which is specified to behave like a sheet symbol, where its pins are associated with ports on a sheet which descends hierarchically from the sheet part. The “descending” sheet is defined in the part’s Sheet Part path field. Refer to the chapter *Multi-Sheet Design and Project Management* for more information.

Setting the Net Identifier Scope

The net identifier scope determines the method of connectivity being used in a multi-sheet design. The net identifier scope chosen here should be the same as that chosen when creating the netlist.

For a complete explanation of the net identifier scope refer to the *Creating a Netlist* and *Multi-Sheet Design and Project Management* chapters.

Setting up the Electrical Rules Matrix

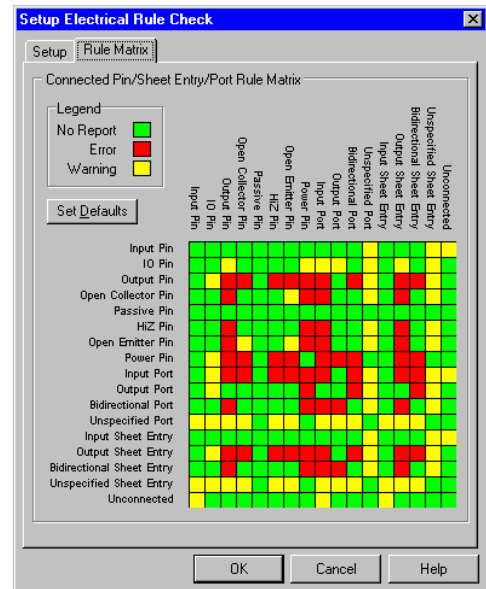
This matrix allows you to control the behavior of the Electrical Rules Check. By setting the color of a matrix element you set how you want the Schematic Editor to respond when it is testing for a particular condition.

The matrix is read in an across / down fashion. For example, to see what the report condition for an input pin connected to an output pin would be, locate the input pin row down the left of the matrix, then locate the output pin column across the top of the matrix, then go across and down from each to where these two meet. The colored square where they meet is green, signifying that there will be no warning if an input pin is connected to an output pin, as you would expect. Similarly if you examine the report condition for an output pin connected to an output pin, you will see that the square is red, signifying that if this condition is detected it will be flagged as an error.

You can specify either errors or warnings for connections of pin types, ports and sheet entries. Generally you will find the default matrix setting appropriate, but you can easily adjust them for your specific design requirements.

To change the default settings in the matrix click LEFT MOUSE in any matrix square. With each click the square will toggle from No Report (green), to Warning (yellow), to Error (red), then back to No Report, and so on.

The process of running an ERC is integral to producing a valid netlist for a project. The presence of electrical or logical violations will not prevent the Schematic Editor from generating a netlist, however incomplete or invalid. Carefully check and resolve all reported errors prior to netlist generation.



Error Report Format

```
Error Report For : C:\CLIENT\SCH3\EXAMPLES\DEMO1.ERC      30-Dec-1997
15:18:05
1 Error  Duplicate Designators POWER.SCH C3 At (320,471) And CPU.SCH C3
At (374,109)
2 Error  Multiple Net Identifiers : CPU.SCH RESET At (270,220) And
CPU.SCH RST At (330,220)
3 Warning Unconnected Input Pin On Net N00121
C:\CLIENT\SCH3\EXAMPLES\CPU.SCH(U5-6 310,620)
4 Error  Floating Input Pins On Net N00121Pin
C:\CLIENT\SCH3\EXAMPLES\CPU.SCH(U5-6 @310,620)
5 Warning Unconnected Net Label On Net CLOCK   CPU.SCH CLOCK
End Report
```

The error report includes information to identify which sheet the error or warning occurred on, and the location on the sheet. Where appropriate there will also be net identifier and component designator information.

Resolving Errors

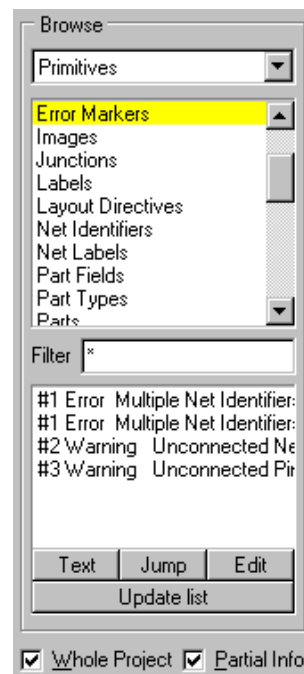
The Schematic Editor includes features to assist in the process of resolving the errors and warnings detected by the Electrical Rule Check.

Using the Browser to Jump to Errors

The lower half of the Schematic Editor panel includes a Browser. This browser can be used to browse through your project for any type of primitive, including error markers.

In the Browser, set the Browse type to Error Markers. Select an error in the list and press the Jump button to have the error marker presented in the center of the active window.

Note that a description of the error condition appears on the Status Line, alleviating the need to switch back and forth to the error report. Pressing the Text button in the Component Browser will pop up a dialog which can also be used to examine the description of the error condition.



Cross Probing



The Schematic Editor supports cross probing to and from other open documents. Cross probing is particularly useful when you want to go from the schematic to the error report. Simply click on the cross probe button, then click on the error marker. The warning or error will be highlighted in the error report.

Typical Causes of Errors

Errors will typically be due to the one or more of the following:

- drafting errors - wires overlapping pins, lines being used instead of wires, the design being wired with the snap grid off so the wire ends don't touch the pin ends or wires / buses finishing under a port instead of touching the end of the port.
- syntax errors - net identifiers with spelling mistakes or buses incorrectly labeled.
- component errors - component pins placed the wrong way around on the component or pins with an inappropriate Electrical Type.
- design errors - a design condition that the ERC detects as an error, such as two output pins connected.

Tracing Errors

To resolve an error, start at the error marker, read the error condition on the Status line and consider the possible causes as described above. If there are no problems at the error marker, trace the net through the design. Use the Up/Down Hierarchy button on the main toolbar to assist tracing a net through a multi-sheet design. Use the Status line for information on what to do after selecting one of these tools.



- ◆ A floating input pin is essentially an “open circuit” condition. While this condition is reported at the input pin, the break could be anywhere between the driving output pin and the floating input pin. If there is nothing wrong at the input pin, trace the net back to the output pin, checking along the way for the “break”.
- ◆ If all the nets of a bus are reporting an error, look for the problem at the bus level, perhaps a typing mistake in a bus port or a missing net label, perhaps a bus line finishing under a port (which may not be visible). Click on a bus to focus it to see if it ends under a port.

Preparing the Design for PCB Layout

You have completed the schematic and you are ready to transfer the design information to the PCB Editor, to begin the board design. Before you do, use this chapter as a final check to ensure that there is nothing you have overlooked.

Assigning and Re-assigning Designators

All parts in your design must have a unique component designator. The process of re-assigning designators in your design is called *annotation*. This can be done at any stage in the design process, and would typically be done when the design is complete. Select **Tools » Annotate** to reassign the schematic designators. Here you can annotate:

All Parts

Reassign the designators for all parts in the project. This option does not re-use any currently assigned designators.

? Parts

Assign a designator to all parts which currently have a designator of R?, C?, U?, etc.

Reset Designators

sets all designators back to R?, C?, U? and so on. Use this if you wish to renumber all components, starting at 1.

If you have used complex hierarchy in your design you must first convert the design to simple hierarchy, and then re-assign the designators. Refer to the topic, *Model 4 - Complex Hierarchy*, in the chapter, *Multi-Sheet Design and Project Management*, for more information on working with complex hierarchy.

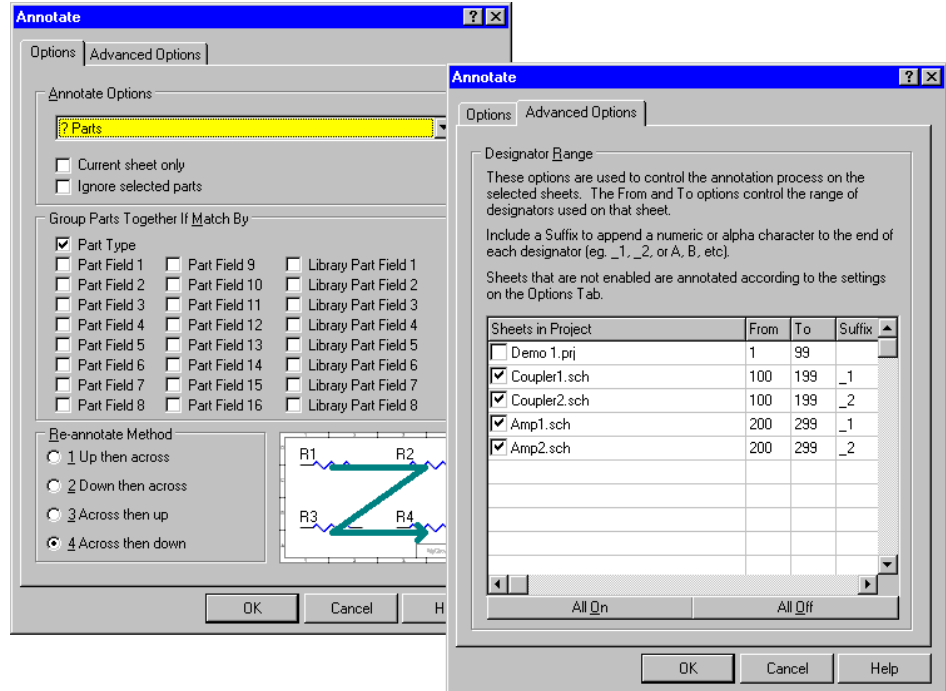
Grouping Parts into the Same Physical Component

To know how to identify and group multi-part components, say, how to package two particular op amps into the same physical component, use the Group Parts Together If Match By fields in the Annotate dialog. The default is to only group by the Part Type field, however combinations of any of the sixteen part fields and eight library text fields can also be used to identify a group.

For the example of two op amps that must be grouped, edit each one and enter a string into one of the part fields that identifies them. For example, you might enter “Stage1” into Part Field 16 for both the op amps. When you perform the annotation, enable Part Field 16 in the Group Parts Together If Match By check boxes – this will ensure that these two op amps are grouped into the same physical component.

Positional Annotation

Schematic components are annotated based on their position on the sheet. The position of each component is determined by the location of its designator within the reference zones around the edge of the sheet. To change the number of sheet border regions enable the Use Custom Style option in the Document Options dialog, and change the X Ref Region Count and the Y Ref Region Count settings.



Use the options in the Annotate dialog to control the annotation process

Controlled Sheet-by-Sheet Annotation

Schematic component annotation can also be selectively controlled for each sheet, with a user-defined designator range and/or suffix. To do this disable the Current Sheet Only option in the Options Tab, then click on the Advanced Options Tab. This Tab will list all the sheets in the project. Enable the check box next to each sheet that requires controlled annotation (any sheet that is not checked is annotated in the normal way according to the settings in the Annotate dialog's main Options Tab).

There are 2 controlled annotation methods, by Range and by Suffix. To annotate by range enter a From and To value for each sheet, for example From=1, To=1000. Note that the annotation process does not permit duplicate designators, you must include a unique suffix if the range is repeated for more than one sheet.

The Suffix options allow you to repeat the range for multiple sheets. Both alpha suffixes (A, B, C, etc) and numeric suffixes (_1, _2, _3, etc) are supported. Use these annotation options if you are annotating a multi-channel design and you want common in-channel numbering across all channels, for example, R1_1, R2_1, C1_1, etc, for channel 1, then R1_2, R2_2, C1_2, etc, for channel 2, and so on.

Excluding Components from the Annotation Process

Components can be excluded from the annotation process by selecting them. Enable the Ignore Selected Parts option in the Annotate dialog to exclude these parts from the annotation process.

Check for Missing Footprints

All parts in the design must have a footprint specified (a footprint with this name must exist in an available PCB library when you transfer the design information to the PCB).

The easiest way to check for missing footprints is to export the schematic footprint information into a spreadsheet. To do this select **Edit » Export to Spread** from the menus. The Schematic Export Wizard will pop up.

The Wizard allows you to selectively export information from the schematic. You will only need to export the Designator and Footprint attributes from the Part Primitives.

The information will automatically appear in a spreadsheet. Here you can quickly locate footprint fields without a value. If you prefer you can enter the footprint information in the spreadsheet and update the schematic (select **File » Update** in the Spreadsheet menus), or you can edit the parts in the schematic.

Perform an ERC

You should perform an Electrical Rules Check as a normal part of the design cycle. This will detect both drafting and electrical problems with your design, and will help prevent problems occurring later in the design process. Read the *Design Verification* chapter for tips on performing an ERC and resolving the errors.

Including PCB Layout Specifications

You can include PCB layout specifications in the schematic, by attaching PCB Layout directives to the schematic nets. Select **Place » Directives » PCB Layout** from the menus to place a layout directive. The PCB Layout directive information is translated into design rules when the design is transferred to the PCB Editor if the Generate PCB Rules option is enabled in the Update Design dialog – refer to the next chapter, *Passing the Design Information to the PCB*, for details on how to do this. After placing the directive (so that the bottom point of the symbol is touching the net), edit it to define:

Track Width – specify the width that this net must be routed at.

Via Width – specify the diameter of any vias used when routing this net.

Topology – specify the connection topology to be used with this net.

Priority – specify the routing priority for this net.

Layer – use this to restrict the routing of this net to a particular layer.

Ready for PCB Layout

You are now ready to begin the PCB layout stage of the design process. You can either create a new empty PCB document, then manually define the board boundary, or use the PCBMaker Wizard (in the Wizards Tab of the New Document dialog). Refer to the chapter, *Defining the Board*, in the *PCB Design* section of the Handbook, for more information.

After Defining the board boundary the design information can then be transferred from the schematic to the PCB – refer to the next chapter, *Passing the Design Information to the PCB*, for details on how to this.

Transferring the Design Information to the PCB

Protel 99 SE includes a powerful design synchronization tool, that makes it very easy to transfer design information from the schematic to the PCB (and back again).

The Synchronizer automatically extracts the component and connectivity information from the schematic, locates the footprints in the PCB libraries and places them in the PCB workspace, then adds the connection lines between connected component pins.

Design Synchronization is the process of updating the *Target* document, based on the latest design information in the *Reference* document. When you run the synchronizer you tell it which direction you want to transfer by picking:

- ◆ **Update PCB** to transfer design changes from the schematic, to the PCB
- ◆ **Update Schematic** to transfer design changes from the PCB, to the Schematic

Transferring the Design Information

To transfer the schematic design information select **Design » Update PCB** from the Schematic Editor menus. When you do the Update dialog will appear, with the name of the target document in the title.

It does not matter which of the schematic sheets in your project is active when you select **Update PCB**, the synchronizer will automatically analyze all the sheets in the project.

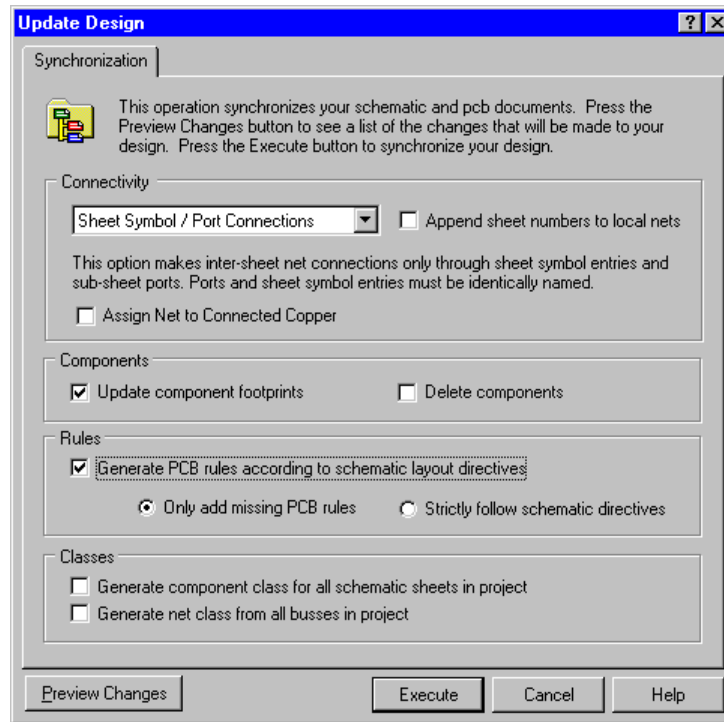
◆ An empty PCB is automatically created if there is no PCB available.

Check for Warnings

When you select **Design » Update PCB** from the menus the Synchronizer quickly checks the schematic for potential issues that may cause problems in transferring the design information. It checks for; components that do not have their designator set, components with duplicate designators, components with no footprint specified, and if there are any PCB libraries currently available in the PCB Editor.

If there are any warnings the Update dialog will include an extra Tab, summarizing the different warnings. Check these before carrying out a Synchronization. For complete details of the warnings press the Report button at the bottom of the Warnings Tab. The previous chapter, *Preparing the design for PCB Layout*, includes information that will help resolve these warnings, including how to re-annotate the designators, and how to locate missing footprints.

The Synchronization Update Dialog



The Update Design dialog includes the following options:

Connectivity

Net Identifier Scope – this option defines how the sheet-to-sheet connectivity is created in a multi-sheet design. If the sheet symbols in your design use sheet entries to connect to the sub-sheets below, then you should choose Sheet Symbol/Port Connections. If your design uses nets or ports to connect directly between sub-sheets, you should choose one of the other options. For more information on sheet-to-sheet connectivity refer to the chapter *Multi-Sheet Design and Project Management*.

Assign Net to Connected Copper – if this option is enabled the net attributes of all the routing (tracks, vias, etc) is checked and updated to match the net attribute of the pads that the routing connects to. Enable this if you have modified the connectivity on the schematic. Once you have updated the PCB you can use the DRC error markers to find any routing that must be updated. This Assign Net to Connected Copper action can be performed at any time in the PCB Editor, select **Design » Netlist Manager** to do this.

Components

Delete Components – Automatically remove any components that are present in the target document, but have no matching component in the reference document.

Typically you would use this when you have made a large number of design changes in the schematic, and you want to automatically remove any redundant components in the PCB.

Update Footprints – If this option is enabled then changes to footprint information are transferred from the reference document, to the target document.

Rules

The Synchronizer can transfer PCB layout information from the PCB Layout directives in the schematic, to design rules in the PCB. Enable the Generate PCB Rules option to do this. There are two modes for this feature:

Only add missing PCB rules – new rules are created if there is no rule defined for this net, existing rules are updated to match the settings in the PCB Layout directive.

Strictly follow Schematic directives – new rules are created if there is no rule defined for this net, existing rules are updated to match the settings in the PCB Layout directive, any other net scope rules of this type are removed.

Refer to the topic, *Transferring PCB Layout Information* later in this chapter, for more details on how the PCB Layout directives map to design rules.

Classes

Generate Component Classes – enable this to automatically create a PCB component class from the set of components on each schematic sheet. Each component class is given the same name as the schematic sheet it is created from, with any spaces removed. Multipart components that span more than one sheet are included in the class of the sheet that contains the first part of the component.

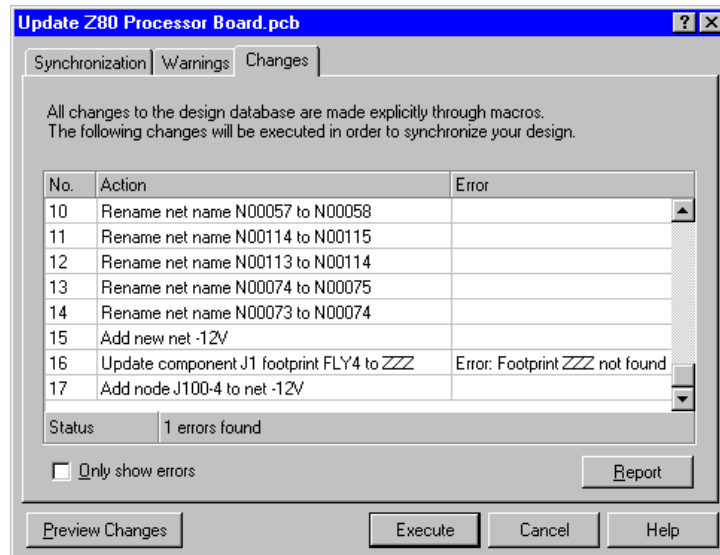
A PCB placement room is also created for each component class. These placement rooms are spread across the board, ready for positioning. For information on working with placement rooms refer to the topic, *Working with Placement Rooms*, in the *PCB Design* section of the Handbook.

Creating Net Classes from Buses – A PCB net class can be created for each schematic bus.

Preview Changes

At the bottom of the Update dialog there is a button labeled Preview Changes. If you would like to examine what changes the Synchronizer is about to perform, before it carries them out, press this button.

The Changes Tab will appear in the Update dialog, listing each change that will be made to the target document. Note that you can show just the errors by enabling the option at the bottom of the dialog. If necessary, you can delete a macro by right-clicking on the macro in the list, and selecting Delete from the floating menu. Press the Report button for complete details of each macro in the change list.



Preview the changes before they are applied, and check for errors

How the Synchronizer knows which PCB to Use

Wondering how the Synchronizer knows which PCB goes with which schematic? The following strategy is used. It looks for a PCB in the same Folder in the Design Database as the schematic project. If it finds a PCB it uses this. If there is more than one PCB in this Folder you are prompted to select the correct one. If there are none in the same Folder, the entire Design Database is searched, and you are prompted to select the correct PCB from a list of all the PCBs in the Design Database. If there are none in the Design Database a new PCB is created, in the same folder as the schematic project.

Where the Components are Placed in the PCB Workspace

The Synchronizer will place the components in rows of matching footprints. If you defined a placement outline on the PCB keepout layer prior to transferring the design information, the components are placed to the right of this outline. If there is no placement outline defined in the workspace the components are placed at the absolute center of the workspace, coordinate 50000, 50000.

Transferring PCB Layout Information

The Synchronizer will also translate PCB Layout directives placed on the schematic, into PCB Design Rules. The settings in each Layout directive are translated into appropriate design rules, with a Rule Scope of Net.

PCB Layout attribute	Design Rule
Track Width	Width Constraint
Via Width	Routing Via Style
Topology	Routing Topology
Priority	Routing Priority
Layer	Routing Layers

Passing Forward Schematic Design Updates

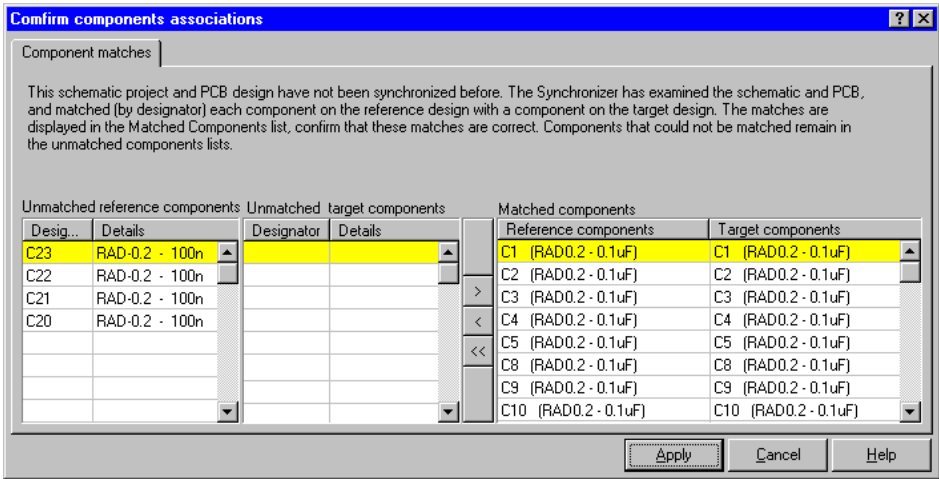
A normal part of the design process is to carry out changes to the schematic design, and then pass these changes forward to the existing PCB layout.

This is done in exactly the same way that the design information was initially transferred from the schematic to the PCB, by selecting **Design » Update PCB** from the menus. When you do, the synchronizer analyzes both the schematic and PCB, and identifies any differences. It will then create a set of macros, one for each difference, and execute these macros to re-synchronize the schematic and PCB designs.

How the Synchronizer Associates the Schematic and PCB Components

When a schematic and its PCB have been synchronized, matching schematic and PCB components are assigned a matching identifier. This approach means that you are free to separately re-annotate the schematic or the PCB. They can be brought back into harmony at any time by running an Update from the design menu.

If a component without a matching identifier is found in either the schematic or the PCB, the Synchronizer attempts to find a matching component for it. These matches are shown in the Confirm Components Association dialog, which automatically appears whenever unmatched components are detected by the Synchronizer.



Matched and unmatched components are listed in the Confirm Components Associations dialog

The Synchronizer does this initial match by designator, always check that the matches are correct. When you click on the Apply button the matched components will be given a matching ID.

If the Update is from schematic to PCB, the unmatched reference components are added to the PCB and the unmatched target components are either removed or reported, depending on the Components option in the Update dialog.

If the Update is from PCB to schematic, the unmatched reference components are listed in the Preview Changes Report, and the unmatched target components are either removed, or reported, depending on the Components option in the Update dialog.

How the Synchronizer Transfers the Design Information

To transfer the design information from the schematic to the PCB, the synchronizer extracts the component and connectivity information and creates a set of macros. Each action that it has to perform, such as adding a new component, adding a new net, or adding a node to a net, is defined by a macro.

If any macro can not be carried out (for example there is no footprint available), it will be flagged as an error in the Changes Tab of the Update dialog.

Resolving Macro Errors

Prior to executing the macros it is good practice to resolve any errors or warnings. Following is a description of each error/warning. The Error descriptions below indicates which macro can report that error, and what has caused the error.

Net not found

A macro is attempting to: add or remove a node; remove a net; or change a net name when that net can not be found in the PCB netlist.

Component not found

A macro is attempting to: add or remove a node when the component designator is incorrectly specified in the macro or the component can not be found in the PCB netlist; remove a component; or change a footprint, designator or comment when the component can not be found in the PCB netlist.

Node not found

A macro is attempting to: add or remove a node from a component which does not have that pin; or remove a node which does not exist in the specified net.

Net already exists

A macro is attempting to: add a net name when a net with that name already exists in the PCB netlist.

Component already exists

A macro is attempting to: add a component when a component with that designator already exists in the PCB netlist.

New footprint not matching old footprint

A macro is attempting to: change a component footprint when the *used* pins on the old footprint do not match the *used* pins on the new footprint. This can occur if the new component has fewer pins than the old, or if the pin numbering in the (which comes from the schematic component pin numbers) is different to the pin numbering on the PCB component.

Footprint not found in Library

A macro is attempting to: add a new component or change a component footprint when the specified footprint could not be found in any of the libraries in the current library list and no alternative library reference could be found in the Cross Reference file (ADVPCB.XRF).

Alternative footprint used instead (warning)

A macro is attempting to: add a new component or change a component footprint when the specified footprint could not be found in any of the libraries in the current library list. An alternative library reference was found in the Cross Reference file (ADVPCB.XRF) and this component will be loaded from one of the libraries in the current library list. Always confirm that the alternative footprint is appropriate before executing a macro with this warning.

◆ When a macro attempts to load or change a component footprint which can not be located in any of the libraries, it then uses the component Comment to look-up the Cross Reference file (Advpcb.XRF). The Cross Reference file lists components by their type, against any appropriate footprint(s) for that component. For example, if the component U1 was a 74LS00, but you had forgotten to include the footprint, when the macro to add this component was created it would look up 74LS00 in the XRF file. 74LS00 has DIP14 as a footprint, which would be loaded from one of the libraries in the current library list.

Summary

Most problems with passing a schematic design to the PCB Editor generally fall into two categories.

1. Component footprints – missing components occur when: a footprint is missing from the component information in the netlist; you have forgotten to add the required PCB libraries to the current library list; or the footprint is not available in any of the libraries in the library list.
2. New footprint not matching old footprint – the cause is usually that the pin numbering on the schematic component differs from the pin numbering on the PCB footprint.

Schematic libraries contain specific components and devices. The PCB libraries contain generic footprints, which can belong to various specific components – each having different pin assignments.

For example, a transistor shape can represent various combinations of “E,” “B” and “C,” – each of which must be assigned to the correct pin number in the PCB Editor. Diodes are a similar case, with pins often named “A” and “K” in the schematic.

You will need to either, modify the PCB footprint pin numbers to match the Schematic pin numbers, or change the schematic component pin numbers to match the PCB footprint.

Printing Your Schematic Design

Overview

Completing the schematic layout is only part of the design process, in most cases you will need to generate permanent drawings that can be filed and viewed “off-line.” The Schematic Editor includes support for a wide variety of hard copy options for this stage of the design process. Virtually any device that is supported by Windows can be used to print or plot your drawings.

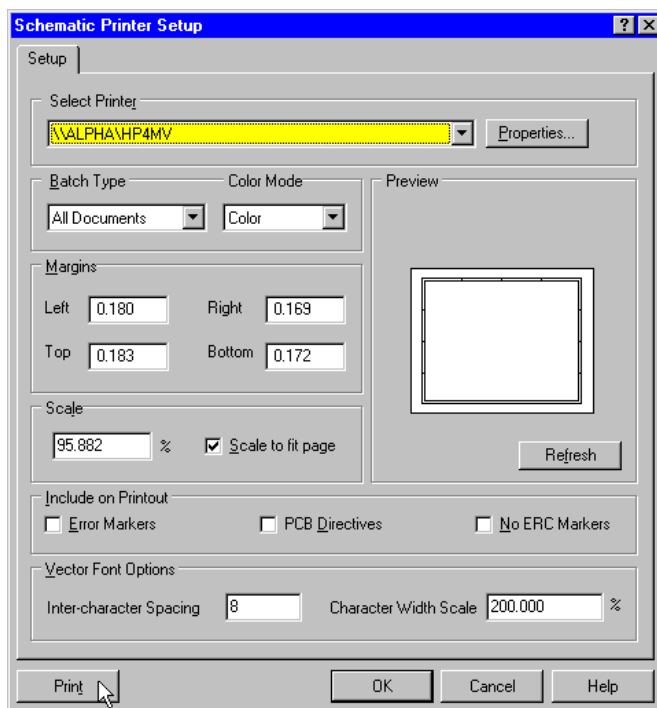
◆ Press the Properties in the Printer Setup dialog to set the portrait/landscape mode.

Generating a print or plot

Advanced schematic printing and pen-plotting are handled similarly to other Windows applications. Windows manages the printing (or plotting) process, and provides a range of raster and PostScript printer drivers and vector plotter drivers. These range from 9 pin dot matrix printers and multi-pen plotters, to high-resolution raster imagesetters.

To setup to print or plot from the active Schematic Editor or Library Editor window, select the **File » Setup Printer** menu item (shortcut: F, R).

The following topics describe each option in this dialog.



Selecting a Printer

The available output device options will include those that have been installed using the Windows Control Panel (see your Microsoft Windows User's Guide for details). These devices are supported by drivers delivered with your Windows software, or by the device manufacturer. You should note that new and updated drivers are released for both new and existing devices on a regular basis. For the latest information about print drivers, contact Microsoft Windows support or the device manufacturer.

◆ Rotation of fonts is not supported for all printers, and the substituted fonts will only be used if the text on your schematic is in a standard horizontal (or landscape) orientation, and within the size capability of the printer. PostScript printers support rotation of fonts at any angle.

Batch Type

When using this option from the Schematic Editor, this option prints either a single sheet or a batch of all open sheets (including all opened projects). Using this option from the Library Editor to choose between printing a single component (from the window that is the current focus) or all components in the current library. The latter option allows you to print out an entire component library in a single operation. When you choose this option, all representations of a component are printed, including each part's DeMorgan and IEEE equivalents, when applicable. Component description fields are also added to the sheet. This option works with all other print/plot options, including scaling, and so on.

Color mode

Two choices are available, Color or Monochrome. If you select color the schematic screen color assignments are used to assign colors to the print or plot, based upon the options available in the print or plot driver. Monochrome PostScript or HP-PCL devices will print grayscale representations of color. The number of gray levels, and the assignment of color to grayscale depends upon the driver and device. The Monochrome option prints images in solid black/white only. No dithering or grayscale support is provided. This option is appropriate for low resolution dot matrix and single pen plotting.

Margins

You have total control over margins, limited only by the margin limits built into printers or plotters that do not allow printing to the sheet edge (e.g., PostScript printers). When used with the Scale and Scale to Fit Page options (described below), this option will size the print area to fit as closely within the margins as allowed by the aspect ratio of the print area. To do this set all the Margin fields to zero and press the Refresh button. The Margins will automatically be set at the minimum for the currently selected printer.

Scaling prints and plot

Prints and plots can be scaled to a known factor or automatically scaled to fit within pre-defined page margins.

Scale

Type a scale factor from .001 % to 400%.

Scale to Fit Page

The print or plot will be expanded or contracted to fit within your pre-defined margins, on the page size selected up for the target printer. The plot will be shrunk or expanded to use the available space, keeping the correct aspect ratio.

◆ When you use scale to fit you can set the margins to zero, the printer driver will automatically cater for the no-print zones.

Tiling

When the size of the sheet or library document to be printed exceeds the print area available on the target device, the Schematic Editor will automatically cut the print into two or more sheets, or *tiles*. The sheets are tiled such that the correct margin is maintained at the outer edge of each sheet. You can preview the result of tiling by pressing the Refresh button.

◆ It is often possible to reduce the number of sheets required to tile a print, by changing the printer page orientation and adjusting margins. Use the Preview view in the Printer Setup dialog to help work out the best match.

Printer Properties

Pressing the Properties button opens the Print Setup dialog, where options for the target device are available. Depending upon the device, options include: sheet size/orientation, the printer tray to use, and so on. From this dialog you will have access to the printers own setup dialog.

Refresh Button

After changing any of the setup parameters you must press the Refresh button to refresh the preview display.

Printing

Once all options have been entered, Click Print to proceed with the print or plot, OK to save all the set-ups or Cancel to leave the Printer Setup dialog without saving the new parameters. As the print or plot is generated (either directly to the output device, or to a

filename) the current page and layer being printed is displayed in a dialog. If printing or plotting to a file, you will be prompted to supply an Output File Name.

PostScript Printing Issues

Some PostScript printers will “time out” and discard the current data when they don’t receive the end of page marker within a specified time. This can cause problems where you seem to be missing pages from your plots. If you experience this problem using a PostScript printer or any other printing device then you should go to the Control Panel, select the printer icon, select the printer and click the Configure button. Change the Transmission Retry to 500 seconds, or some other large number. This will allow the printer sufficient time to catch up before the Print Manager gives up.

If you find your printout is incomplete, say all the components are there but not all the wires, there may be insufficient memory in the printer. Laser printers must capture the entire image in memory before printing it, so if does not all fit in memory, then the image in memory is printed as is.

Creating Schematic Reports

Bill of Materials

When you select **Reports » Bill of Materials** the Schematic Bill of Materials Wizard will start. Follow the instructions on each Tab of the Wizard. Three formats of the BOM report can be produced;

Protel Format

A tabulated ASCII text format that lists part type fields, quantities of each type, and the designator fields associated with each type, named <FILENAME>.BOM.

CSV Format

A CSV (Comma Separated Value) format report, includes complete sheet-level and library level descriptions for each component, by designator. The report is named <FILENAME>.CSV.

Client Spreadsheet

Includes complete sheet-level and library level descriptions for each component, by designator. This version is automatically loaded in the Client Spreadsheet editor, from there it can be saved as an Excel format file.

Cross Reference

This ASCII text report gives a listing of part designators, type and the sheet location (filename) for each part. The report is named <FILENAME>.XRF.

Project hierarchy

This ASCII text report gives a listing of project files for the active design. The report is named <FILENAME>.REP.

Netlist Compare

This ASCII text report lists the differences between two netlists. Use this report to document changes made to a project from one revision to another. This feature works with Protel, Protel 2 and Tango format netlists. Among other details, the report lists matched nets, partially matched nets, extra nets in the first or second netlist, total nets in each netlist. The report is named <FILENAME>.REP.

Linking to Databases

The Schematic Editor includes two powerful and flexible methods of linking to external databases.

- ‘Hot linking’ of component part fields to external databases. This feature allows mapping of external database information directly to the component part fields on the schematic sheet.
- Database import and export features. These allow you to export and import the value of any attribute, of any object, placed in the Schematic Sheet Editor.

Hot Linking to a Database

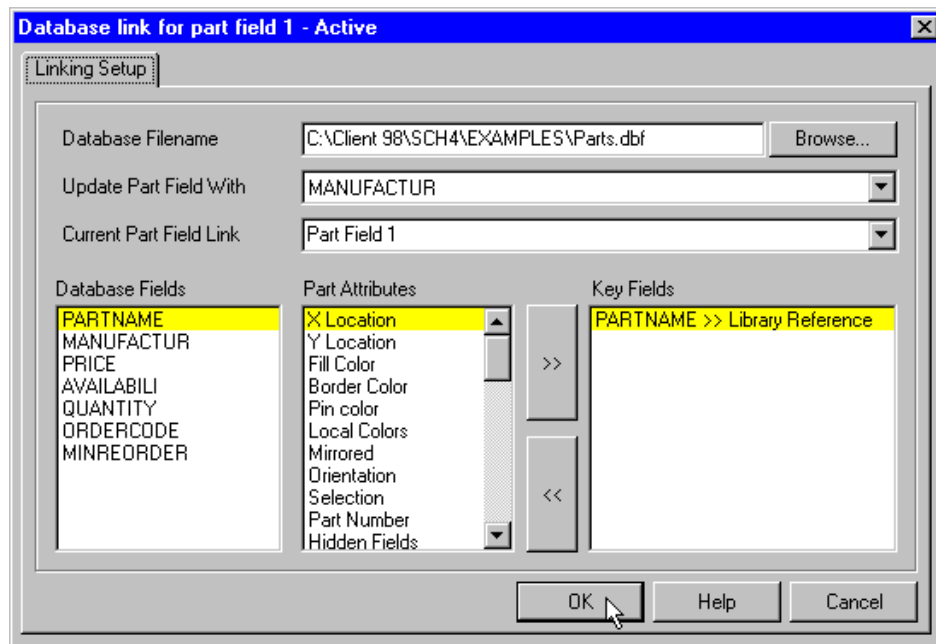
Hot linking to a database allows real time merging of external database information with the part fields of the components used in your schematic.

The links that are created to an external database do not belong to a individual component or a particular sheet, they are applied to every component on every sheet that is opened in the Schematic Editor. These links are stored in the schematic INI file, and are available each time the Schematic Editor is run.

To define the links, select the **Options » Database Links** menu item to pop up the Linking Setup dialog. Each of the sixteen component part fields can be linked to a database. They can all be linked to the same database, or each to a different database. Click the checkbox next to a Part Field name to activate that link. When a link is active the Schematic Editor will attempt to read from the database and update this part field as often as specified in the Update Every “XX” Minutes option.



Establishing the Links



To establish the links between a part field and a database, click to select the Part Field in the Linking Setup dialog and press the Configure Button. This will pop up the Database Linking for Part Field “X” dialog.

◆ Note that the title bar of the Database Link dialog will also say Active or Inactive, indicating if this link is currently enabled in the Linking Setup dialog.

Specify the Database

The first step is to select the database you wish to link to. Enter the Database Filename (including the full path), or use the browse button to locate the database. Supported database formats include dBase III and dBase IV.

Select the Database Field

Once you have selected the database the Schematic Editor will read in all the available fields. The next step is to choose which field in the database you wish to map into the Part Field you are configuring. To do this select the database field in the **Update Part Field With** drop down list. In our example we are mapping MANUFACTURER to Part Field 1.

When the contents of the key fields match (PARTNAME=Lib Ref), then the information is transferred from the database to the mapped Part Field (in this example all the linked Part Fields use the same key field).

The 'Part' dialog box is shown with the 'Attributes' tab selected. It contains a table of parts and a form for editing a specific part. The table has columns: PARTNAME, MANUFACTURER, PRICE, AVAILABILITY, QUANTITY, ORDERCODE, and MINREORDER. The form has fields for: Manufacture, Price, Availability, Quantity, Order Code, Min Reorder, Part Field 7, and Part Field 8. Arrows indicate the mapping of data from the table to the form fields.

PARTNAME	MANUFACTURER	PRICE	AVAILABILITY	QUANTITY	ORDERCODE	MINREORDER
6264	Texas Instruments	\$1.00	3 Days	30	T-47832	15
2764	Intel	\$2.00	1 Day	30	I-875543	10
Z80ASIO	Intel	\$4.00	2 Weeks	5	I-890765	2

Note - the Part Field names have been updated in the Library Editor for all the parts in the project, making it easy to identify the contents of each Part Field.

Map the Key Fields

The purpose of mapping the key fields is to establish how to identify the desired record in the external database. For example, consider a database that contains a list of components, with each component record including details like manufacturer, price, availability, etc. In our example we are using the PARTNAME in the database as the record identifier. This field name is then mapped to the schematic component's library reference, so that *when their contents match, the link is established*. Once the link is established between a record in the database and a component part field, the contents of any field in that record can be extracted from the record and loaded into the component part field.

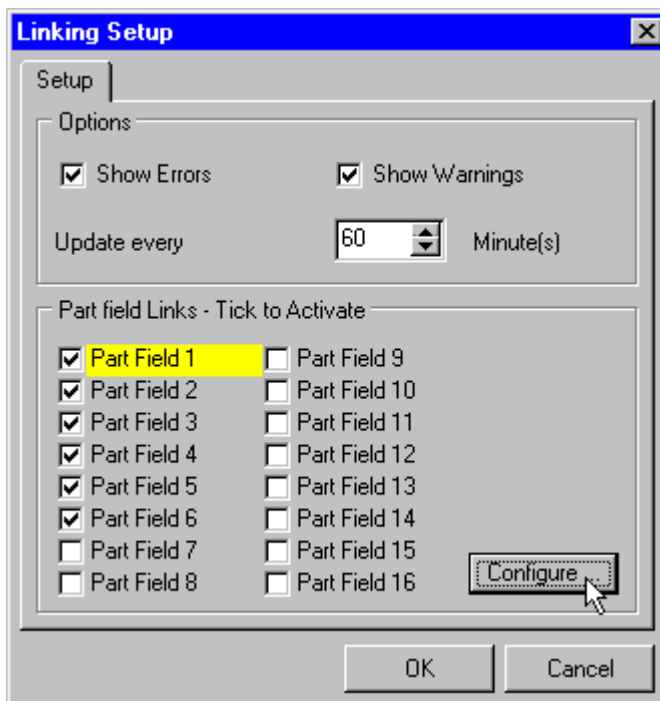
- ◆ To successfully link to an external database, the *contents* of the Key Fields must match. For example, if you map a database field with the name PARTNAME to the schematic component attribute LibReference, when the *contents* of PARTNAME is the same as the *contents* of the LibReference attribute, the link is established. The contents of one of the fields of this record will then be transferred to the part field.

To Map the Key Fields for the Part Field select the field in the Database Fields list box (PARTNAME), then select the attribute of the part that it maps to (Library Reference), then click on the >> button to add it to the Key Fields list.

- ◆ Each Part Field can link to a different database, and Each Part Field can also use a different Key Field.

Updating the Data

The links between databases and component part fields are dynamic. That is, the information is refreshed to keep it up to date. This is done automatically at the time interval specified. To disable the automatic updating (make the link Inactive), disable the Active check box in the Linking Setup dialog.



Importing and Exporting to a Database

Each object (or primitive) that can be placed in the Schematic Editor has a set of attributes. For example, a wire has three attributes; color, width and selection. A component has 33 attributes. The import and export facilities allow you to select which attributes are to have their values transferred to or from the database, it may be for two of the attributes, or it may be all. You can also specify the *scope* of the export or import. The scope can be the current sheet, the current project or all open sheets.

Once the values of the selected attributes have been exported into a database, they can then be processed by any DBMS (DataBase Management System) or spreadsheet application that can read the chosen database format. The supported formats include dBase III and dBase IV. The querying, manipulation and editing facilities of the DBMS can then be used to edit the database.

To be able to successfully export to a database, manipulate the data and then import from that database, there needs to be some way of uniquely identifying each instance of a primitive within the scope of the import.

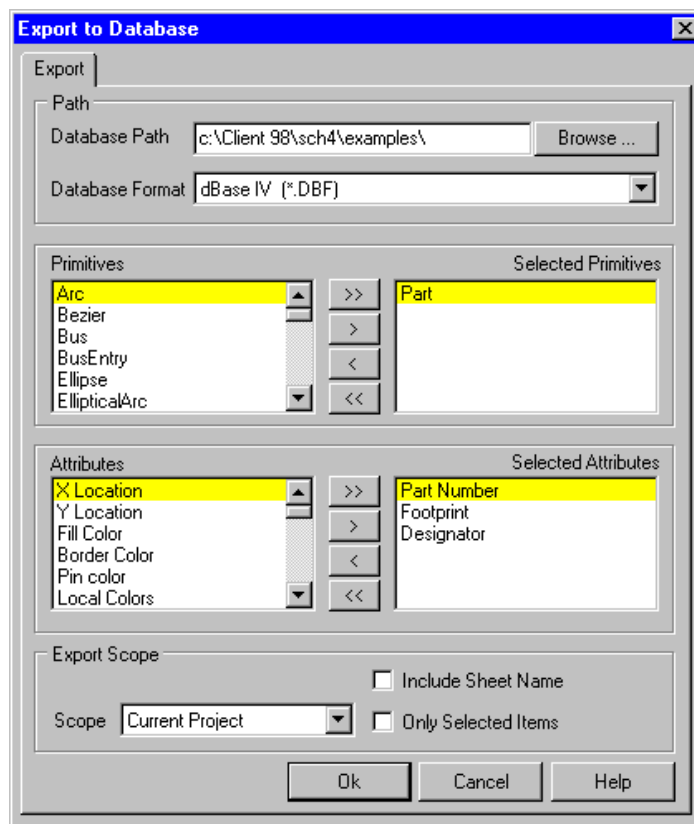
Special Key Attributes

To identify each primitive on the sheet and on which sheet that primitive belongs, primitives include special *location* attributes. There are two special location attributes. Every primitive has the *Document File Name* attribute to identify which sheet it came from. Primitives which do not have their own X - Y location information, such as wires or lines, also have a *Vertex Array* attribute which creates an array of the vertices that make up each wire or line.

When importing from a database these location attributes are then used as *Key Attributes*, attributes which the software will use to identify each instance of a primitive as it loads the information from the database.

Exporting to a Database

Selecting **File » Export to Database** to pop up the Export to Database dialog.



A default name will be provided for each database you export, based on the primitive selected.

Selecting the Primitives

The Primitives list box provides access to each of the primitives (or objects) available in the Schematic Editor. The desired primitives are selected and added to the Selected Primitives list box.

Once a primitive is added you then select which attributes of that primitive you would like to export in the Attributes list box. Continue with this process for each of the primitives you require; select the primitive, then select the attributes for that primitive.

◆ There is a separate database created for each primitive selected.

Each type of attribute will become a *field of cells* in the database, which must be identified. A field name is automatically assigned for each type of attribute when you export to a database.

Selecting the Attributes

For each primitive you select, you choose which attributes of that primitive are to be exported.

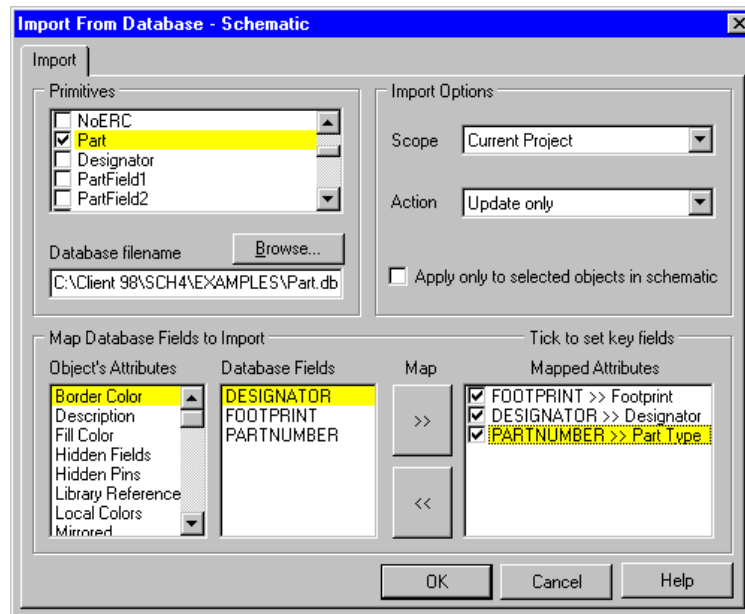
♦ When exporting, you *must* include the appropriate location attributes if you intend to later import the database. Without these location identifiers, the import feature will not be able to match the database fields with each instance of the primitive on their respective sheets.

Setting the Scope

Once the primitives and their attributes have been selected, set the export scope. The Schematic Sheet Editor has three options for the scope, the Current Sheet, the Current Project or All Open Sheets.

Importing from a Database

Selecting **File » Import from Database** will pop up the Import from Database dialog. Select the primitive type that you wish to import, then Browse to locate and select the database file. After setting the Scope and the Update Action, you are ready to Map the Database fields to the Object's Attributes.



Importing from a database is a process of transferring information in the database to the primitives in your design. To achieve this you must do two things; establish what you wish to extract from the database, and identify which instance the information is to go to.

To establish what you wish to extract you map the Schematic Object's Attributes to the Database Fields. To identify where it is to go you set the Scope and check the Key Attributes.

Mapping the Attributes

In the Database Fields list box select the field you wish to import from. Then select which of the Schematic Object's Attributes this field is to map to, and press the >> Map button. If the contents of that field can be mapped to the chosen attribute, they will appear in the Mapped Attributes list box. If the field chosen in the database cannot be mapped to that attribute of the object, a dialog will pop up warning that they are not the same *Type*. To be the same type, they must both be integers, or real numbers, or ASCII values, etc.

Setting the Key Fields

To identify how to locate the target object on a sheet, key field(s) must be selected. Each attribute in the Mapped Attributes list box has a check box next to it. The check box specifies if an attribute is to be used as a key field.

In the Mapped Attributes list box, select an attribute which uniquely identifies an instance of that object and check its check box. Examples of attributes that are appropriate for key fields are X location, Y location, Document File Name and Vertex Array. You must select the appropriate key attributes to identify each instance of the object for the scope of the import. If the import scope is current sheet, X location and Y location (or vertex array) will be sufficient. If the scope is over more than one sheet, include the Document File Name attribute as well.

Import Options - Scope

As with exporting, the scope of the import includes the Current Sheet, the Current Project and All Open Sheets. The Only Selected Items check box allows you to further narrow the scope.

Import Options - Action

There are three ways the data being imported from the database can be used. The data can be:

- imported as new objects - use this option to create a new schematic from the information in the database.
- used to update existing objects - use this option if you are updating an existing design.
- update existing objects if they exist and add new objects if they do not - use this option if you wish to update an existing design and add new objects to the sheet.

Interfacing to Third-Party Tools

This chapter includes information on how to pass design information from the Schematic Editor in Protel 99 SE, to other design environments.

Protel 99 SE's Schematic Editor can work with a number of third-party design tools, including the HP®-EEsof high frequency simulation tools, Xilinx® FPGA design tools, Orcad DOS schematic capture tools, and the Orcad Capture® tools.

A common way of interfacing to third-party tools is via a netlist – this chapter also includes information on creating a netlist from the Schematic Editor.

- ◆ Design examples for EEsof and Xilinx are included with Protel 99 SE, these examples include documentation on how to interface to these design environments.
- ◆ Check the Protel web site for information on translating Orcad and Protel DOS schematics.

Translating an Orcad Capture® Design

Protel 99 SE can translate Orcad Capture 7.x and 9.x DSN design files directly into Protel schematic sheets and libraries. The translation process is completely automated, simply import the original DSN file into a Design Database, and it is automatically translated into a Schematic project, and a matching schematic library.

Importing and Translating the DSN file

1. Create a new Design Database.
2. Right-click in the Design Database folder view to display the floating menu, and select **Import**.
3. Locate the DSN file in the Import dialog, and click the Open button.
4. The DSN file is first imported into the Design Database, and then translated.

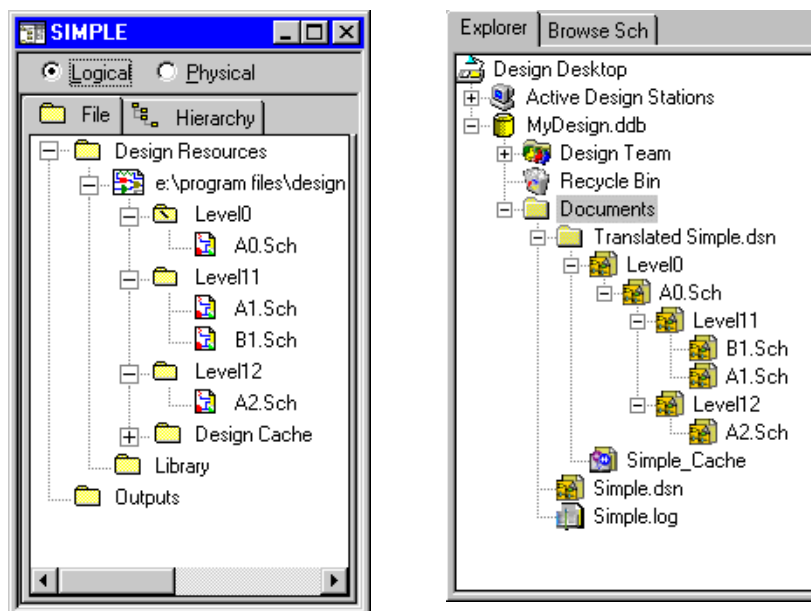
◆ It is a good idea to remove any schematic libraries from the current library list before importing and translating a DSN file.

Orcad does not store components with the actual schematic sheets, they are held in an internal design library, called the Design Cache.

So that you do not need to translate this cache into a Protel library first, and then make it available in the Protel 99 SE Schematic Editor when the sheets are translated, the Design Cache is automatically translated into a schematic library. Once this is done, the database that you are working in is added to the Schematic Editor library list, making this new library (the translated design cache) available during the sheet translation process.

How the Orcad Design Structure is Mapped

All designs are imported as either simple hierarchy or complex hierarchy. The following diagram illustrates how the design structure is mapped.



Sample design shown in Orcad, and translated in Protel 99 SE. Protel 99 SE creates the hierarchy that exists in the Orcad design. Note the original design file (Simple.Dsn) is included inside the Protel Design Database. You can re-translate the entire design at any time by double-clicking on the icon.

- Each Orcad schematic folder is translated to a Protel schematic sheet, with a sheet symbol for each Orcad schematic page within that schematic folder.
- Each Orcad schematic page is translated to a Protel schematic sheet.
- Hierarchical pins are translated to Sheet Entries.
- Hierarchical ports are translated to Ports.

The connectivity between Orcad schematic pages in an Orcad schematic folder is created with off-page connectors. This horizontal connectivity is translated to vertical connectivity in Protel 99 SE in the following way:

- Off-page connectors are translated to Ports, and shown in red.
- A matching Sheet Entry is created on the sheet symbol representing this sheet, which is also shown in red.
- Wires with net labels create the inter-sheet connectivity on the parent sheet.

Notes

- Use the Sheet Symbol/Port Connection Net Identifier Scope for ERC and PCB Design Synchronization.
- The location of user-defined symbols can not be exactly determined, check the location of all user-defined symbols after translating.
- Pictures are not imported, a red rectangle will mark their location.
- Bus pins are not supported in Protel 99 SE, these are translated into normal pins and a warning is issued.
- Orcad supports the connection of buses with different names, this is not supported.

Mechanical CAD Interface

Protel 99 SE includes full support for importing from and exporting to both DWG and DXF format files. Select **File » Import** or **File » Export** from the Schematic Editor menus to start the process.

Summarizing the Mechanical CAD Interface features:

- Full support for DWG/DXF import/export to the Schematic editor, all versions from 2.5 to 2000.
- Component to block and block to component translation.
- Import scaling option.
- Option to include the schematic template during export.

Use the Help button and What's This help in the dialogs for more details on the import/export options.

Creating a Netlist

Netlists are common to most electronics design environments. Simply stated, a netlist is a summary of all the components and connections (or networks) that comprise a circuit.

Generally, netlists are simple ASCII text files. The typical netlist format includes descriptions of components, such as the designator and package type, combined with the pin-to-pin connections that define each net. Loading a netlist into a printed circuit board layout package automates many of the tedious and error prone operations inherent in the design process.

Netlist formats

Netlists come in many different formats, but are usually generated as ASCII text files and contain the following types of information:

- 1) Descriptions of the components in the circuit.
- 2) A list of all pin-to-pin connections in the circuit.

- 3) Some netlist formats can include additional information in component text or net text fields. This information can be used to link netlists with simulators and board layout in ways that aren't covered by netlist hierarchy and connectivity. Examples include data for simulation and PCB layout.

Some netlist formats combine the component and connection data in a single description. Others, including the Protel formats, separate the two sets of data into separate sections.

As straight-forward text files, netlists are readily translated into other formats using a simple, user-written program. Netlists can also be created (or modified) manually using a simple text editor or word processor.

If you intend to manually edit a netlist, make sure that you save the results in an “unformatted” or “text only” form, as hidden “control characters” can render the netlist unreadable by the target software.

Generating the Netlist

You can generate a netlist for a project at any time while using the Schematic Sheet Editor. Select the **Design » Create Netlist** (shortcut: T, N) menu item, the Netlist Creation dialog will open. Options include:

Netlist Output Format Options

The following table lists the available netlist output format options.

Algorex	EESof Touchstone	Protel – Hierarchical
AppliconBRAVO	FutruureNet	Protel Wirelist
AppliconLEAP	Hilo	Racal Redac
Cadnetix	Integraph	Scicards
Calay	Mentor BoardStation 6	Spice
Calay90	Multiwire	Spice Hierarchical
Case	Orcad – PLDnet	Star Semiconductor
CBDS	Orcad - PCB II	Tango
ComputerVision	PADS Ascii	Telesis
EDIF 2.0	PCAD	Vectron
EDIF 2.0 Hierarchical	PCAD NLT	VHDL
EEDesigner	Protel	Xilinx XNF
EESof Libra	Protel 2	

Net Identifier Scope

The Net Identifier Scope defines how the inter-sheet connectivity is created. There are essentially two ways the inter-sheet connectivity can be created, either vertically (from a sheet entry down to a matching port), or horizontally (directly from a port or net label to a matching port or net label). It is important that the Net Identifier Scope is set to suit the structure of the design.

Refer to the chapter, *Multi-Sheet Design and Project Management*, for more information on how to structure a multi-sheet design. The three Net Identifier Scope options are:

Net Labels and Ports Global

With this option, net labels are assumed to apply to all sheets in a project. In other words, nets are global and each instance of a net label and port is deemed to be connected to all other identically named net labels and ports (note – ports do not connect to net labels and net labels do not connect to ports). This model works like Protel Schematic 3 (DOS), where net labels are always global to all project sheets. This option creates the inter-sheet connectivity *horizontally*.

Only Ports Global

This option treats net labels as local only, only connecting within each sheet. Inter-sheet connections occur through identically labeled ports. This model works like the Orcad SDT “flat” project model. This option also creates the inter-sheet connectivity *horizontally*.

Sheet Symbol / Port Connections

This option makes inter-sheet net connections only through sheet symbol entries and sub-sheet ports. Ports are deemed to be connected only to identically named sheet entries in their sheet symbols on parent sheets. This model works like the Orcad SDT “hierarchical” project model. This option creates the inter-sheet connectivity *vertically*.

Sheets to Netlist

Active Sheet – netlist the active sheet only.

Active Project – open and netlist all sheets that are in the project hierarchy.

Active Sheet plus sub-sheets – open and netlist all sheets below this sheet, use this option if your design is part of the entire PCB schematic.

Options

The other options that define the netlist contents include:

Append Sheet Number to Local Net Names

Add the sheet number (**Design » Options** dialog, Organization Tab) to each net. If you have selected one of the net identifier scopes where net labels are local, this option adds the sheet number to the net, ensuring that each net in the netlist is unique. This option can also be used as an aid in debugging netlist problems, where nets should be connecting across sheets but are not. By appending the sheet number you can identify which sheet a net is isolated on.

Descend Into Sheet Parts

Enabled this option when using Sheet Parts. Sheet Parts are parts which are specified to behave like a sheet symbol, where the pins connect to identically named ports on a child sheet. The Sheet Path field in the Edit Part dialog is used to identify the child sheet. When this option is enabled, the netlist will include the

sheets hierarchically below the sheet parts. Refer to the topic, *Model 5 - Using Sheet Parts to Create Hierarchy* in the chapter, *Multi-Sheet Design and Project Management*, for more details on using this feature.

Protel Netlist Formats

If the netlist is going to be loaded into the PCB Editor, designators and package descriptions (footprint) are limited to 12 alphanumeric characters. Comments can be up to 32 characters long. Net names can be 20 characters. Pin numbers in netlists are limited to four alphanumeric characters. No blank spaces may be used within these strings.

Any number of components or nodes can be included in a Protel or Protel2 netlist, limited only by available memory.

Protel Netlist

The standard Protel netlist format is a simple ASCII text file, split into two sections. The first part of a Protel netlist describes each component:

[	Marks the start of each component description.
U8	The Component Designator (label).
DIP16	The Package Description (footprint). A footprint with this name must be available in the open PCB libraries.
74LS138	Part Type, (or comment).
(blank)	These 3 lines are not used.
(blank)	
(blank)	
]	Marks the end of the component description.

The second part of a Protel netlist describes each net:

(	Marks the start of each net.
CLK	Name of the net.
U8-3	First component (by designator) and pin number. Pin numbers in library footprints must be an exact match.
J21-1	Indicates the second node in the net.
U5-5	Third node.
)	Marks the end of the net.

Protel Netlist 2.0

This format is similar to the standard Protel netlist, with the addition of several fields that include schematic part fields (used for documentation and simulation) plus layout directives which provide net attributes. Advanced PCB version 2.0 or later load this format.

PROTEL NETLIST 2.0	The netlist header.
[	Begin component delimiter.
DESIGNATOR	(Each field is first named)

U1	Component designator.
FOOTPRINT	
DIP20	Library pattern (footprint).
PARTTYPE	
AmPAL16L8	Part Type field (when placed).
DESCRIPTION	
Description	Description field from schematic.
PART FIELD 1	(Field name can be defined in schematic)
Part Field 1	Part fields (1-16) from schematic.
(etc.)	
LIBRARYFIELD1	
Library Part Field 1	Library fields (1-8) from schematic lib.
]	End component delimiter.
(	Begin net delimiter.
VCC	Net name.
U1-20 AMPAL16L8-VCC POWER	
	First node in net.
	Includes: Component -pin designator.
	(single blank space)
	Part type-Pin name.
	(single blank space)
	Pin electrical type.
U2-14 4001-VCC POWER	
	Last component-pin node in net.
)	End net delimiter.
{	Begin Layout Directive delimiter.
TRACK	(Each field is first named).
10	Size of tracks (mils).
VIA	
50	Diameter of vias (mils).
NET TOPOLOGY	
SHORTEST	Net Topology for routing.
ROUTING PRIORITY	
MEDIUM	Routing priority.
LAYER	
UNDEFINED	Routing layer.
}	End Layout Directive delimiter.

Creating a VHDL Netlist

The VHDL netlist follows the IEEE VHDL Standard 1076/93. It defines standard logic vectors in accordance with IEEE std_logic 1164. The design can be a mixture of schematic sheets and VHDL code.

Schematic Component Libraries

For schematic symbols use the 2000 to 7000 series Xilinx libraries. These libraries are in the Xilinx.Ddb Library Database.

Creating a Mixed-Mode Design

The top sheet of a mixed-mode design must be a schematic sheet. A VHDL source file is referenced from a Sheet Symbol in the same way as a schematic sheet, by entering the filename in the Sheet Symbol Filename field.

The top sheet of the VHDL design can also be referenced from the “physical” component on the PCB schematic – enter the name of the VHDL top sheet in the Sheet Path Field of the “physical” component.

How the Design is Netlisted

An Entity declaration is created for the top sheet in the design, and the circuitry within the sheet is translated into the matching Architecture declaration. Each component and sheet symbol on the sheet becomes a VHDL component within that architecture declaration.

This process is then repeated for each sheet symbol/sub-sheet on the top sheet – an Entity declaration is created for the sheet symbol, and the components and sheet symbols on the sub-sheet become VHDL components within the matching Architecture declaration.

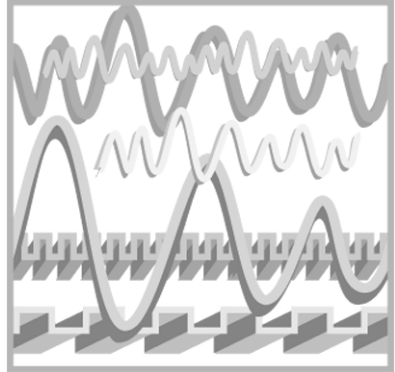
Netlisting the Design

To create the netlist select **Design » Create Netlist** from the Schematic Editor menus. In the Netlist Creation dialog set the Output Format to VHDL. The Sheets to Netlist list has three options:

Active Sheet – netlist the active sheet only.

Active Project – open and netlist all sheets that are in the project hierarchy.

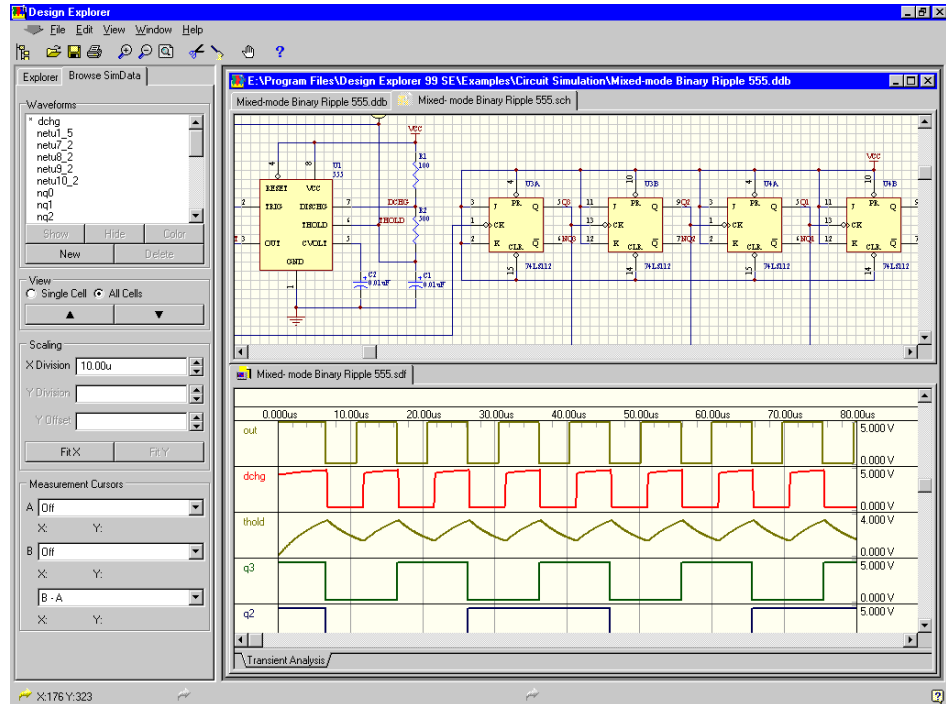
Active Sheet plus sub-sheets – open and netlist all sheets below this sheet (but none of the sheets above). Use this option if you design is part of the entire PCB schematic.



Section 4 ——— **Mixed-Signal Simulation**

<i>Mixed-Signal Simulation – Feature Highlights</i>	177
<i>Getting Started with Simulation</i>	180
<i>Setting up and Running a Simulation</i>	183
<i>The Waveform Analysis Window</i>	211
<i>Voltage and Current Sources</i>	219
<i>Components and Models</i>	235
<i>Working with Circuits that will not Simulate</i>	252
<i>SPICE Variables and Analog Options</i>	256
<i>Digital Simulation</i>	262

Mixed-Signal Simulation – Feature Highlights



This section of the Protel Designer's Handbook explains how to perform mixed-signal simulations of analog and digital designs. The Circuit Simulator is tightly integrated with the Schematic Editor, once the design is complete you can simulate directly from the schematic sheet.

As you read this section you will find all the information you need to get up and running with the simulator; learning how to use the features required to prepare your circuit, perform a variety of types of simulation analyses, and plot and manipulate the result waveforms.

Capture your Design

The first step in being able to simulate a design is to capture the circuit in the Schematic Editor. To be able to simulate the design The Circuit Simulator requires special information about each circuit element, such as the simulation model to use, what type of component it is, and so on. This information is stored in the simulation-ready schematic symbols libraries.

Mixed-Signal Simulation

There is a comprehensive set of simulation-ready schematic libraries in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb Library Database. Each of the symbols in these libraries includes a link to a simulation model. Once your design is complete you can setup and perform simulations directly from the schematic. You can re-configure and re-run the simulation at any time.

Advanced Simulation Technology

The Circuit Simulator performs accurate, “real-world” simulations of analog, digital and mixed-signal circuits. It will give you results like you would get from an actual breadboard. Devices function just like real-world parts, and each individual model functions like its real-world counterpart. For example, digital ICs have accurate propagation delays, setup and hold times. Outputs of the devices see the effect of loading on them, and nearly all the parameters of the real world are taken into account.

There are a wide variety of analyses that can be used to test and analyze various aspects of your design.

SPICE Compatibility

The Circuit Simulator uses an enhanced version of Berkeley SPICE3f5/Xspice, allowing you to accurately simulate any combination of analog and digital devices, without manually inserting D/A or A/D converters. This “mixed-signal” or “mixed-mode” simulation is possible because the simulator includes accurate, event-driven behavioral models for its digital devices, including TTL and CMOS digital devices.

SimCode for Digital Simulation

The Circuit Simulator is a true mixed-signal simulator, meaning that it can analyze circuits that include both analog and digital devices. However, due to the complexity of digital devices it is generally not practical to simulate them using standard, non-event-driven, SPICE instructions. For this reason, the simulator includes a special descriptive language that allows digital devices to be simulated using an extended version of the event-driven XSPICE. The digital devices included in the simulation library are modeled using the *Digital SimCode™* language, a proprietary language created specifically for use with the Protel Circuit Simulator.

Support for Device Manufacturers' Models

The simulator supports models from model providers such as Motorola, Texas Instruments and others, who deliver pure SPICE models for maximum compatibility with analog simulators. The simulator can read these models directly, providing true SPICE-compatibility.

Comprehensive Model Libraries

The Circuit Simulator includes a comprehensive set of libraries, in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb Library Database. Each of the symbols in these libraries is “simulation ready”, simply place one on your schematic sheet and it is automatically linked to an appropriate simulation model. These libraries are constantly maintained and developed by the Altium Library

Development Center, which specializes in developing libraries for all Protel products. These libraries are available from the Protel web site, www.protel.com.

Simulation Limits

The Circuit Simulator allows unlimited circuit-level analog simulation and unlimited gate-level digital simulation. The circuit can be single or multi-sheet, and the circuit size is only limited by the amount of RAM you have in your system.

Supported Analyses

The Circuit Simulator supports many types of analyses, including AC small signal, Transient, Noise and DC transfer. Beyond these basic analyses there is also Monte Carlo analysis, parameter and temperature sweeping, and Fourier analysis.

Mathematical Functions and Waveforms

As part of the analysis of your design you may want to perform a mathematical operation on one or more of the simulation signals, and view the resultant waveform. This feature is an integral part of the simulator's waveform viewer, you can construct a mathematical expression based on any signal available in the simulated circuit.

Dependent Sources

The Circuit Simulator includes linear and non-linear dependent sources. These can be used to define “black-box” circuit behavior.

Linear dependent sources

The E, F, G and H devices offer “predefined” equations to model simple linear dependencies. These devices offer a simple way to simulate simple linear effects.

Non-linear dependent sources

These devices allow you to define a voltage or current equation using a variety of functions (log, ln, exp, sin, etc), based on any voltage(s) in the circuit.

Getting Started with Simulation

Once you have created the circuit from the simulation-ready libraries, there are 3 simple steps to perform to be able to run a simulation:

1. Add the appropriate sources to power and excite the circuit
2. Define the points that you wish to observe
3. Setup the analyses

You are then ready to run the simulation. Before you start simulating your own circuits you might like to explore some of the example circuits. The example circuits are all stored in one Design Database, you can access all the examples in the Protel 99 SE Examples group in the Windows Start menu.

Example Circuits

The Circuit Simulator includes a large number of example circuits which demonstrate the various analysis types, as well as showing how the various types of devices can be used.

Refer to these examples as you explore the various Setup options for the simulator. As you experiment with the examples refer to the chapter, *Setting Up and Running a Simulation*, for information about how to configure each type of analysis.

◆ Check out the example designs in the folder
\\Program Files\\Design Explorer 99 SE\\Examples

Creating the Circuit

Protel 99 SE includes approximately 5800 simulation-ready analog and digital schematic components, each with a link to a simulation model. These components are all PCB compatible, with PCB pin-outs and footprints (where appropriate).

Placing the Components

The simulation-ready schematic libraries are in the \\program files \\Design Explorer 99 SE\\library\\sch\\sim folder. For tips on how to find components in these libraries refer to the *Component Libraries* topic in the *Components and Models* chapter.

Setting the Component Designators

There are no restrictions to component designator prefixes. For example, you could designate the ICs as U1, U2 – or IC1, IC2 – or X1, X2, and so on.

Component and Simulation Multipliers

The only letters that can follow a component or simulation value are scale factors (or multipliers), such as “n” for nano, “k” for kilo, and so on. Valid scale factors are shown in the table, all other letters are ignored. Any letters after the scale factor are also ignored. Note that the scale factor must be *immediately* after the number, spaces between the number and the scale factor are not permitted. The simulator is not case sensitive, letters can be upper or lower case. Following are 3 examples:

10, *10V*, *10Volts*, and *10Hz* all represent the same number, 10. The letters are ignored in all cases as none are a valid scale factor.

M, *m*, *MA*, *MSec*, and *MMhos* all represent the same scale factor, 10^{-3} . In each case the letters after the first “m” are ignored.

1000, *1000.0*, *1000Hz*, *1e3*, *1.0e3*, *1KHz*, and *1K* all represent the same number, 1000.

Scale factor	Represents
T	10^{12}
G	10^9
Meg	10^6
K	10^3
mil	25.4^{-6}
m	10^{-3}
u (or μ)	10^{-6}
n	10^{-9}
p	10^{-12}
f	10^{-15}

Adding the Source Components

Before you can run a simulation you will need to add the appropriate source components to power and excite the circuit. These can be placed from the **Simulate » Sources** sub menu. Pick the closest source to what you require from the menu, you can adjust the values once it has been placed. You can also place them directly from the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb Library Database.

The simulator includes DC, Sin, Pulse, Exponential, Piece Wise Linear and FM source components, as well as linear and non-linear dependent source components.

Once the source is placed double-click on it to set the values. Refer to the chapter *Voltage and Current Sources* for information on configuring source components.

Defining the Points to be Plotted

The points in the circuit that will have their waveforms displayed are specified in the Analyses Setup dialog (select **Simulate » Analyses Setup**). At the bottom of the General Tab there are two lists, Available Variables, and Active Variables. A waveform will be automatically displayed for each variable in the Active Variables list. It does not matter if you do not include a variable now though, you can add and remove variables in the waveform display after the waveforms appear.

Refer to the topic *Specifying the Simulation Data that you want Collected, Displayed and Stored* in the *Setting up and Running a Simulation* chapter for more information about defining what is displayed.

Setting up the Analyses

Select **Simulate » Analyses Setup** to pop up the Analyses Setup dialog, where all the simulation setup is performed.

The dialog will open with the General Tab displayed. Use this Tab to enable each analysis that you wish to perform. Each of the remaining Tabs in the dialog is used to configure that type of analysis. Refer to the chapter *Setting up and Running a Simulation* for details on setting up each type of analysis.

The Advanced button at the bottom of the Analyses Setup dialog gives you access to a number of SPICE variables. These are advanced settings, generally you will only need to change these if the circuit is difficult to simulate, or you have a specific reason to change them. Refer to the chapter *Setting up SPICE Variables* for more information on each variable. If you are having difficulties getting the circuit to simulate refer to the chapter, *Working with Circuits that will not Simulate*.

Running a Simulation

Once the analyses have been configured you are ready to run a simulation. This can be done by pressing the Run Analyses button in the Analyses Setup dialog, by selecting **Simulate » Run** from the menus, or by pressing the Run button on the Simulation Tools toolbar.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Viewing the Simulation Results

Once the simulation is complete the results are automatically displayed in a separate waveform analysis window. Click on the Tabs at the bottom of this window to display the results of each type of analysis. You can adjust your view of the waveforms with the Scaling controls on the panel. Select **View » Panel** from the menus to display the Panel if it is not visible.

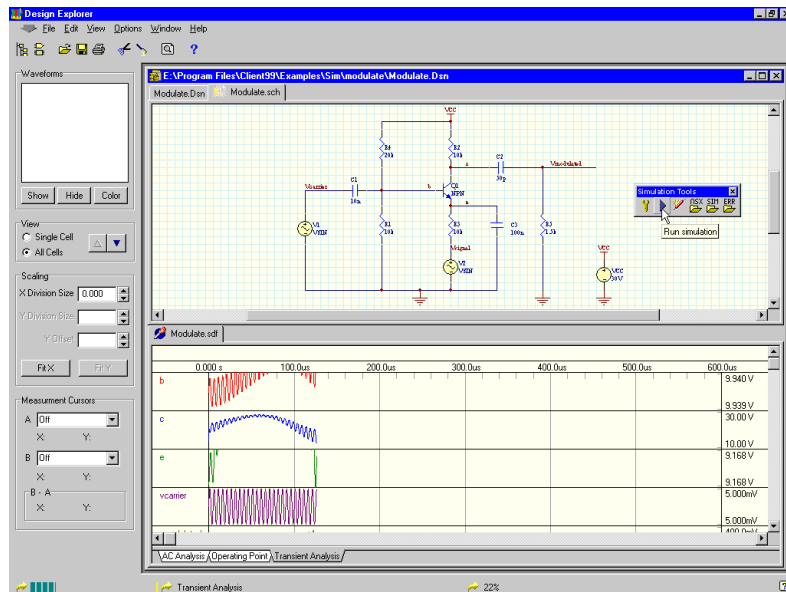
Refer to the chapter *Using the Analysis Window* for information on using the Scaling controls and the measurement cursors.

Setting up and Running a Simulation

Select **Simulate » Analyses Setup** from the menus to display the Analyses Setup dialog, where you configure all options and analyses for the simulator. The Analyses Setup dialog includes a General Tab, where the analyses are enabled and the data collection options are defined, and a separate Tab to configure each analysis type.

Once you have created the circuit you can run a simulation by clicking the Run Analyses button at the bottom of the Analyses Setup dialog, by clicking the Run button on the Simulation Tools Toolbar, or by selecting **Simulate » Run** from the menus. Refer to each analysis topic for details on how to configure the simulator to perform that particular type of analysis.

The Status bar displays the progress whenever a simulation is running. When you select **Simulate » Run** from the menus the first thing that happens is the circuit is netlisted. Once the netlist has been created it is passed to the simulation engine, which displays the simulation waveforms as the simulation progresses.



The Status bar shows the progress, and the waveform window displays the results.

Once the simulation is complete the results will be displayed in the Simulation analysis window. This window includes a Tab at the bottom for each type of analysis that has been performed. Refer to the chapter, *The Waveform Analysis Window*, for more information on working with the waveforms.

Specifying the Simulation Data that you want Collected, Displayed and Stored

The bottom of the General Tab of the Setup Analyses dialog includes two options that specify what type of analysis data is saved in the result file, and what waveforms will be automatically displayed in the waveform window.

Collect Data Option

The Collect Data For option specifies exactly what kind of information you want to be calculated and stored in the result file. Select the required option from the drop down list. The list has five options. Select one of the first four options to specify exactly what information you want the simulator to calculate (voltages, currents, power, impedances, etc) and store in the results file.

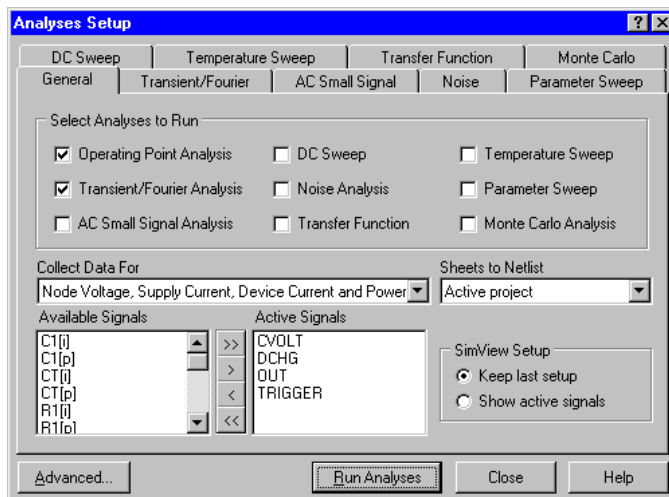
For each of the first four options the data is stored for *all* available signals in the circuit. The fifth option, Active Signals, instructs the simulator to only store data for the signals that you have added to the Active Signals list below. Use this option when you need to minimize the size of the result file.

The Active Signals option is also used to limit which nodes are examined during the multi-pass simulations (Monte Carlo, Parameter Sweep and Temperature Sweep). If the Active Signals option is not chosen data is collected for all nodes in the circuit for these analyses.

Available Signals and Active Signals

The Available Signals list shows all the available circuit signals for the Data Collection option that has been selected in the Collect Data drop down list. The simulator will store data for *all* of the signals shown in this list (unless the Active Signals option was chosen in the Collect Data drop down list).

Each signal that you include in the Active Signals will be automatically displayed in the result waveform window.



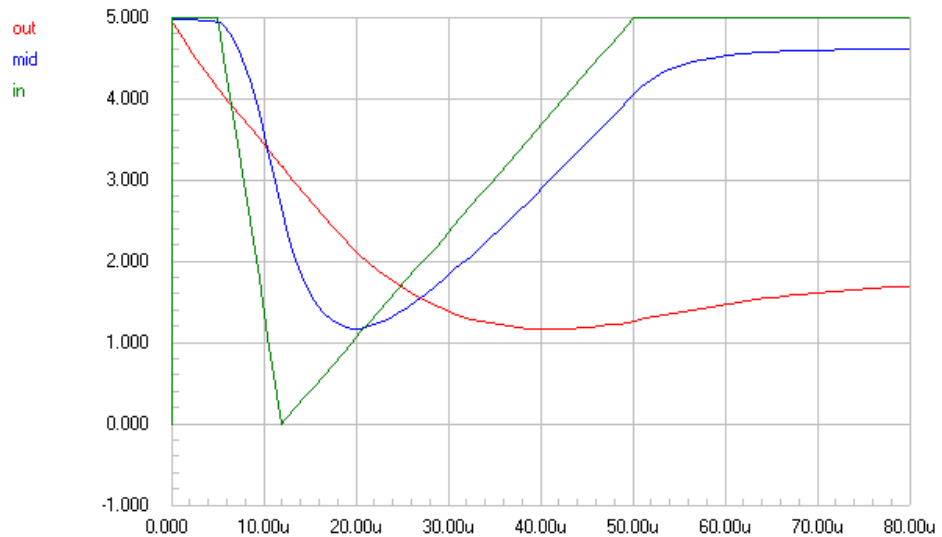
SimView Setup

When you run a simulation the results are displayed in the SimView Waveform Window. In the Waveform Window you can rearrange the display to suit your needs; you can change the scaling, add different nodes, remove existing nodes, and so on. When you close the Simulation Data File (.SDF), the setup information is saved with the file.

By default the display setup is retained, so that you can re-open a simulation data file and the results will be as you left them. If you re-run the simulation and you want it to display the Analyses and nodes that you have just setup, change the SimView Setup from Keep last setup, to Show active signals.

Transient Analysis

A Transient analysis generates output like that normally shown on an oscilloscope, computing the transient output variables (voltage or current) as a function of time, over the user-specified time interval. An Operating Point analysis is automatically performed prior to a Transient analysis to determine the DC bias of the circuit, unless the Use Initial Conditions option is enabled. Refer to the topic *When to Use the Initial Conditions Option* later in this topic for more information on this option.



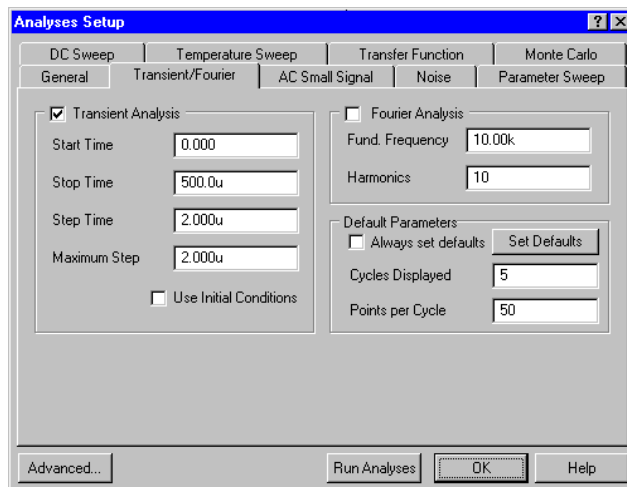
A Transient analysis calculates the voltages and currents as a function of time

Setting up for a Transient Analysis

Transient analysis is set up in the Transient/Fourier Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).

A Transient analysis always begins at time zero. In the time interval between zero and Start Time, the simulator analyzes the circuit but does not store the results. In the time interval between Start Time and Stop Time it continues to analyze the circuit, storing the results ready for display on completion of the analysis.

Step Time is the nominal time increment used in the analysis, however the actual timestep is varied automatically by the simulator in order to achieve convergence. The Maximum Step limits the varying size of the timestep that the simulator can use when calculating the transient data; by default it chooses either Step Time or $(\text{Stop Time} - \text{Start Time})/50$, whichever is smaller. Typically Step Time and Maximum Step are set to the same value.



Automatically calculating the Transient Analysis Parameters

If you are not sure what values to enter, press the Set Defaults button to automatically calculate the Transient analysis parameters. Start Time is set to zero and Stop Time, Step Time and Maximum Step are calculated using the Defaults; Cycles Displayed and Points per Cycle, based on the lowest frequency Source in the circuit. For example, if the lowest frequency source in the circuit was 10KHz and the Defaults were Cycles Displayed = 5 and Points per Cycle = 50 then:

$$\begin{aligned}\text{Stop Time} &= 1/10\text{KHz} * 5 \\ &= 100\mu\text{S} * 5 \\ &= 500\mu\text{S}\end{aligned}$$

$$\begin{aligned}\text{Step Time} &= (1/10000) / 50 \\ &= 2\mu\text{S}\end{aligned}$$

$$\begin{aligned}\text{Max Step} &= \text{Step Time} \\ &= 2\mu\text{S}\end{aligned}$$

If the Always set defaults for transient analysis option is enabled, the simulator acts as if this button is pressed before each simulation.

When to Use the Initial Conditions Option

If you enable the Use Initial Conditions option the Transient analysis begins from the defined initial conditions, *bypassing* the Operating Point analysis. Use this option when you wish to perform a transient analysis starting from other than the quiescent operating point.

To use this option you must either define the initial condition for each appropriate component in the circuit, or place .IC devices. Double-click on a component to edit the IC Part Field. Refer to the chapter *Components and Models* for information on each component kind. Alternatively, place .IC devices from the Simulation Symbols.lib library.

An initial value of zero is assumed for a component which does not have the Initial Condition defined.

◆ Generally it is better to use an .IC device to set the transient initial conditions. Refer to the topic, *Specifying Initial Circuit Conditions* at the end of this chapter for details on the .IC device.
Note: The IC value of a component overrides an .IC object attached to a net.

Running a Transient Analysis

To run a Transient analysis:

1. Set up the Transient analysis parameters as described above.
2. Enable the Transient/Fourier Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

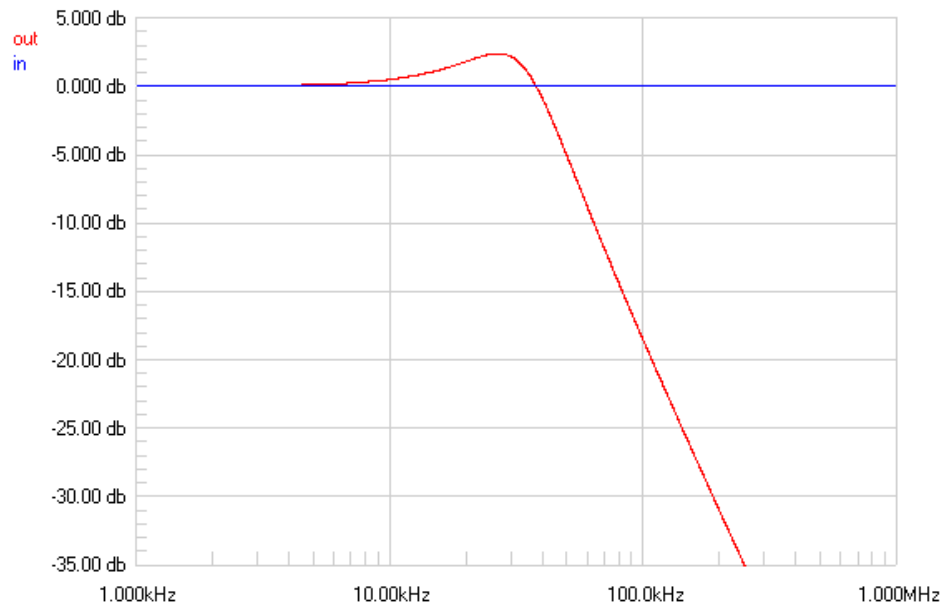
The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view and measure voltage, current and power dissipation waveforms of the circuit in the analysis window that is displayed. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

AC Small Signal Analysis (AC Sweep)

AC analysis generates output that shows the frequency response of the circuit, calculating the small signal AC output variables as a function of frequency. It first performs an Operating Point analysis to determine the DC bias of the circuit, replaces the signal source with a fixed amplitude sine wave generator, then analyzes the circuit over the specified frequency range. The desired output of an AC small signal analysis is usually a transfer function (voltage gain, transimpedance, etc.).

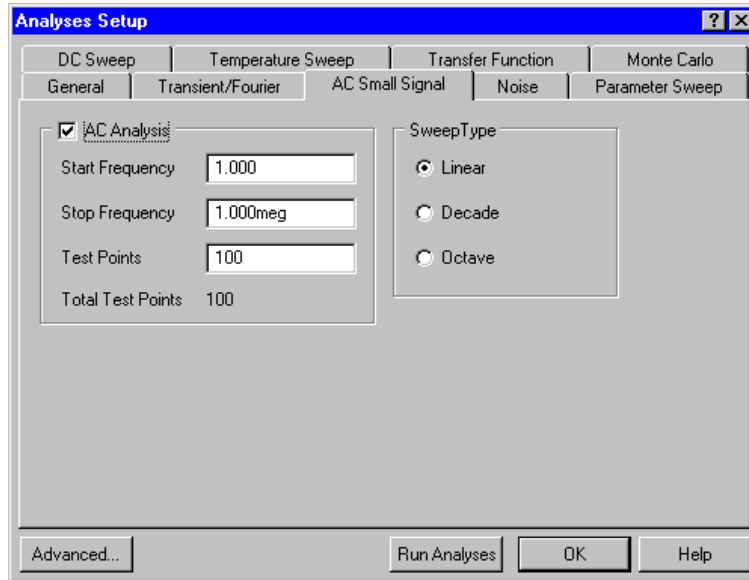


An AC small signal analysis shows the frequency response of the circuit

Setting up for an AC Analysis

AC Small Signal analysis is set up in the AC Small Signal Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).

◆ The circuit must contain at least one Source with a value in the AC Part Field.



At least one Source in the circuit must include a value in the AC Part Field. This Source is replaced with a sine wave generator during simulation which has its frequency swept from Start Frequency to Stop Frequency, stepping in increments defined by Test Points and the Sweep Type.

Set the Sweep Type to define how the test points are determined. The Sweep options are defined below:

Sweep Option	What it Means
Linear	Total number of Test Points in the sweep
Decade	Number of Test Points per decade in the sweep
Octave	Number of Test Points per octave in the sweep

The amplitude and phase of the swept sine wave are specified in the Part Fields of the Source. Double-click on the source to set these values. Enter the amplitude in the AC Part Field (in volts) and the phase in the AC Phase Part Field (in degrees). Units are not required.

◆ Set the AC amplitude to 1 so that the output variables are relative to 0 dB.

Running an AC Small Signal Analysis

To run an AC Small Signal analysis:

1. Set up the AC Small Signal analysis parameters as described above.
2. Ensure that there is at least one Source in the circuit with a value in the AC Part Field.
3. Enable the AC Small Signal Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
4. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

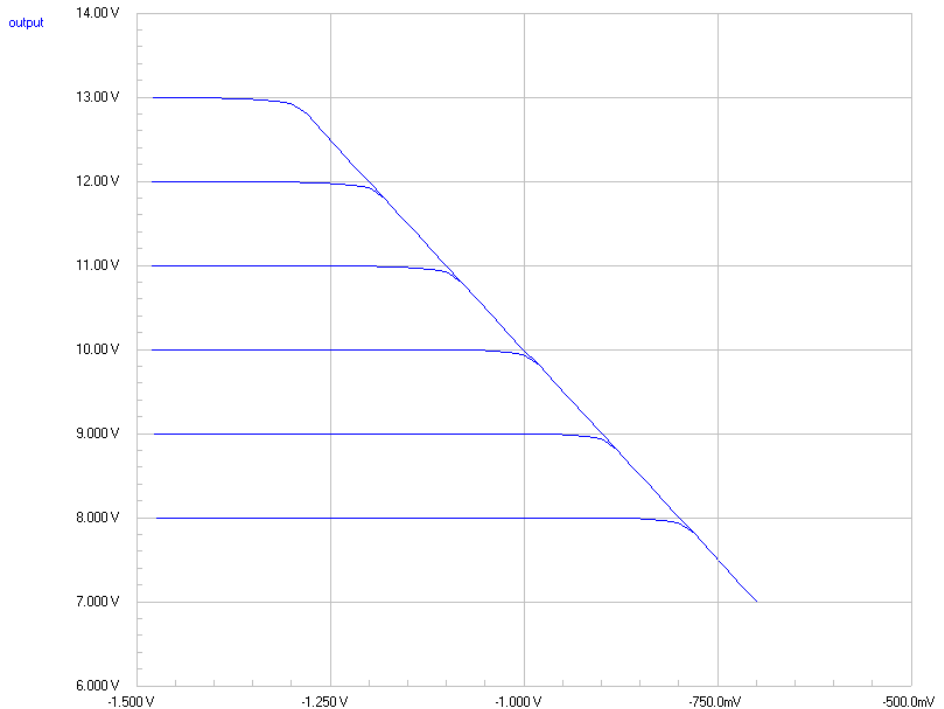
The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view and measure voltage, current and power dissipation waveforms of the circuit in the analysis window that is displayed. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

DC Sweep Analysis

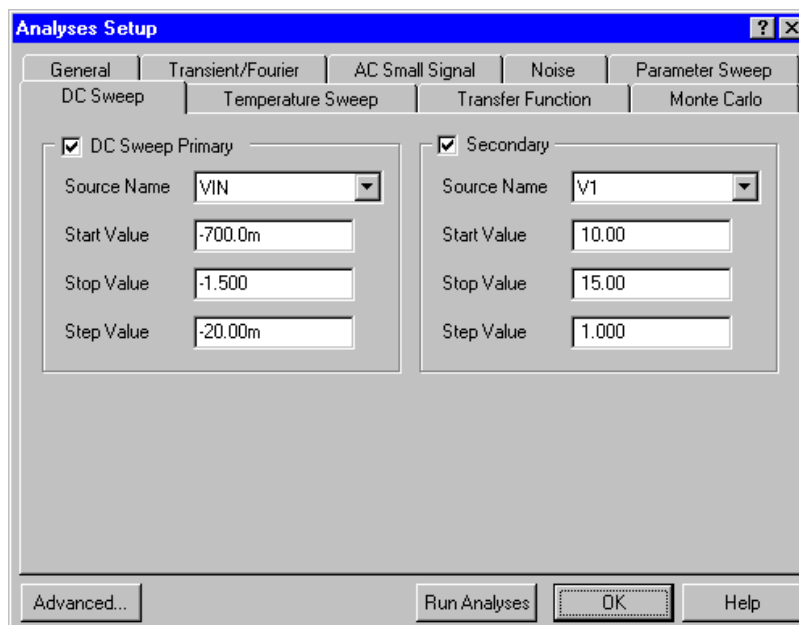
The DC Sweep analysis generates output like that of a curve tracer. It performs a series of Operating Point analyses, modifying the voltage of a selected source in pre-defined steps, to give a DC transfer curve. You can also specify an optional secondary source.



A DC Transfer analysis – each curve shows the output voltage, as the input voltage (primary source) is stepped from -0.7V to -1.5V . The supply voltage (secondary source) is stepped to give the set of curves.

Setting up for a DC Sweep Analysis

DC Sweep analysis is set up in the DC Sweep Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).



Source Name is the name of the independent power source in the circuit that is to be stepped. The start, stop and step values define the sweep range and resolution. The primary source is required, and the secondary source is optional. If a secondary source is specified, the primary source is stepped over its entire range for each value of the secondary source.

Running a DC Sweep Analysis

1. Set up the DC Sweep analysis parameters as described above.
2. Enable the DC Sweep Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view and measure the transfer curves from the waveforms in the analysis window that is displayed. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

DC Operating Point Analysis

Prior to performing a Transient or an AC Small Signal analysis, the simulator must first perform an Operating Point analysis. The Operating Point analysis determines the DC bias of the entire circuit, with inductors considered to be short circuit and capacitors open circuit. It also determines linearized, small-signal models for all nonlinear devices in the circuit, that are used in the AC Small Signal analysis. It does not take into account the existence of any AC sources.

Running an Operating Point Analysis

The Operating Point analysis is performed automatically whenever a Transient or AC Small Signal analysis is enabled. These results are used internally by the simulation engine. If you want to examine the results of the operating point calculations you can enable the Operating Point analysis. To do this:

1. Enable the Operating Point Analysis in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
2. To include the current, power and impedance calculations set the Collect Data For option in the General Tab of the Analyses Setup dialog to Node Voltages, Supply Currents, Device Currents and Powers.
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

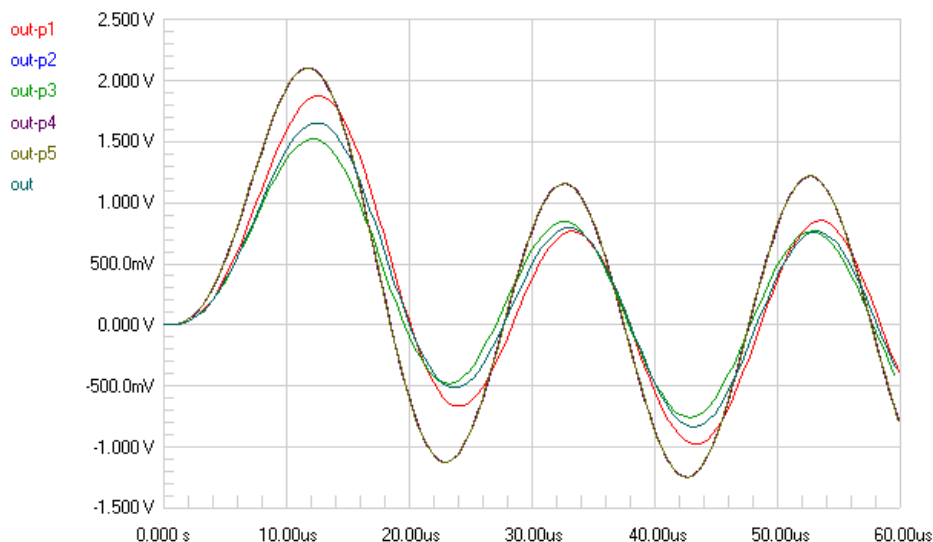
The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

You can then view the results in the Operating Point Tab of the analysis window that is displayed. You can add nodes and devices to the list of results by selecting them in the Waveforms list in the panel, and clicking the Show button.

Monte Carlo Analysis

A Monte Carlo analysis performs multiple simulation runs, as device tolerances are randomly varied across specified tolerances. You can use this feature only when you have enabled one or more of the standard analyses (AC, DC, or Transient). The simulator only saves Monte Carlo data for nodes that have been added to the Active Variable list in the Setup Analyses dialog.

Subcircuit data is not varied during the Monte Carlo analysis, only basic components and models can be varied.



Use the Monte Carlo analysis to examine the circuit performance as the component tolerances vary

Setting up for a Monte Carlo Analysis

Monte Carlo analysis is set up in the Monte Carlo Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).

The Monte Carlo options are set up as follows:

Simulation Runs

Enter the number of simulation runs you want the simulator to perform. For example, if you enter 10, then 10 simulation runs will be performed, with different device values on each run, within the specified tolerance range.

Simulation Seed

The simulator uses the seed to generate random numbers for the Monte Carlo runs. The default seed value is -1 . If you want to run a simulation with a different series of random numbers you must change the seed value to another number.

Default Distribution

You can choose from the following three distributions for random number generation in the Monte Carlo analysis:

Uniform distribution

Uniform distribution is a flat distribution. Values are uniformly distributed over the specified tolerance range. For example, for a 1K resistor with a tolerance of 10 percent there is an equal chance of the generated value being anywhere between 900 ohms and 1100 ohms.

Gaussian distribution

Values are distributed according to a Gaussian (bell-shaped) curve, with the center at the nominal value and the specified tolerance at ± 3 standard deviations. For a resistor with a value of 1K $\pm 10\%$ the center of the distribution would be at 1000 ohms, $+ 3$ standard deviations is 1100 ohms, and -3 standard deviations is 900 ohms. With this type of distribution there is a higher probability that the generated value will be closer to the specified value.

Worst Case distribution

This is the same as the uniform distribution, but only the end points (worst case) of the range are used. For a 1K $\pm 10\%$ resistor the value used would be randomly chosen from the two worst case values of 900 ohms and 1100 ohms. On any one simulation run there is an equal chance that the high-end worst case value (1100) or low-end worst case value (900) will be used.

Specifying Default Tolerances

You can specify default tolerances for six general categories of devices: resistor, capacitor, inductor, DC source, transistor (beta forward), and digital Tp (propagation delay for digital devices).

Tolerances can be specified as actual values, or as percentages. For example, you can enter a resistor tolerance as 10 or 10%. If a 1kohm resistor has a tolerance of 10, it varies between 900 and 1010 ohms. With a tolerance of 10%, a 1kohm resistor varies between 900 and 1100 ohms.

Each device is randomly varied independent of other devices. For example, if a circuit has two 10kohm resistors, and the default tolerance is set to 10%, then during the first pass of the simulation, one resistor might have a value of 953 ohms, and the other one could be 1022 ohms. The simulator uses a separate and independent random number to generate the value for each device.

Specific Device Tolerances

You can also override Default Tolerances with specific device tolerances. To add a Specific Device Tolerance click RIGHT MOUSE in the Specific Device Tolerance region of the dialog, and select Add from the pop-up menu that appears (shortcut: press the INSERT key). The Monte Carlo Device and Lot Tolerances dialog will pop up. The attributes of this dialog are:

Designator

Select the required circuit device.

Parameter

Include a parameter if the device requires it. Supported parameters include; the propagation delay of a digital component, the Beta forward of a transistor, and the resistance of a potentiometer.

Device Tolerance

Tolerance of this device.

Device Tracking Number

Assign a common tracking number to devices when you require the variation in their tolerance to be correlated. If you give two devices the same Device Tracking Number and Device Distribution then the same random number is used for both devices when the device values for a simulation run are calculated.

Device Distribution

Select the distribution kind for this device, as described earlier in this topic.

Lot Tolerance, Tracking and Distribution

These settings are used in exactly the same way as the Device settings. They provide a second way of defining and correlating device tolerances. Both device and lot tolerances are allowed, but only one or the other is required. The simulator calculates device and lot tolerances independently (using different random numbers) and then adds them together.

Combined device and lot tolerances are useful where values are not completely correlated, but are not completely independent either. An example would be two different resistor packs. Here, the lot tolerance can be large (that is, the variation from wafer to wafer), while the device tolerance (the variation from resistor to resistor in the same package), is small. In this case the device tolerance should not be ignored because it may limit the overall performance of a circuit.

Consider the following example:

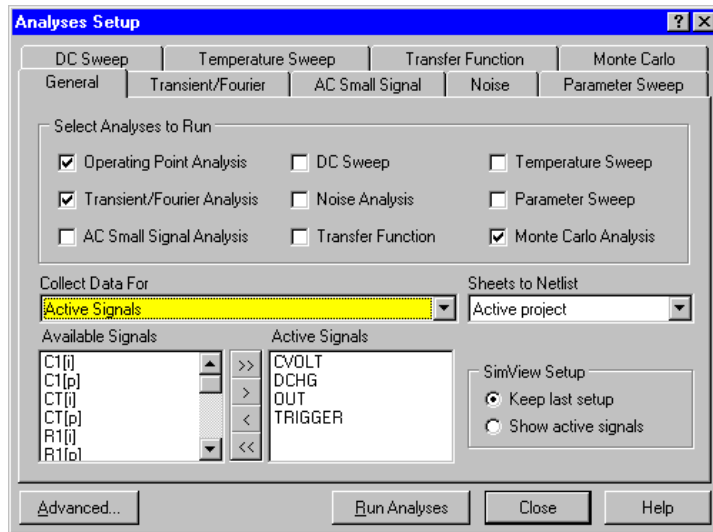
Assume R1 and R2 are both 1k, with a Device Tolerance of 1% (no device tracking) and they have a Lot Tolerance of 4%, with the same Lot Tracking number. For each Monte Carlo run the resistors are first assigned the same lot variation (a nominal value) between +/- 4%. Then each resistor is assigned a device tolerance between +/- 1%. This gives a total tolerance of 5% (1% + 4%).

However, during the same run the values of each resistor cannot be any farther than +/- 1% from their nominal value, or 2% from each other.

Running a Monte Carlo Analysis

To run a Monte Carlo analysis:

1. Set up the Monte Carlo analysis parameters as described above.
2. Enable the Monte Carlo Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).



3. Monte Carlo analysis can result in a large amount of data being calculated. To limit the amount of data that is calculated you can set the Collect Data For option in the Analyses setup dialog to Active Variables. With this option Monte Carlo data is only calculated for the Variables currently listed in the Active Variables field. In the figure shown above this would mean that Monte Carlo data is only collected for the OUTPUT variable.
4. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

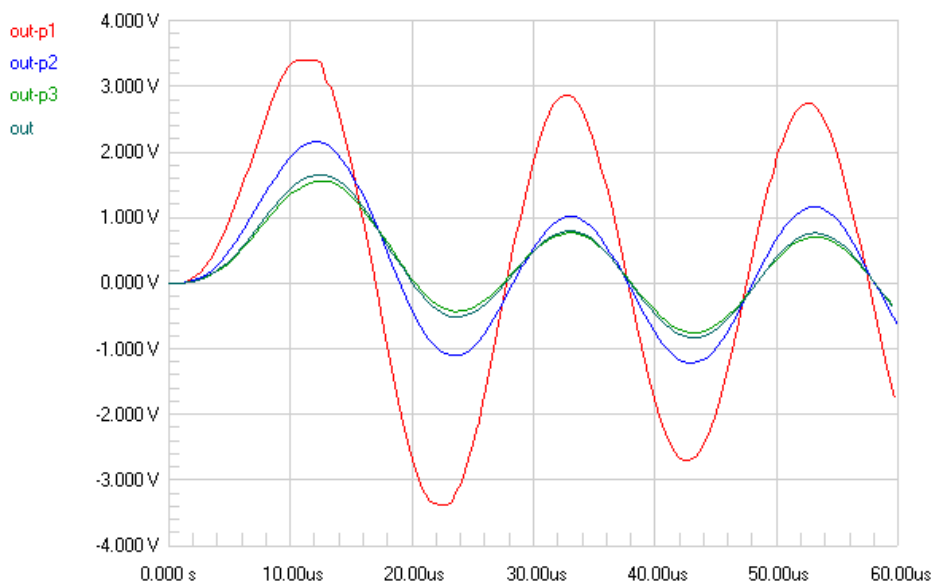
Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

The simulator displays the results of the Monte Carlo analysis in the AC, DC, or Transient analysis window(s), depending on which analyses were enabled. You can then view and measure the Monte Carlo analysis results from the waveforms in the analysis window. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

Parameter Sweep

The Parameter Sweep feature allows you to sweep the value of a device in defined increments, over a specified range. The simulator performs multiple passes of the enabled analyses (AC, DC, or Transient). The Parameter Sweep can vary basic components and models, note that subcircuit data is not varied during the analysis.

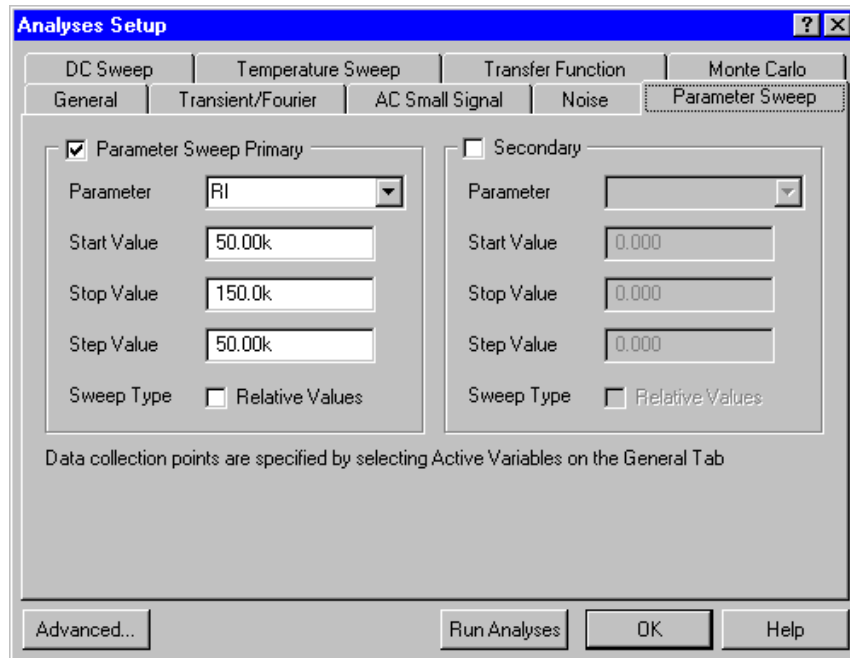
You can also define a Secondary parameter to be swept. When a Secondary parameter is defined the Primary parameter is swept for each value of the Secondary parameter. For example, if the step sizes are set so that there will be 3 passes of the Primary sweep and 3 passes of the Secondary sweep, the component values will be; P1 and S1, P2 and S1, P3 and S1, then P1 and S2, P2 and S2, P3 and S2, and so on.



The voltage at the node OUT, as one of the component values in the circuit is swept

Setting up for a Parameter Sweep

A Parameter Sweep is set up in the Parameter Sweep Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).



The Parameter Sweep requires the following data: Parameter, Start Value, Stop Value, and Step Value. The parameter can be a single designation (such as C2), or a designation with a device parameter in brackets (such as U5[tp_val]). Following are some valid examples:

Example	What it Varies
RF	Resistor with designation RF
Q3[bf]	Beta forward on transistor Q3
R3[r]	Resistance of potentiometer R3
option[temp]	Temperature
U5[tp_val]	Propagation delays of digital device U5

♦ Normally you would use a Temperature Sweep to vary the temperature for simulation; however, temperature can also be varied in the Parameter Sweep. This is useful if you want to vary the temperature as either the primary or secondary parameter in a two-parameter sweep.

When to enable the Relative Values Option

If you enable the Use Relative Values option in the Parameter Sweep Tab the values entered in the Start Value, Stop Value, and Step Value fields are added to the parameter's existing or default value. For example, consider a Parameter Sweep with the following conditions:

1. The parameter is a 1kohm resistor.
2. The Start, Stop, and Step fields are -50, 50, and 20, respectively.
3. You enable Use Relative Values.

The following resistor values would be used in the simulation runs: 950, 970, 990, 1010, 1030, and 1050.

Running a Parameter Sweep

To perform a Parameter Sweep,

1. Parameter Sweep analysis can result in a large amount of data being calculated. To limit the amount of data that is calculated you can set the Collect Data For option in the Analyses setup dialog to Active Variables. With this option data is only calculated for the Variables currently listed in the Active Variables field.
2. Set up the Parameter Sweep parameters as described above.
3. Enable the Parameter Sweep option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
4. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

The simulator displays the results of the Parameter Sweep in the AC, DC, or Transient analysis window(s), depending on which analyses were enabled. You can then view and measure the Parameter Sweep results from the waveforms in the analysis window. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

Temperature Sweep

A Temperature Sweep can be used in conjunction with one or more of the standard analyses; AC, DC, or Transient. The circuit is analyzed at each temperature in the specified range, producing a series of curves, one for each temperature setting.

Setting up a Temperature Sweep Analysis

A Temperature Sweep is set up in the Temperature Sweep Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).

Set up one or more standard analysis so that each analysis is performed at the indicated temperatures.

Running a Temperature Sweep Analysis

To run a Temperature Sweep analysis,

1. Temperature Sweep analysis can result in a large amount of data being calculated. To limit the amount of data that is calculated you can set the Collect Data For option in the Analyses setup dialog to Active Variables. With this option data is only calculated for the Variables currently listed in the Active Variables field.
2. Set up the Temperature Sweep parameters in the Temperature Sweep Tab of the Analyses Setup dialog.
3. Enable the Temperature Sweep option in the General Tab of the Analyses Setup dialog.
4. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

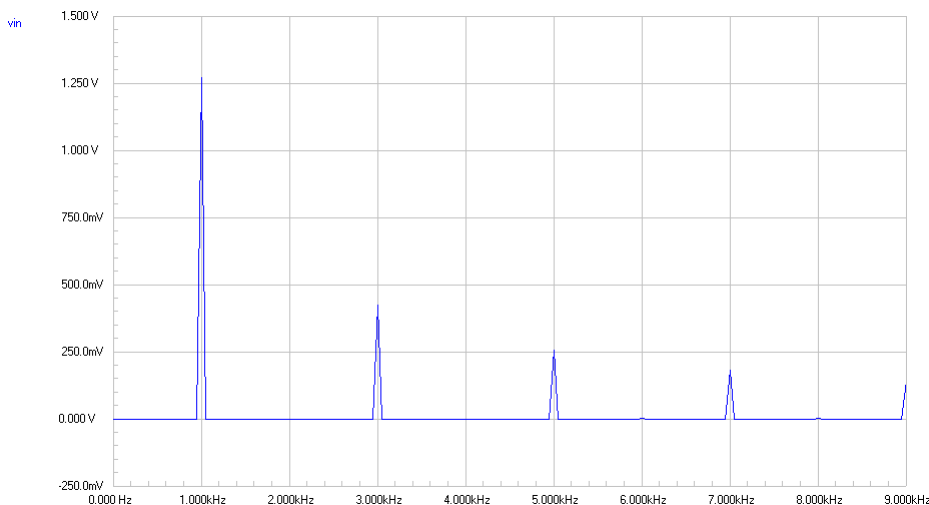
The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

The simulator displays the results of the Temperature Sweep in the AC, DC, or Transient analysis window(s), depending on which analyses were enabled. You can then view and measure the Temperature Sweep results from the waveforms in the analysis window. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

Fourier Analysis

The simulator can perform a Fourier analysis based on the last cycle of transient data that is calculated during a Transient analysis. For example, if the fundamental frequency is 1.0kHz, then the transient data from the last 1ms cycle would be used for the Fourier analysis.



The results of a Fourier analysis, showing the frequency spectrum of a 1KHz square wave

Setting up a Fourier Analysis

Fourier analysis is set up in the Transient/Fourier Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**). You must enable the Transient analysis in order to do the Fourier analysis. Enter the Fundamental Frequency of the analysis, and the number of Harmonics required.

Running a Fourier Analysis

To run a Fourier analysis,

1. Set up the Fourier analysis parameters as described above.
2. Enable the Transient/Fourier Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would

like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view and measure the Fourier analysis waveforms in the analysis window that is displayed. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

◆ Refer to the simulation information file (*DesignName.SIM*) for details of the magnitude and phase of each harmonic.

Transfer Function Analysis

The Transfer Function analysis calculates the DC input resistance, DC output resistance, and DC gain.

Setting up a Transfer Function Analysis

Transfer Function analysis is set up in the Transfer Function Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**). Select the Source to be used as the input reference for the calculations, and the node that the calculations are referenced to (the default is 0).

Running a Transfer Function Analysis

To run a Transfer Function analysis,

1. Set up the Transfer Function analysis parameters as described above.
2. Enable the Transfer Function option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view the DC input resistance, DC output resistance, and DC gain at each node in the circuit on the Transfer Function Tab of the analysis window. Refer to the chapter, *The Waveform Analysis Window*, for more information on working in this window.

Noise Analysis

Noise analysis lets you measure the noise in your circuit due to noise contributions of resistors and semiconductor devices. The simulator can plot the Noise Spectral Density, which is the noise measured in Volts squared per Hertz (V^2/Hz). Capacitors, inductors, and controlled sources are treated as noise free. The following noise measurements can be made in the simulator:

Output Noise

The noise measured at a specified output node.

Input Noise

The amount of noise that, if injected at the input, would cause the calculated noise at the output. For example, if the output noise is 10p, and the circuit has a gain of 10, then it would take 1p of noise at the input to measure 10p of noise at the output. Thus the equivalent input noise is 1p.

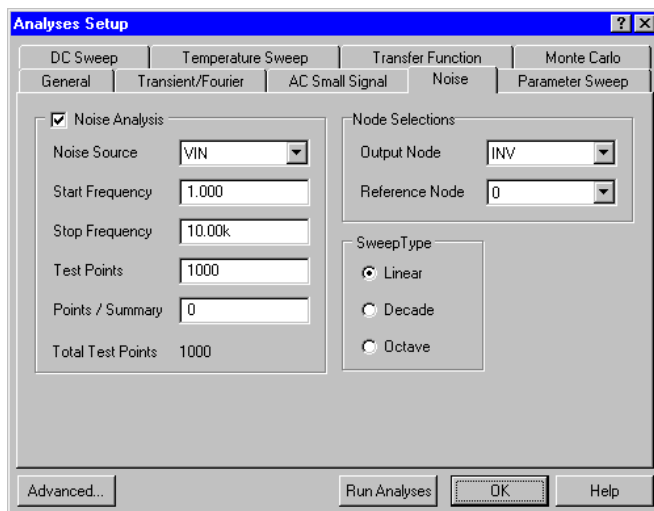
Component Noise

The output noise contribution of each component in the circuit. The total output noise is the sum of individual noise contributions of resistors and semiconductor devices. Each of these components contributes a certain amount of noise, which is multiplied by the gain from that component's position to the circuit's output. Thus the same component can contribute different amounts of noise to the output, depending on its location in the circuit.

Setting up a Noise Analysis

Noise analysis is set up in the Noise Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).

Select the Source to be used as the input reference for the noise calculations. Enter the Start and Stop Frequencies and the number of Test Points that the calculations will be performed at. The distribution of these test points over the frequency range is defined by the Sweep Type. Enter a 0 in the Points Summary field to measure only input and output noise, enter a 1 to measure the noise contribution of each component.



Running a Noise Analysis

To run a Noise analysis,

1. Set up the Noise analysis parameters as described above.
2. Enable the Noise Analysis option in the General Tab of the Analyses Setup dialog (select **Simulate » Analyses Setup**).
3. After enabling this option you can either press the Run Analyses button at the bottom of the dialog, or select the **Simulate » Run** menu item to start the simulation process.

The simulation progress is displayed on the status bar. If an error is detected during netlisting the simulation is stopped and a message box appears, asking if you would like to view the error file. Review this file and correct any errors. For more information refer to the chapter, *Working with Circuits that will not Simulate*.

Once the design is netlisted the waveform window will appear, displaying the simulation results as they are calculated.

You can then view the Noise analysis waveforms in the analysis window that is displayed. The Input and output noise waveforms are labeled NI(*output node*) and NO(*output node*).

Impedance Plot Analysis

An Impedance Plot analysis shows the impedance seen by any two-terminal source. Normally plotted in the AC Analysis window, an Impedance Plot does not have a separate setup dialog box.

Setting up an Impedance Plot Analysis

To include Impedance Plot analysis results select either of the two following options in the Collect Data For drop down list in the General Tab of the Analyses Setup dialog:

- Node Voltages, Supply and Device Currents
- Node Voltages, Supply Currents, Device Currents and Powers

Locate the source of interest in the Available Variables list and add it to the Active Variables list. A source with a designator of VIN would appear in the list as @VIN(z).

Running an Impedance Plot Analysis

When you run the simulation an impedance plot will appear in the Analyses windows. This is especially useful to the impedance versus frequency in the AC Analysis window. The impedance measurement is calculated from the voltage at the supply's positive terminal, divided by the current out of that same terminal.

You can also do an impedance plot of the circuit's output impedance. To measure the circuit's output impedance:

Mixed-Signal Simulation

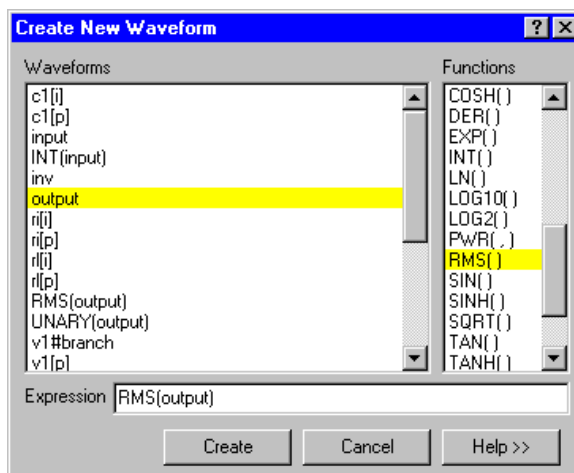
1. Remove the source from the input.
2. Ground the circuit's inputs where the input supply was connected.
3. Remove any load connected to the circuit.
4. Connect a two-terminal source to the output, with the source's positive terminal connected to the output and the source's negative terminal connected to ground.
5. Setup the Variables as described earlier.
6. Run the desired simulation.

For Impedance Plots, you would normally change the Y axis to Magnitude. To do this the waveform display must be in single cell mode. Right-click in the waveform window and select Scaling from the pop up menu. Set the Y Axis to Magnitude in the Scaling Options dialog.

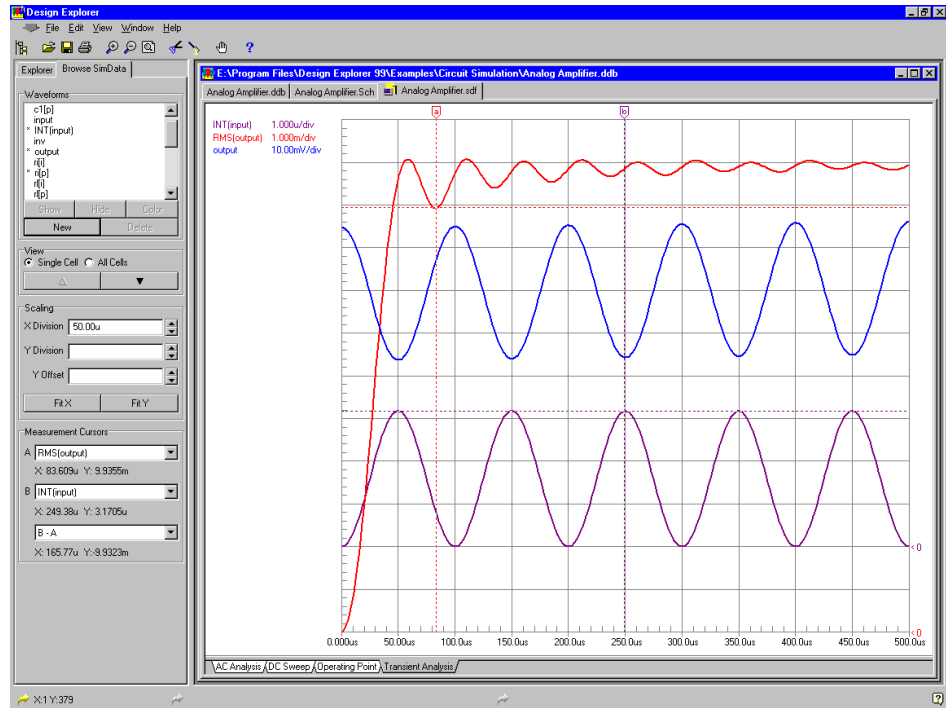
Mathematical Functions and Waveforms

As part of the analysis of your design you may want to perform a mathematical operation on one or more of the simulation signals, and view the resultant waveform. This feature is an integral part of the simulator's waveform viewer, you can construct a mathematical expression based on any signal available in the simulated circuit.

To define a mathematical function click the New button in the Browse SimData Panel, which pops up the Create New Waveform dialog. There are 3 parts to the dialog; a list of available Functions, a list of currently available Waveforms, and an Expression building field. The Expression can be constructed by either typing it in directly, or by clicking to select a function in the Functions list, then clicking to select the signal that you want to apply that function to.



Build the mathematical expression based on any waveform



Create mathematical expressions and display the results with the signal waveforms

Listing of Functions and Operators

Formulae can be based on any available waveform and support the following operators and functions:

Operator/Function	Description
+	Addition operator.
-	Subtraction operator.
*	Multiplication operator.
/	Division operator.
^	Power operator, y^x returns the value of "y raised to the power of x." Same as PWR(,).
()	Precedence Indicators. Use to set precedence of math operations. Operations contained within () will be performed first.
ABS()	Absolute value function. ABS(x) returns the value of $ x $.

Mixed-Signal Simulation

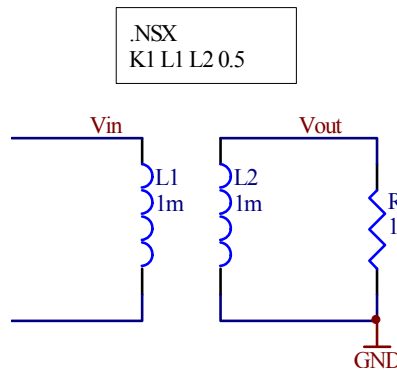
ACOS()	Arc cosine function.
ACOSH()	Hyperbolic arc cosine function.
ASIN()	Arc sine function.
ASINH()	Hyperbolic arc sine function.
ATAN()	Arc tangent function.
ATANH()	Hyperbolic arc tangent function.
AVG()	Average function. Returns the running average of the wave data.
BOOL(,)	Boolean function. In the expression BOOL(wave, thresh), wave would be the name of a waveform, thresh would be the switching threshold. Returns a value of one for wave arguments greater than or equal to thresh and a value of zero for wave arguments less than thresh.
COS()	Cosine function.
COSH()	Hyperbolic cosine function.
DER()	Derivative function. dx/dt . Returns the slope between datapoints.
EXP()	Exponential function. EXP(x) returns the value of "e raised to the power of x", where e is the base of the natural logarithms.
INT()	Integral function. Returns the running total of the area under the curve.
LN()	Natural logarithm function. Where $LN(e) = 1$.
LOG10()	Log base 10 function.
LOG2()	Log Base 2 function.
PWR(,)	Power function. Same as ^ operator. PWR(y,x) returns the value of "y raised to the power of x."
RMS()	Root-Mean-Square function. Returns the running AC RMS value of the wave data.
SIN()	Sine function.
SINH()	Hyperbolic sine function.
SQRT()	Square root function.
TAN()	Tangent function.
TANH()	Hyperbolic tangent function.
UNARY()	Unary minus function. UNARY(x) returns -x.
URAMP()	Unit ramp function. Integral of the unit step: for an input x, the value is zero if x is less than zero, or if x is greater than zero, the value is x.
USTEP()	Unit step function. Returns a value of one for arguments greater than zero and a value of zero for arguments less than zero.

Including extra SPICE Information in the Netlist

Extra SPICE simulation information can be included in the netlist by placing a Text Frame on the schematic sheet. The first line of the text frame must be the string “.NSX”. All subsequent lines in the text frame will be included in the simulation netlist, exactly as they are written.

Example of Using a Text Frame to Specify the Inductor Coupling

The example below demonstrates using a text frame to specify the inductor coupling between 2 discrete inductors. In this example K1 is a reference for the inductor coupling (it can be anything, as long as it starts with the letter K), L1 and L2 are the designators of the 2 coupled inductors, and 0.5 is the coupling coefficient.



Creating and Simulating from the Netlist

When you run a simulation from a schematic the design information is passed to the simulation engine in the form of a netlist. This netlist is created as a temporary file, and automatically deleted once the simulation is complete.

You can also simulate directly from the netlist. To do this the netlist must be currently open in the document editing window (there is a Tab for each document that is currently open, check for the appropriate .NSX Tab). It does not need to be the active document. To create, view and edit the netlist select **Simulate » Create Spice Netlist** from the menus.

Note: A netlist that stored in the Design Database is not used for simulation – if a netlist is not currently open then a temporary netlist is created and used, if a netlist is currently open then it is used.

Specifying Initial Circuit Conditions

Certain designs, such as astable and bistable circuits, may require node voltages to be pre-defined before the simulation will converge. The simulator includes a number of features to do this, including the Nodeset and Initial Condition devices, and the Initial Conditions option. The Nodeset and Initial Condition devices are placed from the Simulation Symbols.lib library.

.NS Device (Nodeset)

The Nodeset device is used to specify the starting voltage for a node in the circuit during a preliminary pass of the operating point analysis. After the preliminary pass the restriction is released and the iterations continue to the true bias solution. After you place the device you will need to specify the following attributes:

Attribute / Part Field	Setting
Designator	Each Nodeset device must have a unique designator
Part Type	Amplitude of the node voltage (eg 12)

NS1

12



.IC Device (Initial Condition)

The Initial Condition device is used for setting transient initial conditions. The way that the simulator determines the initial conditions depends on the IC devices, and the setting of the “Use initial Conditions” option in the Transient/Fourier Tab of the Analyses Setup dialog.

If the Use Initial Conditions option *is not enabled*, the node voltage is held at the value specified by the IC device during operating point analysis. During the subsequent transient analysis this constraint is removed. This is the preferred method since it allows the SPICE engine to compute a consistent DC solution.

Attribute / Part Field	Setting
Designator	Each IC device must have a unique designator (eg IC1)
Part Type	Amplitude of the node voltage (eg 5)

IC1

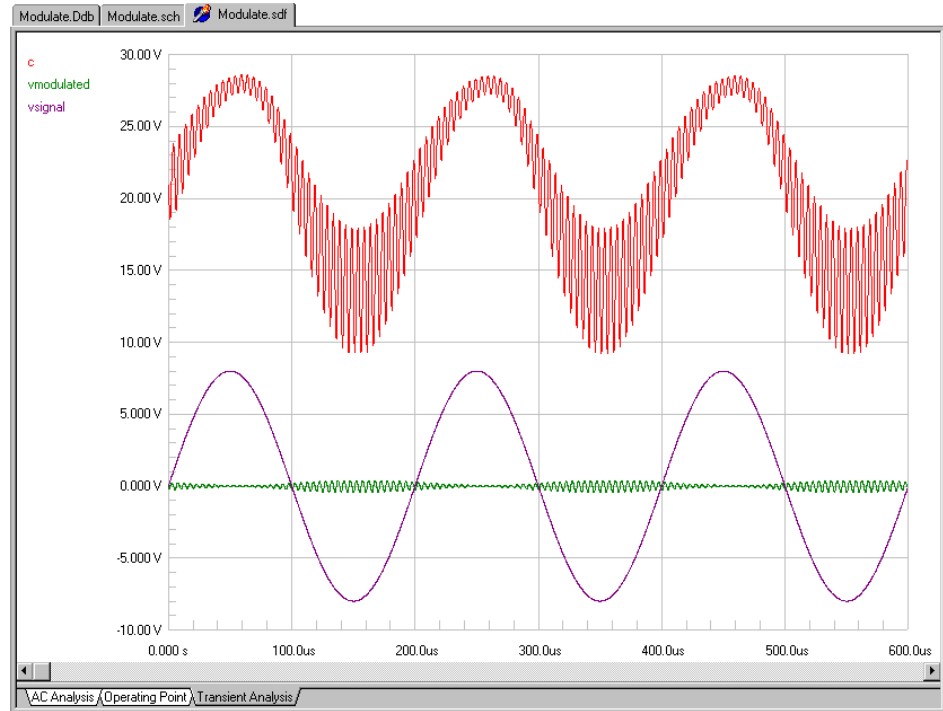
12



Using Initial Conditions Option

If the Use Initial Conditions option *is enabled* in the Transient/Fourier Tab of the Analyses Setup dialog an operating point analysis is *not* performed. Instead the node voltages specified by the .IC devices are used to compute the capacitor, diode, BJT, JFET, and MOSFET initial conditions. As there is no operating point analysis performed you must specify all the appropriate node voltages. Refer to the *Transient Analysis* topic in the *Setting up and Running a Simulation* chapter for information on enabling the Use Initial Conditions option.

The Waveform Analysis Window



The simulator displays simulation data and waveforms using a multi-tabbed Waveform analysis window, through which you can quickly and easily analyze the simulation results.

The results from each enabled analysis are displayed on a separate Tab in the Waveform analysis window. For more information on setting up each type of analysis refer to the chapter *Setting Up and Running a Simulation*.

The Waveform analysis window operates much like an oscilloscope. Simply adjust the scale options in the Panel to show exactly the part of the waveform that you would like to examine. The Waveform analysis windows also includes measurement cursors that you can use to take measurements directly from the waveforms.

Displaying Waveforms

The simulation results are plotted while the simulation is being carried out. Once the simulation is complete you can click on the appropriate Tab at the bottom of the

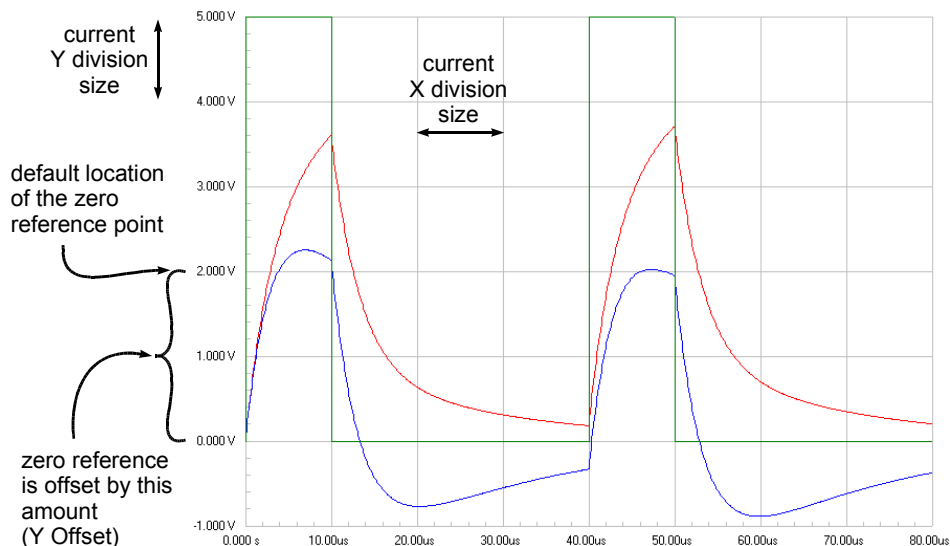
Window to display the results for that type of analysis. Operating Point results are displayed as a list of voltage, current and power calculations for nodes or devices. Note that the Operating Point results that are displayed depend on the Collect Data setting in General Tab of the Analyses Setup dialog.

◆ Wondering how to specify the nodes in the circuit that you want to be displayed in the waveform window? Refer to the topic *Specifying the Simulation Data that you want Collected, Displayed and Stored* in the *Setting up and Running a Simulation* chapter.

Resizing the Waveforms

The waveforms are automatically scaled when they are first displayed. In the Y direction all the waveforms are scaled by the same amount, such that all the waveforms are visible, and the largest waveform almost fills the window. The X direction will be scaled according to setting for that type of analysis. For example, the total width of the window in the X direction for a transient analysis is defined by *Start Time – Stop Time*.

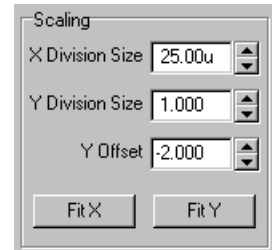
Use the scaling controls on the Panel to change the size or position of the waveforms (select **View » Panel** to display the panel).



Use the Scaling controls to change the scale

Resizing in the X Direction

Change the X Division Size on the Panel to expand or contract the waveforms horizontally. You can change the scaling by either entering a new number, or by pressing the small up and down arrows that are next to each scaling box.



Resizing in the Y Direction

Change the Y Division Size on the Panel to expand or contract the waveforms vertically. Note how the position of the scaled waveforms is affected by the current setting of the Y Offset. Temporarily set the Y Offset to zero if the waveform disappears as you change the Y Division Size.

Adjusting the Y Offset

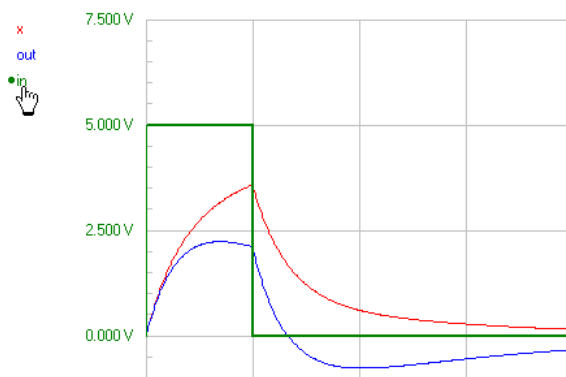
The Y Offset is the amount the zero reference of the waveform is displaced along the Y axis. The default location of the zero reference is in the center of the Y axis. A negative value Y Offset indicates that the zero reference has been moved down, a positive value Y Offset indicates that the zero reference has been moved up. Enter a new Y Offset, or use the small up and down arrows to adjust it.

Resetting the Waveform Scaling

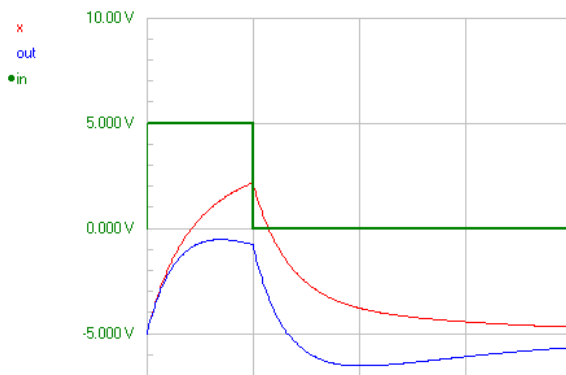
To reset the scaling for all the waveforms to be the same; first make sure that none of the waveforms are selected (none has a dot next to its name), then click the Fit Y button on the panel.

Resizing an Individual Waveform

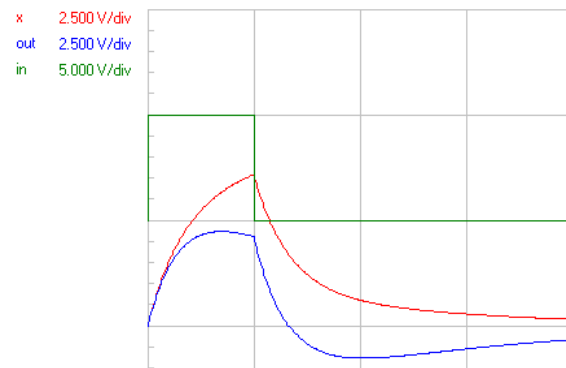
Before you can scale an individual waveform you must select it. Select the waveform by clicking on its name in the list at the side of the waveforms. Note how the scale color changes to indicate that the scale only applies to the selected waveform.



Use the Scaling controls to re-scale the selected waveform only. You can change both the Y Division Size, and the Y Offset. In this example the Y Division size has been changed from 2.5V/div to 5V/div, and the Y Offset changed to move the “In” waveform up.



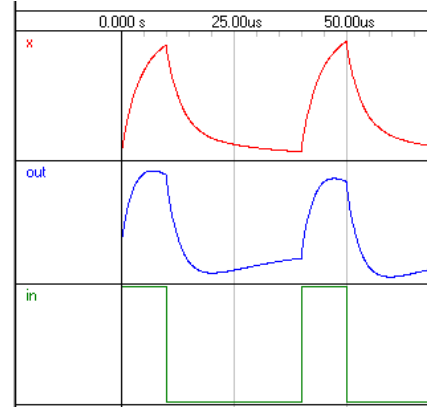
The “In” waveform has been de-selected by clicking on its name again. Note how the scale along the Y axis has disappeared, instead each name has its scale shown next to it. This only happens when they are scaled differently.



Viewing Waveforms in Separate Cells

To view the waveforms with each shown in a separate region (or cell), click on the All Cells option in the Panel. Use this feature when you need to view digital signals separately.

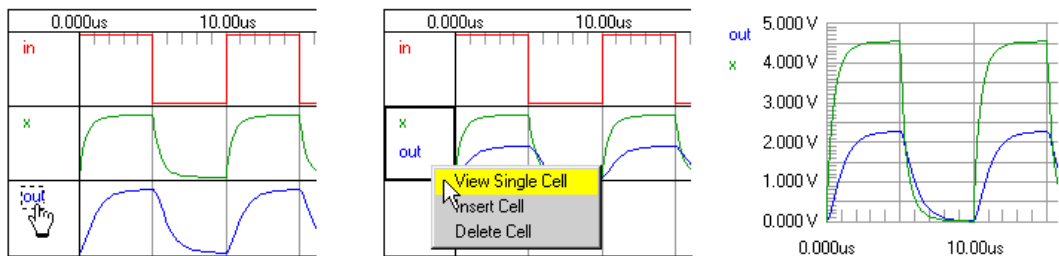
- ◆ Right mouse click in a cell to toggle back and forth from All Cells, to Single Cell.



How the waveforms appear in separate cells

Displaying Multiple Waveforms in the same Cell

When you first simulate a circuit the default behavior is to display the selected waveforms, with each shown in a separate cell. You can easily merge waveforms to share a cell, simply click and drag on a waveform, dropping it into another cell. When you toggle the view to display as a Single Cell, these 2 waveforms will be shown together. To toggle the view right-click and select **View Single Cell** from the floating menu.



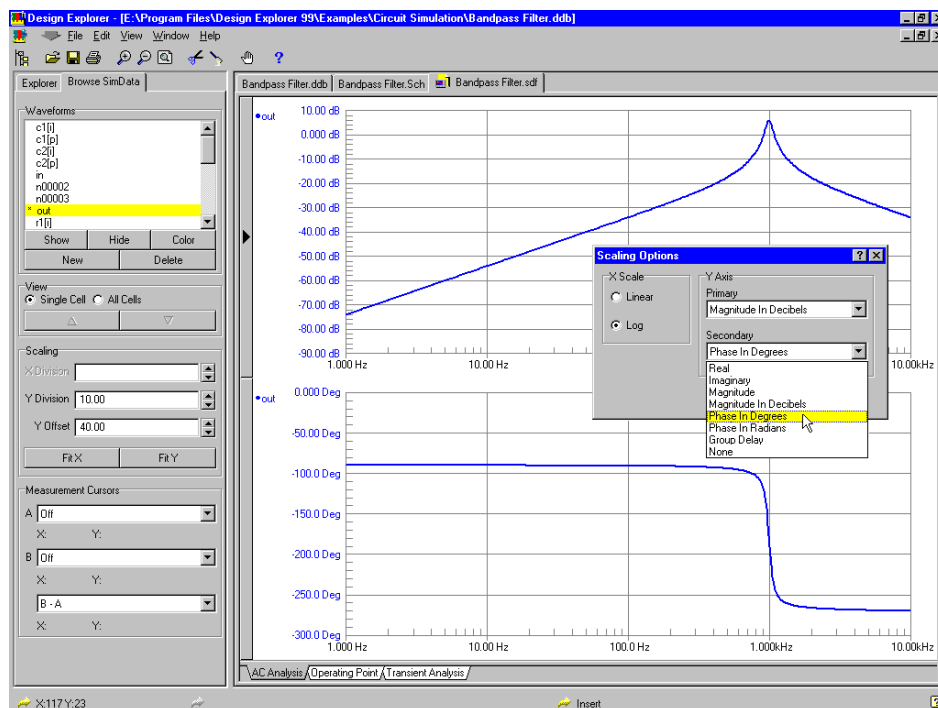
Click, drag and drop – to show 2 waveforms in the same cell – then right-click to toggle to single cell mode

Displaying a Waveform Scaled Two Ways

Often you need to analyze and simultaneously display the same waveform scaled in different ways, for example frequency and phase, or frequency and group delay.

To do this first change the waveform display to single cell, then right-click and select **Scaling** from the floating menu. Set the primary Y axis scaling as required, then add the second scaling by setting the Secondary field as required. The single cell window will be split horizontally displaying the 2 versions of the waveform.

Measurement cursors can be added in the normal way, either through the panel, or by right-clicking on a waveform name in the window and selecting from the floating menu. There is also a selection region on the left of the waveform window, click on this to set the currently active view.



Viewing the frequency and phase response at the same time

Scaling the Waveforms Axes

AC Analysis waveforms can be displayed in a number of modes, including linear or logarithmic along the X axis; and absolute, magnitude or phase values along the Y axis. To change the scaling select **Options » Scaling** from the menus when a waveform is displayed. You can also display the Scaling Options dialog by selecting Scaling from the floating menu that appears when you right-click in the Waveform analysis window.

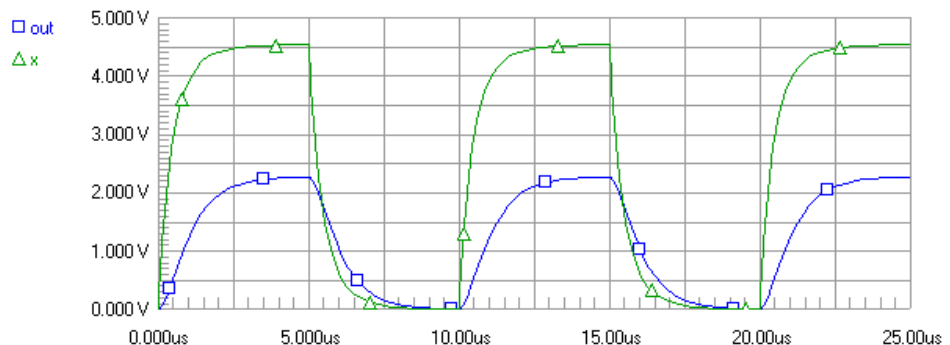
Displaying the Simulation Data Points

If you are unsure about the accuracy of the waveforms, perhaps they look sharp and jagged instead of smooth and curved, you can turn on the simulation data points to check if the results have been calculated often enough.

To display the simulation data points select **Design » Options** from the menus and enable the Show Data Points option in the Document Options dialog. A small circle will be displayed at each point that data was calculated.

Identifying Waveforms on a Single Color Printout

The Waveform analysis window includes a feature to easily identify each waveform on a single color printout. When you select **Design » Options** from the menus, and enable the Show Designation Symbols option, identifier symbols are added to each waveform. If the waveforms are shown in individual cells, then all waveforms use a square symbol. If 2 or more waveforms share a cell, then a different shape is used for each waveform.



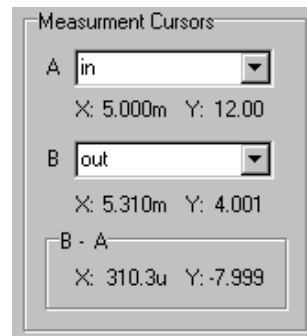
Use the designation symbols to clearly identify each waveform

Using the Measurement Cursors

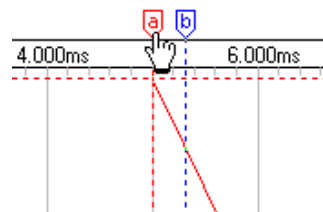
There are two measurement cursors that let you take measurements directly from the waveforms. You take measurements of time and amplitude from the transient analysis results, or find the frequency of the 3dB point from the AC small signal analysis results. You can also use the two cursors together to take difference measurements from two points on the same waveform, or two points on different waveform.

Each of the two measurement cursors can be attached to any of the visible waveforms. To use the measurement cursors:

1. Select the waveform that you wish to attach a cursor to in the drop-down list.
2. A small Tab with the letter of that cursor will appear at the top of the window. Click and drag this Tab to change the location of the cursor.
3. Read the X and Y values of the current cursor point from the Measurement Cursors region of the panel.



Attach a cursor to a waveform by picking the waveform name in the list



Position the cursor by dragging the Tab at the top of the window

How Multi-pass Results are Displayed

Monte Carlo, Parameter Sweep and Temperature Sweep actually perform multiple passes of the analysis, varying one or more circuit parameters with each pass. Each pass is identified by adding a letter and a number after the waveform name. The letter signifies the type of multi-pass analysis (**m** for Monte Carlo, **t** for temperature, and **p** for parameter sweep), and the number signifies the pass. There is also a waveform for the simulation run which was done with the nominal circuit values, which does not have a character and number appended to its label. Following are some examples:

output-p1	Voltage at node <i>output</i> for Parameter Sweep run 1
output-p1	Voltage at node <i>output</i> for Parameter Sweep run 2
output-p1	Voltage at node <i>output</i> for Parameter Sweep run 3
output	Voltage at node <i>output</i> for nominal run
output-m1	Voltage at node <i>output</i> for Monte Carlo run 1
output-t1	Voltage at node <i>output</i> for Temperature Sweep run 1

Voltage and Current Sources

Before you can simulate your design you must include appropriate power and excitation sources.

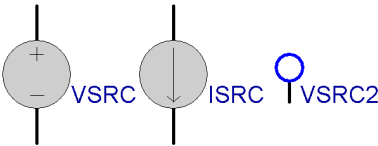
◆ You do not need to enter units (V or A) when you are setting up the source. Include a scale suffix if required (eg. meg, u, k, etc). Refer to the *Component and Simulation Values* topic in the *Getting Started with Simulation* chapter for more information on scale suffixes.

Locating the Source Components

You can place sources from the **Simulate » Sources** sub menu, or place them directly from the Simulation Symbols library in the \Program Files\Design Explorer 99 SE SE\Library\Sch\Sim.ddb Library Database. The image that is shown with each source includes the name of that source in this library. After placing the source double-click on it to set the attributes.

Constant Source

Use the constant source to power your circuit. After you place the source you will need to specify the following attributes of the Source component.

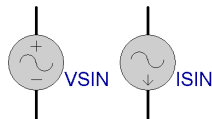


Attribute / Part Field	Setting
Designator	Set to the required supply rail name (eg VDD)
Part Type	Amplitude of the source, voltage or current (eg 12)
AC Magnitude (V or A)	Set this if you plan to do a small signal AC analysis based on this source (typically 1 volt)
AC Phase (degrees)	Phase of the small signal voltage

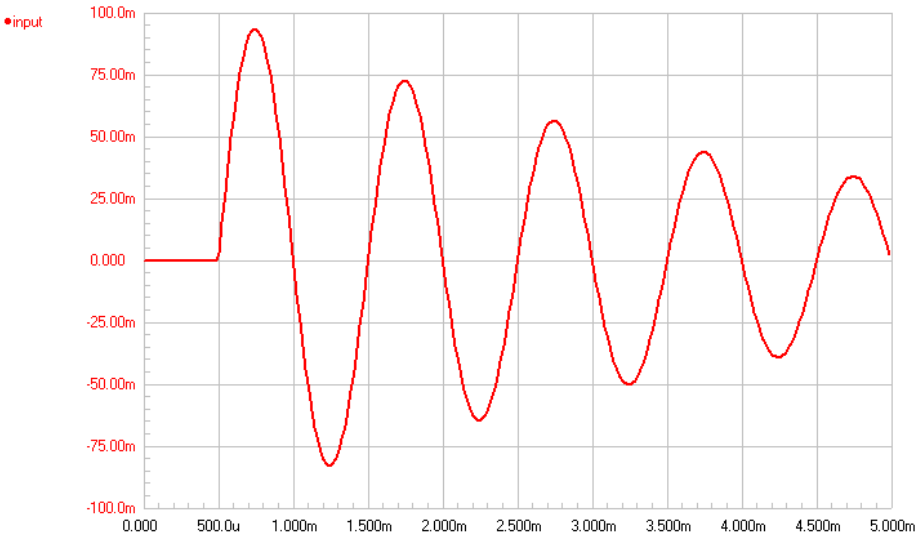
Note: VSRC2 references GND by default.

Sinusoidal Source

Use this source to create sinusoidal waveforms. After you place the source you will need to specify the following attributes in the Part Fields of the Source component.



Attribute / Part Field	Setting
Designator	Set to the required source name (eg INPUT)
DC (V or A)	DC offset used in operating point analysis
AC (V or A)	Set this if you plan to do an AC small signal analysis (typically 1 volt)
AC Phase (degrees)	Phase of the AC small signal voltage or current
Offset (V or A)	DC offset voltage or current
Amplitude (V or A)	Peak amplitude of the sinusoid (eg 100m)
Frequency (Hz)	Frequency of the sinusoidal voltage or current (eg 1000)
Delay (s)	Delay time until the source voltage commences (eg 500u)
Damping Factor (1/s)	Sets the rate at which the sinusoid decreases in amplitude (eg 250)
Phase (degrees)	Phase shift of the sinusoid at time zero (eg 0)



Example of the waveform produced by a Sinusoidal Voltage Source (VSIN) with the attributes set to the example values in the table above.

Definition of the Sine Wave

The waveform, beginning at Start Delay, is described by the following formula where t = instance of time:

$$\begin{aligned} V(t_0 \text{ to } t_{SD}) &= VO \\ V(t_{SD} \text{ to } t_{STOP}) &= VO + VA \sin(2\pi F (t-SD)) e^{-(t-SD)THETA} \end{aligned}$$

DC Offset (VO)

Used to adjust the DC bias of the signal generator with respect to the negative terminal (usually ground), measured in volts or amps.

Peak Amplitude (VA)

Maximum amplitude of the output swing, excluding the DC Offset, measured in volts or amps.

Frequency (F)

Frequency of the output in hertz.

Start Delay (SD)

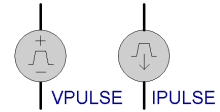
Provides a phase shift of the output by delaying the start of the sine wave.

Damping Factor (THETA)

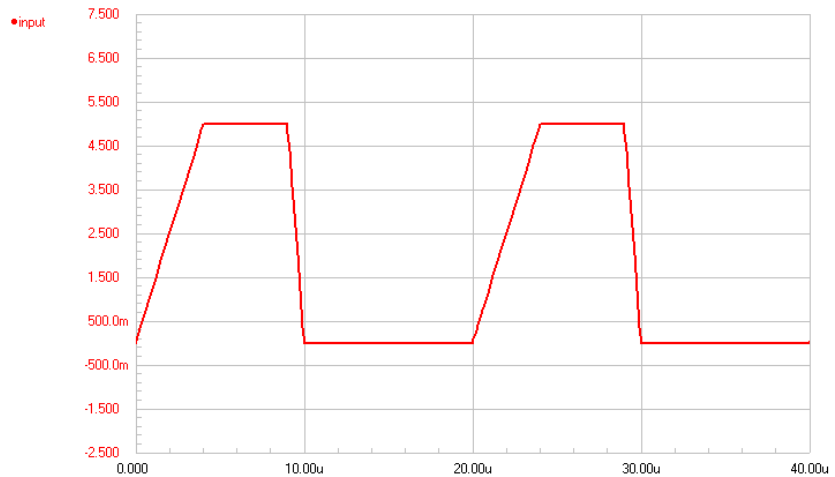
A positive value results in an exponentially decreasing amplitude; a negative value results in an exponentially increasing amplitude.

Periodic Pulse Source

Use this source to create a repeating sequence of pulses. After you place the source you will need to specify the following attributes in the Part Fields of the Source component.



Attribute / Part Field	Setting
Designator	Set to the required source name (eg INPUT)
DC (V or A)	DC offset used in operating point analysis
AC (V or A)	Set this if you plan to do a small signal AC analysis (typically 1 volt)
AC Phase (degrees)	Phase of the small signal voltage or current
Initial (V or A)	Voltage or current at time zero (eg 0)
Pulsed (V or A)	Voltage or current at time Delay+Rise Time (eg 5)
Time Delay (s)	Delay before the source changes from Initial to Pulsed
Rise Time (s)	Time it takes to rise from Initial Voltage to Pulsed Voltage, must be >0 (eg 4u)
Fall Time (s)	Time it takes to fall from Pulsed Voltage back to Initial Voltage, must be >0 (eg 1u)
Pulse Width (s)	Time that the source remains at Pulsed Voltage (eg 0)
Period (s)	Time between the start of the first pulse to the start of the second pulse (eg 5u)



Example of the waveform produced by a Periodic Pulse Source with the attributes set to the example values in the table above.

Definition of the Pulse Shape

The waveform is described as follows where t = instance of time. Intermediate points are set by linear interpolation:

$$\begin{aligned}
 V(t_0) &= VI \\
 V(t_{SD}) &= VI \\
 V(t_{SD}+t_{TR}) &= VP \\
 V(t_{SD}+t_{TR}+t_{PW}) &= VP \\
 V(t_{SD}+t_{TR}+t_{PW}+t_{TF}) &= VI \\
 V(t_{STOP}) &= VI
 \end{aligned}$$

Initial Amplitude (VI)

Initial amplitude of the output with respect to the negative terminal (usually ground), measured in volts or amps.

Pulse Amplitude (VP)

Maximum amplitude of output swing, in volts or amps.

Period (=1/freq)

Duration of one complete cycle of the output.

Pulse Width (PW)

Duration output remains at VP before ramping toward VI.

Rise Time (TR)

Duration of the ramp from VI to VP.

Fall Time (TF)

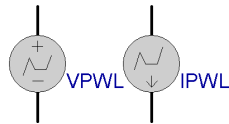
Duration of the ramp from VP to VI.

Delay to Start (SD)

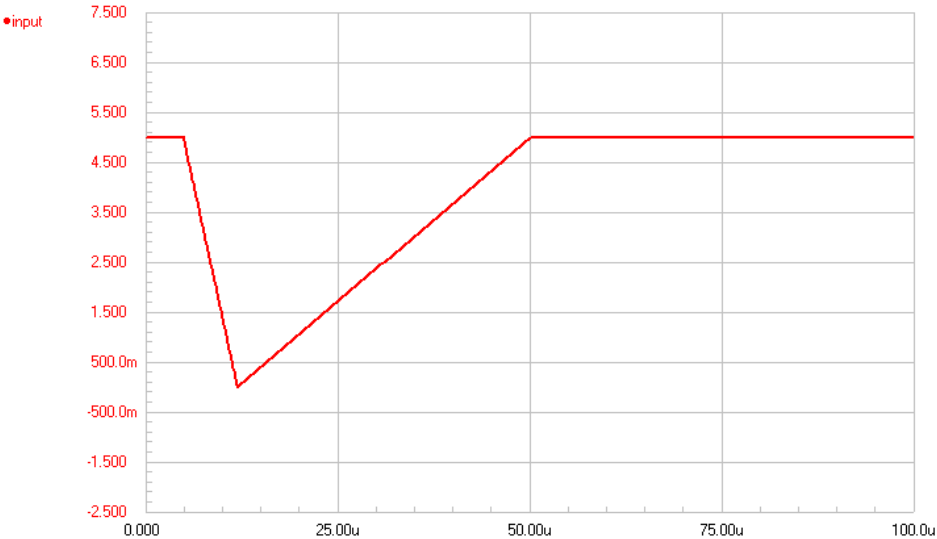
Duration that the output remains at VI before beginning to ramp toward VP the first time.

Piece-Wise-Linear Source

Use this source to create an arbitrary waveform as a set of voltages (or currents) at various points in time. After you place the source you will need to specify the following attributes in the Part Fields of the Source component.



Attribute / Part Field	Setting
Designator	Set to the required source name (eg INPUT)
DC (V or A)	DC offset used in operating point analysis
AC (V or A)	Set this if you plan to do a small signal AC analysis (typically 1 volt)
AC Phase (degrees)	Phase of the small signal voltage
Time – Voltage (or current) Pairs (seconds - V/A)	Specifies the voltage (or current) at each point in time (eg 0U 5V 5U 5V 12U 0V 50U 5V 60U 5V)
File Name	Use this option to reference a PWL waveform defined in an external file. The file must be in the same directory, with the extension .PWL.



Example of the waveform produced by a PWL source with the Time/Voltage attribute set to 0U 5V 5U 5V 12U 0V 50U 5V 60U 5V

Definition of the Piece-Wise Waveform

Piecewise linear data must come from one of two sources:

1. You can describe the waveform with a set of up to eight points that you enter directly into the Time/Voltage Part Field of the PWL source. The time specified for each successive point must be more positive than its predecessor. If it is not the cycle will end, excluding that and all successive points.
2. You can define the waveform in an ASCII text file containing an indefinite number of points. Values must be entered in pairs and each pair must include a time position followed by an amplitude. The first character of each data line must be a plus sign (+) and each line may contain up to a maximum of 255 characters. Values must be separated by one or more spaces or tabs. Values may be entered in either scientific or engineering notation. Comments may be added to the file by making the first character of the line an asterisk (*). For example:

* Random Noise Data

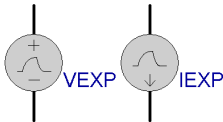
```
+ 0.00000e-3 0.6667 0.00781e-3 0.6372 0.01563e-3 -0.1177
+ 0.02344e-3 -0.6058 0.03125e-3 0.2386 0.03906e-3 -1.1258
+ 0.04688e-3 1.6164 0.05469e-3 -0.3136 0.06250e-3 -1.0934
+ 0.07031e-3 -0.1087 0.07813e-3 -0.1990 0.08594e-3 -1.1168
+ 0.09375e-3 1.4890 0.10156e-3 -0.2169 0.10938e-3 -1.4915
+ 0.11719e-3 1.4914 0.12500e-3 0.1486
```

Intermediate points are determined by linear interpolation.

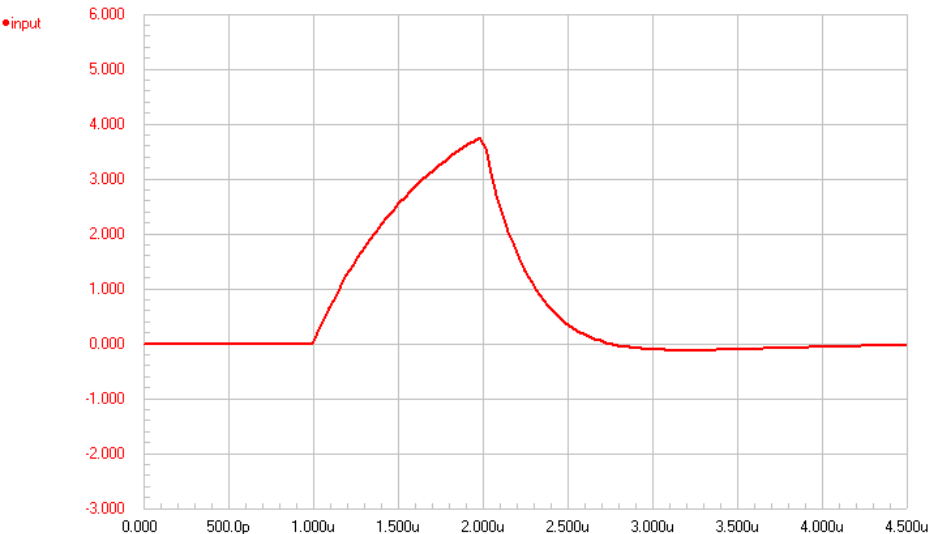
Note: The .PWL file must be located in the same folder in the Design Database as the circuit you are simulating.

Exponential Source

Use this source to create an exponential waveform. After you place the source you will need to specify the following attributes in the Part Fields of the Source component.



Attribute / Part Field	Setting
Designator	Set to the required source name (eg INPUT)
DC (V or A)	DC offset used in operating point analysis
AC (V or A)	Set this if you plan to do a small signal AC analysis (typically 1 volt)
AC Phase (degrees)	Phase of the small signal voltage or current
Initial Value (V or A)	Voltage or current at time zero (eg 0)
Pulse Value (V or A)	Maximum amplitude of the output swing (eg 5)
Rise Delay (s)	Delay before the source changes from Initial to Pulsed (eg 1u)
Rise Time (s)	RC charging time constant (eg 700n)
Fall Delay (s)	Time it takes to fall from Pulsed Voltage back to Initial Voltage, must be >0 (eg 2u)
Fall Time (s)	RC discharging time constant (eg 300n)



Example of the waveform produced by an Exponential Source with the attributes set to the example values in the table above.

Definition of the Exponential Waveform

The waveform is described by the following formulas where t = instance of time:

$$\begin{aligned} V(t_0 \text{ to } t_{RD}) &= VI \\ V(t_{RD} \text{ to } t_{FD}) &= VI + (VP - VI) (1 - e^{-(t - t_{RD}) / t_{RT}}) \\ V(t_{FD} \text{ to } t_{STOP}) &= VI + (VP - VI) (-e^{-(t - t_{RD}) / t_{RT}}) + (VI - VP) (1 - e^{-(t - t_{FD}) / t_{FT}}) \end{aligned}$$

Initial Amplitude (VI)

Initial amplitude of the output with respect to the negative terminal (usually ground), measured in volts or amps.

Pulse Amplitude (VP)

Max. amplitude of output swing, measured in volts or amps.

Rise Time Delay (RD)

The point in time, from t_0 , when the output begins to rise. This provides a phase shift of the output by delaying the start of the exponential waveform.

Rise Time Constant (RT)

Standard RC charging time constant.

Fall Time Delay (FD)

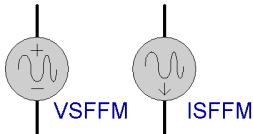
The point in time, from t_0 , when the output begins to fall.

Fall Time Constant (FT)

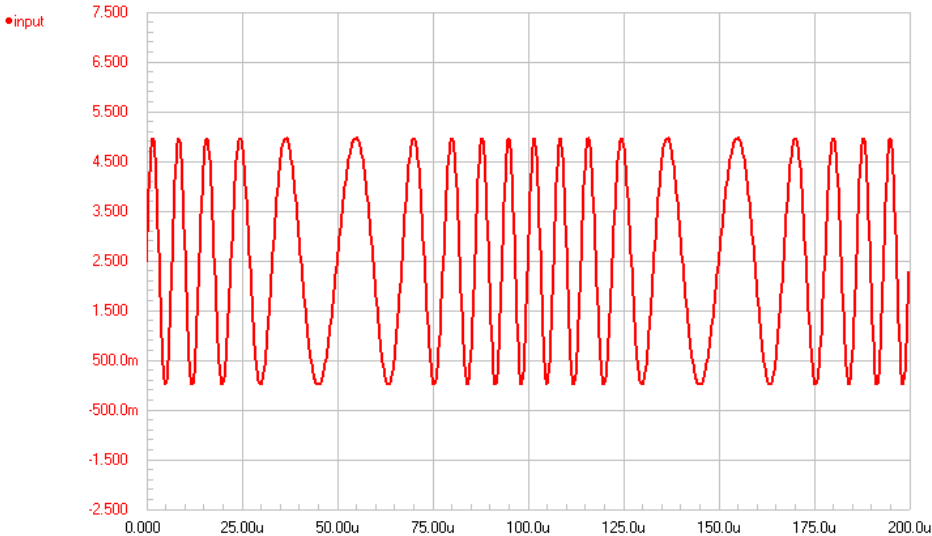
Standard RC discharging time constant.

Frequency Modulation Source

Use this source to create a single frequency, Frequency Modulated waveform. After you place the source you will need to specify the following attributes in the Part Fields of the Source component.



Attribute / Part Field	Setting
Designator	Set to the required source name (eg INPUT)
DC (V or A)	DC offset used in operating point analysis
AC (V or A)	Set this if you plan to do a small signal AC analysis (typically 1 volt)
AC Phase (degrees)	Phase of the small signal voltage or current
Offset (V or A)	DC offset of the signal generator (eg 2.5)
Amplitude (V or A)	Peak amplitude of the output voltage or current (eg 2.5)
Carrier (F)	Carrier frequency (eg 100k)
Modulation	Modulation index (eg 5)
Signal (F)	Modulation frequency (eg 10k)



Example of the waveform produced by an FM Source with the attributes set to the example values in the table above.

Definition of the FM Waveform

The waveform is described by the following formula where t = instance of time:

$$V(t) = V_O + V_A \sin(2\pi F_c t + MDI \sin(2\pi F_s t))$$

DC Offset (VO)

Adjust the DC bias of the signal generator with respect to the negative terminal (usually ground), measured in volts or amps.

Peak Amplitude (VA)

Maximum amplitude of the output swing, excluding the DC Offset, measured in volts or amps.

Carrier Frequency (F_c)

Frequency of the unmodulated output in hertz.

Modulation Index (MDI)

Value corresponding to a function of amplitude of the modulating signal indicating the level of modulation.

$$MDI = (\text{frequency deviation}) / F_s$$

Signal Frequency (F_s)

Frequency of the modulating signal in hertz.

Linear Dependent Sources

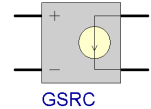
Standard SPICE linear dependent sources are supported. Each linear dependent source has two input terminals and two output terminals. The voltage/current at the output terminals is a linear function of the voltage/current at the input terminals, dependent on the gain/transconductance/transresistance of the source.

Place linear dependent sources from the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb Library Database. The image that is shown with each source includes the name of that source in the library.

Voltage Controlled Current Source

After you place the source define the following attributes:

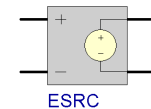
Attribute / Part Field	Setting
Designator	Set to the required source name (eg GSRC1)
Part Type	Transconductance of source in mhos



Voltage Controlled Voltage Source

After you place the source define the following attributes:

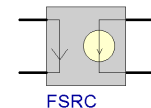
Attribute / Part Field	Setting
Designator	Set to the required source name (eg ESRC1)
Part Type	Voltage gain of source in volts



Current Controlled Current Source

After you place the source define the following attributes:

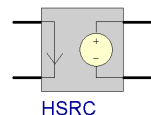
Attribute / Part Field	Setting
Designator	Set to the required source name (eg FSRC1)
Part Type	Current gain of source in amps



Current Controlled Voltage Source

After you place the source define the following attributes:

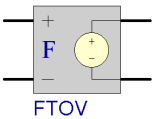
Attribute / Part Field	Setting
Designator	Set to the required source name (eg HSRC1)
Part Type	Transresistance of source in ohms



Frequency To Voltage Converter

After you place the source define the following attributes:

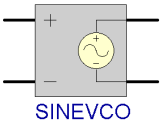
Attribute / Part Field	Setting
Designator	Set to the required source name (eg FTOV1)
VIL	Low level input threshold
VIH	High level input threshold
CYCLES	Cycles per volt output



Voltage Controlled Sine Wave Oscillator

After you place the source define the following attributes:

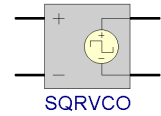
Attribute / Part Field	Setting
Designator	Set to the required source name (eg SINVCO1)
LOW	Peak output low value
HIGH	Peak output high value
C1	Input control voltage point 1
C2	Input control voltage point 2
C3	Input control voltage point 3
C4	Input control voltage point 4
C5	Input control voltage point 5
F1	Output frequency point 1
F2	Output frequency point 2
F3	Output frequency point 3
F4	Output frequency point 4
F5	Output frequency point 5



Voltage Controlled Square Wave Oscillator

After you place the source define the following attributes:

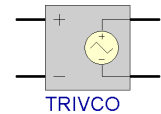
Attribute / Part Field	Setting
Designator	Set to the required source name (eg SQRVCO1)
LOW	Peak output low value (default 0)
HIGH	Peak output high value (default 5)
CYCLE	Duty cycle [0,1] (default 0.5)
RISE	Rise time [0,] (default 1u)
FALL	Fall time [0,] (default 1u)
C1	Input control voltage point 1 (default 0)
C2	Input control voltage point 2 (default 1)
C3	Input control voltage point 3 (default 2)
C4	Input control voltage point 4 (default 3)
C5	Input control voltage point 5 (default 4)
F1	Output frequency point 1 [0,] (default 0)
F2	Output frequency point 2 [0,] (default 1k)
F3	Output frequency point 3 [0,] (default 2k)
F4	Output frequency point 4 [0,] (default 3k)
F5	Output frequency point 5 [0,] (default 4k)



Voltage Controlled Triangle Wave Oscillator

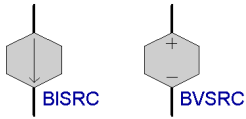
After you place the source define the following attributes:

Attribute / Part Field	Setting
Designator	Set to the required source name (eg TRIVCO1)
LOW	Peak output low value (default -1)
HIGH	Peak output high value (default 1)
CYCLE	Duty cycle [0,1] (default 0.5)
C1	Input control voltage point 1 (default 0)
C2	Input control voltage point 2 (default 1)
C3	Input control voltage point 3 (default 2)
C4	Input control voltage point 4 (default 3)
C5	Input control voltage point 5 (default 4)
F1	Output frequency point 1 [0,] (default 0)
F2	Output frequency point 2 [0,] (default 1k)
F3	Output frequency point 3 [0,] (default 2k)
F4	Output frequency point 4 [0,] (default 3k)
F5	Output frequency point 5 [0,] (default 4k)



Non-linear Dependent Sources

The simulator supports the standard SPICE non-linear dependent voltage and current sources. Place dependent sources from the \Program Files \Design Explorer 99 SE\Library\Sch\Sim \Simulation Symbols.lib library file.



After you place the source you will need to specify the following attributes;

Attribute	Setting
Designator	Set to the required source name
Part Type	Expression defining the source waveform (eg V(IN)^2)

Definition of the Waveform

The voltage or current waveform is described by:

$V = \text{expression}$ or $I = \text{expression}$

The value of V or I determines the voltage across or current through the device. The expression given for V or I may be any function of voltages and currents through voltage sources in the system.

To reference a node in your circuit in an equation you must first name the node in the schematic using a Net Label. You then use the name defined in the Net field of the Net Label's properties to reference the node using the following syntax:

$V(\text{net})$ references the voltage at node Net

$I(\text{net})$ references the current at the node Net

Examples

For example, if you have a node in your circuit labeled with a Net Label called IN, then the following would be valid entries in the Part Type field of the source:

$V(\text{IN})^3$
 $\text{COS}(V(\text{IN}))$

By default the node is referenced to the Spice Reference Net Name, which is specified in the Analog Options dialog (GND by default). You can include a different reference node directly in the equation using the following syntax:

$V(\text{netlabel1}, \text{netlabel2})$
eg $\text{LN}(\text{COS}(\text{LOG}(V(\text{NetLabel1}, \text{NetLabel2})^2))) - V(\text{NetLabel2}) \cdot V(\text{NetLabel1})$

Mixed-Signal Simulation

Supported Functions

ABS	LN	SQRT	LOG	EXP
SIN	ASIN	ASINH	SINH	
COS	ACOS	ACOSH	COSH	
TAN	ATAN	ATANH		

Standard operators

+ - * / ^ unary -If

Notes

If the argument of log, ln, or sqrt becomes less than zero, the absolute value of the argument is used. If a divisor becomes zero or the argument of log or ln becomes zero, an error will result. Other problems may occur when the argument for a function in a partial derivative enters a region where that function is undefined.

Components and Models

The simulator uses the standard SPICE syntax to describe the analog devices used in the circuit. In most cases you simply place a component from the library, set the value (such as the resistance), and simulate. Each component includes all the information required for simulation, including designator prefix information and pin mapping for multi-part components.

SPICE also supports many extended features that allow you to more accurately model component behavior. For example, you may need to simulate a resistor based on geometric information, or specify the operating temperature of a transistor.

This extra information is entered in the component Part Fields after the component has been placed on the schematic sheet. Refer to the *Device Descriptions* topic for a description of what information is entered in each part field, for each type of device.

You can also create your own simulation-ready schematic symbols. Refer to the topic *Creating your own Simulation-Ready Components* later in this chapter for more information on linking a schematic symbol to a model.

Component Symbol Libraries

The simulator includes a comprehensive set of simulation-ready schematic symbol libraries, which provide a broad coverage of the devices typically used in circuit design. The component symbols are grouped into libraries by function. Following is a complete list of the symbol libraries:

74XX.LIB	JFET.lib	Simulation Symbols.lib
7 Segment Displays.lib	Math.lib	Timer.lib
BJT.lib	Mesfet.lib	Transformer.lib
Buffer.lib	Mosfet.lib	Transmission lines.lib
CAmp.lib	OpAmp.lib	Tube.lib
CMOS.lib	Opto.lib	UJT.lib
Comparator.lib	Regulator.lib	Crystal.lib
Diode.lib	Relay.lib	Misc.lib
Fuse.lib	SCR.lib	Triac.lib
IGBT.lib	Switch.lib	

The symbol libraries are in the \Program Files\Design Explorer 99 SE \Library\Sch\Sim.ddb Library Database. Each component in these libraries includes a reference to a simulation model or subcircuit.

Finding Components in the Libraries

Use the powerful Find feature to quickly search the schematic symbol libraries for a particular component. You can search by Library Reference (the name of the component in the library), by component Description, or by both.

Select **Tools » Find Component** to pop up the Find Schematic Component dialog and search for a component.

Tips on Finding Components

- Searching by Library Reference is generally faster.
- Include the * wildcard before and after the search string (as shown in the image), as different manufacturers use different prefix and suffixes.
- Use multiple search strings in the description field to improve the chance of a match when searching by Description for more than one word. Enclose each word with the wildcard character and separate with a space (eg – *barrel* *shifter*).
- If your search produces no results check that the Path is correctly specified. Also, try searching for a component that you know is in a library to check that everything is set correctly.
- Use the Add to Library List button once you locate the correct component. When you press this button the selected library is added to the list of available libraries in the Schematic Sheet Editor.

Models and Subcircuits

The simulator's SPICE engine has built-in models for the following analog component types; resistors, capacitors, inductors, mutual inductors, independent voltage and current sources, dependent sources, lossless and lossy transmission lines, switches, uniform distributed RC lines, and the five most common semiconductor devices; diodes, BJTs, JFETs, MESFETs, and MOSFETs. The simulator also includes a large number of model files that define the behavior of specific instances of these devices. These model files normally have the file extension .MDL. All the simulation models and subcircuits are stored in a single database, \Program Files\Design Explorer 99 SE\Library\Sim\Simulation Models.ddb

The simulator also supports the description of more complex devices, such as op amps, regulators, timers, crystals, etc, using the hierarchical *subcircuit* syntax. A subcircuit consists of SPICE elements that are defined and referenced in a fashion similar to device models. There is no limit on the size or complexity of subcircuits, and subcircuits can call other subcircuits. Each subcircuit is defined in a subcircuit file (*.CKT), which are also often referred to as “models”.

Device Descriptions

◆ Apart from discrete components (such as resistors), the component Part Type is used as the reference to the simulation model. For non-discrete components (op amps, etc) you should only change the Part Type value if you want to reference a different model.

Resistors

As well as supporting the standard resistor, the simulator also supports semiconductor resistors. Semiconductor resistors allow you to model the resistance as a function of the geometry, and specify the temperature at which the device is to operate. Resistor symbols are in the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database.

Attribute / Part Field	Setting
Designator	Resistor designation
Part Type	Resistor value - if the value is specified it overrides the geometric definition
L #	Length of the resistor (meters)
W #	Width of the resistor (meters)
Temp #	Temperature at which this device is to operate (degrees Celsius, default is 27)

These options have defaults, only set them when required.

Capacitor

As well as supporting the standard capacitor, the simulator also supports semiconductor capacitors. Semiconductor capacitors allow you to model the capacitance as a function of the geometry, and the initial voltage across the capacitor. Capacitor symbols are in the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database.

Attribute / Part Field	Setting
Designator	Capacitor designation
Part Type	Capacitor value - if the value is specified it overrides the geometric definition
L #	Length of the capacitor (meters)
W #	Width of the capacitor (meters)
IC	Initial Conditions – time-zero voltage of capacitor. Only applies if the Use Initial Conditions option is enabled in the Setup Analyses dialog.

These options have defaults, only set them when required.

Inductor

Inductor symbols are in the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. The Inductor attributes include:

Attribute / Part Field	Setting
Designator	Inductor designation
Part Type	Inductor value
IC	Initial Conditions – time-zero current flowing through inductor. Only applies if the Use Initial Conditions option is enabled in the Setup Analyses dialog.

These options have defaults, only set them when required.

Coupled Inductors

Inductors can be coupled using the standard SPICE inductor coupling syntax. This is described in the topic, *Including extra SPICE Information in the Netlist*, in the chapter *Setting up and Running a Simulation*.

Diode

Standard diode and Zener diode symbols are in the Diode library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.lib database. The standard diode attributes include:

Attribute / Part Field	Setting
Designator	Diode designation
Part Type	Diode type
Area #	Area factor specifies the number of equivalent parallel devices of the specified model. This setting affects a number of parameters in the model.
Off / On #	Set to “Off” to set diode voltage to zero during operating point analysis. Can be useful as an aid in convergence.
IC	Initial Conditions – time-zero voltage across the diode. Only applies if the Use Initial Conditions option is enabled in the Setup dialog.
Temp (degrees Celsius) #	Temperature at which this device is to operate (default is 27 deg.)

These options have defaults, only set them when required.

Transistors

BJT

The model for the BJT is based on the integral-charge model of Gummel and Poon. However, if the Gummel-Poon parameters are not specified the model reduces to the simpler Ebers-Moll model. BJT symbols are in the BJT library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.lib database. The BJT attributes include:

Attribute / Part Field	Setting
Designator	Transistor designation
Part Type	Transistor type
Area #	Area factor specifies the number of equivalent parallel devices of the specified model. This setting affects a number of parameters in the model.
Off / On #	Set to “Off” to set terminal voltages to zero during operating point analysis. Can be useful as an aid in convergence.
IC	Initial Conditions – time-zero voltages for VBE and VCE (2 comma separated values required). Only applies if the Use Initial Conditions option is enabled in the Setup dialog.
Temp (degrees Celsius) #	Temperature at which this device is to operate (default is 27 deg.)

These options have defaults, only set them when required.

JFET

The JFET model is based on the FET model of Shichman and Hodges. JFET symbols are in the JFET library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. The JFET attributes include:

Attribute / Part Field	Setting
Designator	Transistor designation
Part Type	Transistor type
Area #	Area factor specifies the number of equivalent parallel devices of the model. This setting affects a number of parameters in the model.
Off / On #	Set to “Off” to set terminal voltages to zero during operating point analysis. Can be useful as an aid in convergence.
IC	Initial Conditions – time-zero voltages for VDS and VGS (2 comma separated values required). Only applies if the Use Initial Conditions option is enabled in the Setup dialog.
Temp (degrees Celsius) #	Temperature at which this device is to operate (default is 27 deg.)

Mixed-Signal Simulation

These options have defaults, only set them when required.

MOSFET

The simulator supports Shichman Hodges, BSIM 1, 2 and 3, and MOS 2, 3 and 6 models. MOSFET symbols are in the MOSFET library in the \Program Files \Design Explorer 99 SE\Library\Sch\Sim.ddb database. The MOSFET attributes include:

Attribute / Part Field	Setting
Designator	Transistor designation
Part Type	Transistor type
L #	Channel length (meters)
W #	Channel width (meters)
AD #	Drain area (meters ²)
AS #	Source area (meters ²)
PD #	Perimeter of drain junction (meters)
PS #	Perimeter of source junction (meters)
NRD #	Equivalent no. of drain diffusions
NRS #	Equivalent no. of source diffusions
OFF #	Set to "Off" to set terminal voltages to zero during operating point analysis. Can be useful as an aid in convergence.
IC	Initial Conditions – time-zero voltages for VDS, VGS and VBS (3 comma separated values required). Only applies if the Use Initial Conditions option is enabled in the Setup dialog.
Temp (degrees Celsius) #	Temperature at which this device is to operate (default is 27 deg.)

These options have defaults, only set them when required.

MESFET

The MESFET model is derived from the GaAs FET model of Statz. MESFET symbols are in the MESFET library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. The MESFET attributes include:

Attribute / Part Field	Setting
Designator	Transistor designation
Part Type	Transistor type
Area #	Area factor specifies the number of equivalent parallel devices of the model. This setting affects a number of parameters in the model.
Off / On #	Set to "Off" to set terminal voltages to zero during operating point analysis. Can be useful as an aid in convergence.
IC	Initial Conditions – time-zero voltages for VDS and VGS (2 comma separated values required). Only applies if the Use Initial Conditions option is enabled in the Setup dialog.

These options have defaults, only set them when required.

Voltage/Current Controlled Switches

The simulator includes a number of pre-defined switches, with different threshold voltages and resistance parameters. The schematic symbols for these switches are in the SWITCHES library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. Each component links to a model of the same name in the \Switches model folder in the \Program Files\Design Explorer 99 SE\Library\Sim\Simulation Models.ddb database.

Use these models as a reference when you need to define your own switch model. The following table describes each of the switches in the library.

Library Component	Description
CSW	Default current controlled switch
SW	Default voltage controlled switch
SW05	VT=500.0m
SWM10	VT=-0.01
SWP10	VT=0.01
STTL	VT=2.5 VH=0.1
TTL	VT=2 VH=1.2 ROFF=100E+6
TRIAC	VT=0.99 RON=0.1 ROFF=1E+7

Internally the SPICE engine supports the following switch parameters.

Voltage controlled switch

Name	Parameter and Units	Default Value
VT	Threshold voltage (volts)	0.0 [#]
VH	Hysteresis voltage (volts)	0.0
RON	On resistance (ohms)	1.0
ROFF	Off resistance (ohms)	1/GMIN*

Current controlled switch

Name	Parameter and Units	Default Value
IT	Threshold current (amps)	0.0
IH	Hysteresis current (amps)	0.0
RON	On resistance (ohms)	1.0
ROFF	Off resistance (ohms)	1/GMIN*

[#] The excitation signal must “pass through” this voltage to actuate the switch.

* GMIN is defined in the Analog Options dialog.

Notes on using Switches

The use of an ideal, highly non-linear element such as a switch can cause large discontinuities to occur in the circuit node voltages. The rapid changes associated with a switch changing state can cause numerical roundoff or tolerance problems, leading to timestep difficulties, or erroneous results. You can improve the situation by taking the following steps:

- Firstly, set the switch impedance just high or low enough to be negligible with respect to other circuit elements. Using a switch impedance that is close to “ideal” in all cases aggravates the problem of discontinuities.
- If you are modeling real devices such as MOSFETS, the on resistance should be adjusted to a realistic level, depending on the size of the device being modeled.
- If a wide range of ON to OFF resistance must be used in the switch (ROFF/RON > 1e+12), then the tolerance on errors allowed during transient analysis should be decreased by setting TRTOL to be less than the default value of 7.0 (in the Analog Options dialog). Try setting TRTOL to 1.
- When a switch is placed around a capacitor, then the CHGTOL option should also be reduced (try 1e-16).

These changes tell the simulator’s SPICE engine to be more careful around the switch points, so that errors are not made due to the rapid change in the circuit.

Fuses

There is a generic fuse symbol in the Simulation Symbols library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. Specify the fuse current rating in the Current Part Field. The fuse attributes include:

Attribute / Part Field	Setting
Designator	Fuse designation
Current #	Fuse current rating at rupture, eg 250m (amps)
Resistance #	Series resistance of fuse (default for all fuse models is 1m ohms)

These options have defaults, only set them when required.

Crystals

Crystal symbols are in the CRYSTALS library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. There are a number of pre-defined crystals, as well as a generic crystal component. All the crystals include a Frequency Part Field, however it is not necessary to enter a value in this field unless you need a different value from that specified in the Part Type. If you use the generic XTAL component (which has a default frequency of 1Mhz), enter the required frequency in the Frequency Part Field. The crystal attributes include:

Attribute / Part Field	Setting
Designator	Crystal designation
Freq #	Crystal fundamental frequency
RS #	Series resistance
C #	Crystal capacitance
Q #	Q of the equivalent circuit

These options have defaults, only set them when required.

◆ To check the default values of a crystal open the appropriate simulation model file. The crystal models are in the \Crystal folder in the \Program Files\Design Explorer 99 SE\Library\Sim\Simulation Models.ddb database. The model file has the same name as the Part Type of the crystal symbol. The model header describes each parameter, which have their defaults defined in the .SUBCKT line.

Relays

Relay symbols are in the RELAY library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. The attributes include:

Attribute / Part Field	Setting
Designator	Relay designation
Pullin #	Contact pullin voltage
Dropoff #	Contact dropoff voltage
Contact #	Contact resistance (ohms)
Resistance #	Coil resistance (ohms)
Inductance #	Coil inductance (henrys)

These options have defaults, only set them when required.

Transformers

Transformer symbols are in the TRANSFORMER library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. This symbols in this library link to two types of transformer models - transformer models which support magnetization and leakage inductance (TRANS and TRANSCT), and coupled inductor models which model the transformer as a pair of coupled inductors (KTRANSFORMER and KTRANSFORMERCT).

The TRANS and TRANSCT attributes include:

Attribute / Part Field	Setting
Designator	Transformer designation
Ratio #	Turns ratio = secondary turns / primary turns
RP #	Resistance of primary winding (default 0.1 ohms)
RS #	Resistance of secondary winding (default 0.1 ohms)
Leak #	Leakage inductance (default 1u henrys)
Mag #	Magnetizing inductance (default 1u henrys)

These options have defaults, only set them when required.

Transmission Lines

The simulator supports lossless transmission lines, constant-parameter distributed lossy transmission lines, and lumped RC lossy transmission lines. The transmission line symbols are in the TRANSLINE library in the \Program Files\Design Explorer 99 SE\Library\Sch\Sim.ddb database. The transmission line attributes include:

Lossless Transmission Line

Attribute / Part Field	Setting
Designator	Transmission line designation
ZO #	Characteristic impedance (ohms)
TD #	Transmission delay (refer to Notes below)
F	Frequency (refer to Notes below)
NL	Normalized electrical length of the transmission line with respect to the wavelength in the line, at the frequency F (refer to Notes below).
IC	Initial Conditions - the voltage and current at each of the transmission line ports. Only applies if the Use Initial Conditions option is enabled in the Setup dialog.

These options have defaults, only set them when required.

Specifying the Length of the Lossless Transmission Line

The length of the line must be expressed in one of two forms:

- The transmission delay is specified directly (eg, TD=10ns)
- Alternatively, a frequency F is specified, together with the normalized length, NL. If a frequency is specified but NL is omitted then 0.25 is assumed (that is, the frequency is assumed to be the quarter-wave frequency).

Lossy Transmission Line

This uses a two-port convolution model for single-conductor lossy transmission lines. Note that a lossy transmission line with zero loss may be more accurate than the lossless transmission line due to implementation details.

The Lossy transmission line model includes attributes for resistance, inductance, capacitance, and length. It is not possible to pass these parameters directly from the schematic component, however you can create and reference your own model file. To do this copy the file *ltra*.MDL. Edit this new model file and change the string immediately after the .MODEL statement to be the same as the new file name, then edit the attributes as required. To use this new model from the schematic enter the new model name in the component's Part Type field.

For example, from the existing model file *ltra*.MDL:

```
.MODEL LTRA LTRA(R=0.000 L=9.130n C=3.650p LEN=1.000)
```

You could create a new file, *ltra10*.MDL:

```
.MODEL LTRA10 LTRA(R=0.2 L=32n C=13p LEN=10.000)
```

Uniform Distributed RC Lines (Lossy)

Attribute / Part Field	Setting

Mixed-Signal Simulation

Designator	Transmission line designation
L #	Length of the RC line (meters)
N #	Number of lumped segments to use in modeling the RC line

These options have defaults, only set them when required.

Digital Components

The simulator includes a special descriptive language that allows digital devices to be simulated, using an extended version of the event-driven XSPICE. The digital devices are modeled using the *Digital SimCode™* language. Refer to the *Digital Simulation* chapter for more details on the SimCode language.

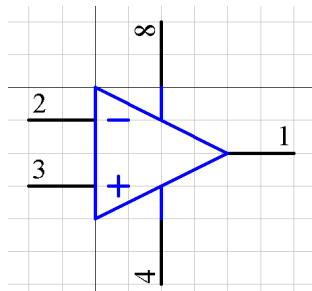
There are two libraries of digital components in the \Program Files\Design Explorer99\Library\Sch\Sim.ddb database, 74xx.lib and CMOS.lib.

The models for the components in these libraries include default values for all attributes, such as propagation delay, loading, etc. If required these defaults can be overridden for an individual component by filling in the appropriate Part Fields in the Component's Part dialog.

Attribute / Part Field	Setting
Propagation #	Device propagation delay. Specify <i>min</i> or <i>max</i> to use either the pre-defined minimum or maximum value.
Loading #	Input-loading characteristics. Specify <i>min</i> or <i>max</i> to use either the pre-defined minimum or maximum value.
Drive #	Output-drive characteristics. Specify <i>min</i> or <i>max</i> to use either the pre-defined minimum or maximum value.
Current #	Device current used to specify device power. Specify <i>min</i> or <i>max</i> to use either the pre-defined minimum or maximum value.
PWR value #	Power supply voltage – enter a voltage value to override the voltage on the device supply pin.
GND value #	Ground supply voltage – this voltage must also be specified if the PWR Part Field is specified.
VIL value #	Low-level input voltage
VIH value #	High-level input voltage
VOL value #	Low-level output voltage
VOH value #	High-level output voltage
WARN	Specify “on” to flag errors in; setup time, hold time, recovery time, pulse width, min max frequency violation, min max supply violation. Warnings are recorded in the .SIM file, errors in the .ERR file.

Creating your own Simulation-Ready Components

The first step to creating your own simulation components is to draw the schematic symbol in the Schematic Library Editor. Once the symbol is created you can enter the simulation information into the component Library Fields (select **Tools » Description** from the Library Editor menus). This information tells the simulator what type of component it is, where to find the model, how to map the pins in a multi-part component, as well as any specific SPICE information the component may require.



After creating the component symbol enter the simulation information in the Library Fields

Library Fields and Part Fields

The simulator reads SPICE and netlist information from the Library Text Fields (also referred to as the Read-Only Fields), and the component Part Fields. Not all components require the same information to be defined in the Text Fields, for example, discrete components do not have a model reference. Refer to the components in the simulator libraries (in the \Program Files\Design Explorer 99 SE \Library\Sch\Sim.ddb Library Database), for examples of what information each type of component requires. The rest of this section explains what information goes in each Library Text Field.

Library Field 1

type=<Device Type>(<SPICE Prefix>)

Specifies the type of device that is to be simulated, and the SPICE Prefix.

e.g. type=NPN(Q)

NPN denotes that the device is an NPN BJT. Q is the prefix required by the SPICE engine. Select a Type from the following list:

Mixed-Signal Simulation

Device Type and SPICE Prefix	Function
CAP(C)	Capacitor
CCCS(F)	Current Controlled Current Source
CCS(W)	Current Controlled Switch
CCVS(H)	Current Controlled Voltage Source
DIODE(D)	Diode
INDUCTOR(L)	Inductor
IPULSE(I)	Pulse Current Source
IPWL(I)	Piecewise Linear Current Source
ISFFM(I)	Single Frequency FM Current Source
ISIN(I)	Sinusoidal Current Source
ISRC(I)	DC Current Source
LTRA(O)	Lossy Transmission Line
MUTUALINDUCTANCE(K)	Mutual Inductor Coupling
NDMOS(M)	N-channel Depletion MOSFET
NEMOS(M)	N-channel Enhancement MOSFET
NJFET(J)	N-channel JFET
NMESFET(Z)	N-channel MESFET
NPN(Q)	NPN Bipolar Junction Transistor
PDMOS(M)	P-channel Depletion MOSFET
PEMOS(M)	P-channel Enhancement MOSFET
PJFET(J)	P-channel JFET
PMESFET(Z)	P-channel MESFET
PNP(Q)	PNP Bipolar Junction Transistor
POT(R)	Variable Resistor or Potentiometer
RES(R)	Resistor
SEMICAP(C)	Semiconductor Capacitor
SEMIRES(R)	Semiconductor Resistor
SIMCODE(A)	Digital SimCode Device
TRA(T)	Lossless Transmission Line
UDRC(U)	Uniformly Distributed RC Line (Lossy)
VCCS(G)	Voltage Controlled Current Source
VCSW(S)	Voltage Controlled Switch
VCVS(E)	Voltage Controlled Voltage Source
NLDS(B)	Non-linear Dependent Source
VPULSE(V)	Pulse Voltage Source
VPWL(V)	Piecewise-linear Voltage Source
VSFFM(V)	Single Frequency FM Voltage Source
VSIN(V)	Sinusoidal Voltage Source
VSRC(V)	DC Voltage Source
ZENER(A)	SimCode Zener Diode
SUBCKT(X)	Subcircuit

Library Field 2

model=<model name>

Specifies the name of the model to use when simulating the device. If the string model=<parttype> is entered here then the Part Type of the component is inserted as the model name.

Library Field 3

file={model_path}\...\<filename>.*

Indicates the location of the file in which the model indicated by the previous field can be found. If you are using the model=<parttype> syntax then this line would be; file={model_path}\...\<parttype>.* Typically the file extension will be .ckt, or .mdl. The contents of this model file are included at the end of the netlist file.

Library Field 4

Pins=1:[pin1<,pin2,pin3,...>]<2:[pin1<,pin2,pin3,...>]...>

Specifies the pins on each part of the symbol. The order here is not fixed, but it is convenient to enter the pin numbers in the order they are required by the simulation model. This makes it straightforward to specify the mapping numbers in Library Field 5. The order information is often detailed in the header of the model file.

e.g. for a quad 741 op amp this line would be:

```
pins=1:[3,2,4,11,1]2:[5,6,4,11,7]3:[10,9,4,11,8]4:[12,13,4,11,14]
```

Library Field 5

netlist=<SPICE Data>|<SPICE Data line 2>|...

This line defines how *this* component is netlisted in the SPICE netlist file. If you need to specify more than one line of netlist information use the vertical bar as a line delimiter.

As well as entering SPICE data directly in this line, you can also reference information in the other Library Fields, as well as the 16 Part Fields. The percent sign (%) indicates a reference to another field, the letter or number that follows the % indicates what field to use. The following options can be used on this line:

%D – Device designation

Inserts the device designator. If the first character of the designator does not match the SPICE prefix, then the prefix is automatically inserted at the beginning of the string in the netlist.

%1, %2, %3, .. %n

Device pins to be added to the netlist, in the order required by the SPICE model. The numbers are not used directly, each is used as an index to the component pin number specified in Library Field 4 (for this part of the component).

For example, a quad 741 op amp (MC4741), with a designator of U1C (part 3 of U1), has the following entries for Library Fields 4 and 5:

Mixed-Signal Simulation

```
pins=1:[3,2,4,11,1]2:[5,6,4,11,7]3:[10,9,4,11,8]4:[12,13,4,11,14]
netlist=%D %1 %2 %3 %4 %5 %M
```

During netlisting, this netlist= line is interpreted in the following way;

Entry	Action	Result in Netlist
%D	Insert designator, adding SPICE prefix if required	XU1C
%1	Look up the first pin for the 3 rd part (remember it is U1C) in the pins= line, and add the net on this pin to the netlist	NetOnPin10
%2	Look up the second pin for the 3 rd part, and add the net on this pin to the netlist	NetOnPin9
%3	Look up the third pin for the 3 rd part, and add the net on this pin to the netlist	NetOnPin4
%4	Look up the fourth pin for the 3 rd part, and add the net on this pin to the netlist	NetOnPin11
%5	Look up the fifth pin for the 3 rd part, and add the net on this pin to the netlist	NetOnPin8
%M	Inserts the model name specified in Library Field 2	MC4741

The result in the netlist file is:

```
XU1C NetOnPin10 NetOnPin9 NetOnPin4 NetOnPin11 NetOnPin8 MC4741
```

Digital devices use the same approach, the numbers listed in Library Text Field 5 are used as an index into the actual pin numbers specified in Library Text Field 4. As with analog symbols the order of the pins in Library Field 5 must match the order defined in the SimCode model. The difference with SimCode models is that the input and output pins are listed separately. Refer to the *Example - Creating a 74LS74 Symbol and SimCode Model* in the *Digital Simulation* chapter for more details.

%F1 to %F16 – Text Fields

%F1 through %F16 refer to the schematic Part Fields 1-16 respectively. Inserts the value of the specified Part Field into the netlist.

%IF() – Conditional Netlist Entry

Inserts the contents of the braces if the fields referenced within contain data. This is non-recursive (nested %Ifs not allowed) and is limited to Part Field and text values (i.e. %Fx or %Px). For example:

```
%IF(AC %F2 %F3) would add; AC 1 0
```

```
%IF(PARAMS: %P6) would add; PARAMS: PartField6Name = PartField6Value
```

%M – Model Name

Inserts the model name specified in Library Field 2.

%P or %P1 through %P16 – Device Parameters

Inserts the part field name, as well as the contents of that part field. %P on its own results in all Part Fields that contain a value being included.

The format that the parameters are inserted is:

<TextFieldName1=TextField1> ... <TextFieldName16=TextField16>

If the Part Field does not contain a value then nothing is inserted.

%R – Library Reference

Inserts the contents of the Library Reference Field.

%T – Part Type

Inserts the contents of the Part Type Field into the netlist. No checking of any kind is performed.

%V – Value

Inserts the contents of the Part Type Field into the netlist. Requires that the value is a number. The number can be an integer, a floating point number, a floating point number followed by an integer exponent, or an integer or floating point number followed by one of the standard scale factors (multipliers). Refer to the *Component and Simulation Values* topic in the *Getting Started with Simulation* chapter for a complete list of allowed scale factors.

Note: If a multiplier is used it must immediately follow the number, spaces are not permitted between the multiplier and the number. Letters that are not multipliers are ignored.

Library Field 8 (Optional)

e.g. defaults=F1:0,F2:15,F3:6,...

This field is used to specify any default parameters that are required in the component Part Fields. This instructs the simulator to read to label from the specified Part Field, then add the default value entered here.

For example, the Lossless Transmission Line (library reference LLTRA) includes the following information in Text Field 8:

defaults=F1:50,F2:10ns

Netlisting this component (between nets IN, O and OUT, O) produces the following:

TLLTR1 IN 0 OUT 0 Z0=50 TD=10ns

Part Fields 1-16

Part Fields 1-16 are used for parameters that can be specified once a component has been placed on the schematic. Use the %F1-16, %P, %P1-16 or %PARAMS syntax in Library Field 5 to include them in the netlist (as described earlier in this topic).

Working with Circuits that will not Simulate

When a circuit will not simulate you must identify if the problem is in the circuit, or the process of simulation. Read the following topics and work through the suggested points, trying one at a time.

When The Netlist can not be Created

When you select **Simulate » Run** from the menus the first thing that happens is the circuit is analyzed and a netlist is generated. This netlist is then passed to the SPICE engine, which simulates the circuit and generates the results.

If errors are detected during netlisting an Error dialog will pop up. The errors are written to *DesignName*.ERR, which is automatically opened if you click Yes in the error message. Likely causes of netlisting errors include:

- The component listed in the error file is not a simulation-ready component. To check if it is double-click on the component, and confirm that the Read Only fields contain the normal simulation reference information. If you are not sure what this looks like refer to a component on one of the example schematics.
- The simulation model file that the component references is not in the location specified in Read Only Field 3. This could happen if the models are not installed, or they have been moved from their original install location. The default location for the Simulation Models.ddb database is \Program Files\Design Explorer 99 SE\Library\Sim\. Check in this folder for the simulation models database, there should be 27 folders of models and subcircuits inside the database. If the database is not present, or models are missing, you can copy the Simulation Models.ddb database from the product CD. Remember to disable the Read-Only attribute whenever you copy a database from a CD.
- The path to the simulation model, referred to as {model_path}, does not match the location of the model. This could happen if the models are moved to a different location on the hard drive. The model path is stored in *Advsim99.INI*.

Understanding SPICE Convergence

The most common simulation problem is failure to converge. What exactly is meant by this term, “convergence”? Like most simulators, the SPICE3 engine in Protel’s simulator uses an iterative process of repeatedly solving the equations that represent your circuit, to find the quiescent circuit voltages and currents. If it fails to find these voltages and currents (fails to converge), then it will not be able to perform an analysis of the circuit.

SPICE3 uses simultaneous linear equations, expressed in matrix form, to determine the operating point (DC voltages and currents) of a circuit at each step of the simulation. The circuit is reduced to an array of conductances which are placed in the matrix to form the equations ($G * V = I$). When a circuit includes nonlinear elements, SPICE uses multiple iterations of the linear equations to account for the nonlinearities. SPICE makes an initial guess at the node voltages then calculates the branch currents based on the conductances in the circuit. SPICE then uses the branch currents to recalculate the node voltages, and the cycle is repeated. This cycle continues until all of the node voltages and branch currents fall within specified tolerances (converge).

◆ Use the Analog Options dialog box to specify the tolerances and iteration limits for the various analyses.

However, if the voltages or currents do not converge within a specified number of iterations, SPICE produces error messages (such as “singular matrix”, “Gmin stepping failed”, “source stepping failed” or “iteration limit reached”) and aborts the simulation. SPICE uses the results of each simulation step as the initial guesses for the next

step. If you are performing a Transient analysis (that is, time is being stepped) and SPICE cannot converge on a solution using the specified timestep, the timestep is automatically reduced, and the cycle is repeated. If the timestep is reduced too far, SPICE displays a “Timestep too small” message and aborts the simulation.

Strategies for Solving Convergence Failures

Use the following techniques to solve convergence problems. When you have a convergence problem, first turn off all the analyses except the Operating Point. Begin with step 1, then consider the recommendations to solve the error.

1. Check the *DesignName.ERR* error file. This will describe any errors that have been detected.
2. Check the circuit topology and connectivity. This includes;
 - Make sure the circuit is wired correctly. Dangling nodes and stray parts are not allowed.
 - Every circuit must have a ground node, and every node in the circuit must have a DC path to this ground. Components that can isolate a node include transformers and capacitors. Voltage sources are considered a DC short circuit, current sources are considered a DC open circuit.
 - Don't confuse zeros with the letter O.
 - Use proper SPICE multipliers (MEG instead of M for 1E+6). Multipliers are not case sensitive.
 - Spaces between values and multipliers are not allowed. For example it should be 1.0uF, not 1.0 uF.
 - Make sure all devices and sources are set to their proper values.
 - Make sure the gain of any dependent source is correct.

- Temporarily eliminate series capacitors or current sources.
 - Temporarily eliminate parallel inductors or voltage sources.
3. Increase the number of iterations in the Analog Options dialog box (ITL1) to 300. This will allow the Operating Point analysis to go through more iterations before giving up.
 4. Add .NS (Nodeset) devices to define the node voltages. If the initial guess of a node voltage is way off the .NS device can be used to pre-define a starting voltage that is used for a preliminary pass of the operating point analysis. Refer to the topic *Specifying Initial Circuit Conditions* in the *Setting up and Running a Simulation* chapter for details on using the Nodeset device.
 5. If the Nodeset device does not assist in convergence try defining the initial conditions by placing .IC devices. In this case the node voltages are held at the specified values during the operating point analysis, then released during the transient analysis.
 6. Enable the Use Initial Conditions option in the Analyses Setup dialog. This option works in conjunction with the .IC devices (or the IC parameter of the components). The difference with this option is the operating point analysis is not performed, the specified voltages are used as the initial conditions for the transient analysis. Refer to the topic *Specifying Initial Circuit Conditions* in the *Setting up and Running a Simulation* chapter for details on defining Initial Conditions.
 7. Specify the series resistance parameters of your models and increase the GMIN option by a factor of 10. Specify the initial condition of semiconductor devices, especially diodes, as OFF.

Solving DC Analysis Failures

1. Check the circuit topology and connectivity. See the common mistakes described in the earlier topic, *Strategies for Solving Analysis Failures*.
2. Increase ITL2 to 200 in the Analog Options dialog. This will allow the DC analysis to go through more iterations for each step before giving up.
3. Make the steps in the DC sweep larger or smaller. If discontinuities exist in a device model (perhaps between the linear and saturation regions of the model), increasing the step size may allow the simulation to step over the discontinuity. Making the steps smaller will allow the simulation to resolve rapid voltage-transition discontinuities.
4. Do not use DC analysis. Some problems (such as hysteresis) cannot be resolved by DC analysis. In such cases, it is more effective to use Transient analysis, by ramping the appropriate power sources.

Solving Transient Analysis Failures

2. Check the circuit topology and connectivity. See the common mistakes described in the earlier topic, *Strategies for Solving Analysis Failures*.
3. Set RELTOL to 0.01 in the Analog Options dialog box. By increasing the tolerance from 0.001 (0.1% accuracy), fewer iterations will be required to converge on a solution and the simulation will complete much more quickly.
4. Increase ITL4 to 100 in the Analog Options dialog box. This will allow the Transient analysis to go through more iterations for each timestep before giving up. Raising this value may help to eliminate “timestep too small” errors improving both convergence and simulation speed.
5. Reduce the accuracy of ABSTOL/VNTOL if current/voltage levels allow. Your particular circuit may not require resolutions down to 1uV or 1pA. You should allow at least an order of magnitude below the lowest expected voltage or current levels of your circuit.
6. Realistically model your circuit. Add realistic parasitics, especially stray/junction capacitance. Use RC snubbers around diodes. Replace device models with subcircuits, especially for RF and power devices.
7. Increase the rise/fall times of the Pulse Generators. Even the best pulse generators cannot switch instantaneously.
8. Change the integration method to Gear. Gear integration requires a longer simulation time, but is generally more stable than trapezoidal. Gear integration may be particularly useful with circuits that oscillate or have feedback paths.

Warning and Error Messages

Sometimes the simulator will display a message reporting errors or warnings. Warnings are saved in *DesignName.SIM*, and errors are saved in *DesignName.ERR*. The following distinguishes warning messages from error messages.

Warning Messages

Warning messages are not fatal to the simulation. They generally provide information about changes that SPICE had to make to the circuit in order to complete the simulation. These include invalid or missing parameters, and so on.

SimCode warnings may include information such as timing violations (tsetup, thold, trec, tw, etc) or significant drops in power supply voltage on digital components.

Note: Valid simulation results are normally generated even if warnings are reported.

Error Messages

Error messages provide information about problems that SPICE could not resolve and were fatal to the simulation. Error messages indicate that simulation results could not be generated, so they must be corrected before you will be able to analyze the circuit.

SPICE Variables and Analog Options

The simulator provides access to the internal settings available in the SPICE engine. These allow you to control certain aspects of a simulation, such as iteration limits, temperature, propagation delays, adding node shunt resistances, and others.

These are advanced user settings, you should only change these when you have been advised to (the chapter, *Working with Circuits that will not Simulate*, suggests changing these under certain conditions), or if you understand exactly what you are doing.

Options

Option Value: Enter

Option	Description	Value	Def
ABSTOL	Absolute current error tolerance [0,]	1.000p	*
ACCT	Display accounting information [Y/N]	NO	*
ADCSTEP	Analog change for digital event [0,]	10.00m	*
AUTOPARTIAL	Use AP computation [Y/N]	NO	*
BADMOS3	Use MOS3 with kappa discontinuity [Y/N]	NO	*
BOOLH	Boolean output high value [0,]	4.500	*
BOOLL	Boolean output low value [0,]	0.000	*
BOOLT	Boolean input threshold value [0,]	1.500	*
BYPASS	Use nonlinear model evaluation [0=OFF,]	1	*

Digital Power Supply values referenced to GND node

VCC: VDD:

Integration method:

Spice Reference Net Name:

OK Cancel Help

Select **Simulate » Analyses Setup** to pop up the Analyses Setup dialog, then press the Advanced button to display the Analog Options – SPICE Variables dialog. The options in this dialog include:

Integration Method

Choose which numerical integration method you want to use. The *Trapezoidal* method is relatively fast and accurate, but tends to oscillate under certain conditions. The *Gear* methods requires longer simulation times, but tends to be more stable. Using a higher gear order theoretically leads to more accurate results, but increases simulation time. The default is Trapezoidal.

Digital Power Supply and the Reference Ground

All the digital components supplied with the simulator have hidden power pins; VCC and GND for the 74xx devices, and VDD and GND for the 4000 series devices. These hidden power pins are automatically connected during netlisting, and assigned the voltages specified in the VCC and VDD fields in the Analog Options dialog. These voltages can be changed if required.

To power your circuit from nets other than VCC (or VDD for CMOS) you must include source components to create the appropriate voltages, un-hide the power pins for each component, and wire the power pins to these nets.

To run a transient simulation which references a net other than ground, enter the net name in the Spice Reference Net Name field.

SPICE Options

Following is a list of the SPICE options and their effect on the simulation. To change the value of a SPICE variable first select the variable, type the new value into the Option Value field, then press the Enter button.

◆ To return an option to its default value enter an asterisk in the Option Value edit field.

Option	What it Does
ABSTOL	Sets the absolute current error tolerance of the program. Set $ABSTOL = RELTOL * (\text{lowest current magnitude in the circuit})$. Default=1 picoamp.
CHGTOL	Provides a lower limit on capacitor charge or inductor flux; used in the LTE timestep control algorithm. Default=1.0e-14 coulombs.
DEFAD	Sets the MOS drain diffusion area. Default=0.0 meters ² .
DEFAS	Sets the MOS source diffusion area. Default=0.0 meters ² .
DEFL	Sets the MOS channel length. Default=100.0 micrometer.
DEFW	Sets the MOS channel width. Default=100.0 micrometer.
GMIN	Sets the minimum conductance (maximum resistance) of any device in the circuit. It also sets the value of the conductance that is placed in parallel with each pn junction in the circuit. Default=1.0e-12 mhos. Note: Raising this value may help with simulation convergence in many circuits, but might decrease accuracy.
ITL1	Sets the Operating Point Analysis iteration limit. Default=100 iterations. Note: This may need to be raised as high as 500 for many circuits.
ITL2	Sets the DC Analysis iteration limit. Default=50 iterations. Note: This may need to be raised as high as 200 for some circuits.
ITL3	Sets the lower Transient Analysis iteration limit. Default=4 iterations. Note: This is not implemented in SPICE3. It is provided for compatibility in

Mixed-Signal Simulation

	creating SPICE2 netlists.
ITL4	Sets the Transient Analysis timepoint iteration limit. Default=10 iterations. Note: Raising this value to 100 or more may help to eliminate “timestep too small” errors improving both convergence and simulation speed.
ITL5	Sets the Transient Analysis total iteration limit. Default=5000 iterations. Note: This is not implemented in SPICE3. It is provided for compatibility in creating SPICE2 netlists.
PIVREL	Sets the relative ratio between the largest column entry in the matrix and an acceptable pivot value. The value must be between 0 and 1. Default=1.0e-3. In the numerical pivoting algorithm the allowed minimum pivot is determined by $EPSREL = AMAX1(PIVREL * MAXVAL, PIVTOL)$ where MAXVAL is the maximum element in the column where a pivot is sought (partial pivoting).
PIVTOL	Sets the absolute minimum value for a matrix entry to be accepted as a pivot. Default=1.0e-13.
RELTOL	Sets the relative error tolerance of the program. The value must be between 0 and 1. Default is 0.001 (0.1%). <i>Larger values mean faster simulation time, but less accuracy.</i>
TEMP	Sets the actual operating temperature of the circuit. Any deviation from TNOM will produce a change in the simulation results. Default=27°C. Note: TEMP can be overridden by a temperature specification on any temperature dependent instance.
TNOM	Sets the nominal temperature for which device models are created. Default=27°C. Note: TNOM can be overridden by a specification on any temperature dependent device model.
TRTOL	Used in the LTE timestep control algorithm. This is an estimate of the factor by which SPICE overestimates the actual truncation error. Default=7.0.
VNTOL	Sets the absolute voltage tolerance of the program. Set $VNTOL = RELTOL * (\text{lowest voltage magnitude in the circuit})$. Default=1 microvolt.
BOOLL	Sets the low output level of a boolean expression. Default=0.0V.
BOOLH	Sets the high output level of a boolean expression. Default=4.5V.
BOOLT	Sets the input threshold level of a boolean expression. Default=1.5V.
BADMOS3	Uses the older version of the MOS3 model with the “kappa” discontinuity. Default=NO (don’t use the older version).
KEEPOPINFO	Retains the operating point information when an AC Analysis is run. Note: This is particularly useful if the circuit is large and you do not want to run a redundant Operating Point Analysis. Default=NO (run OP each time).
TRYTOCOMPACT	Applicable to the LTRA model. When specified, the simulator tries to condense LTRA transmission line’s past history of input voltages and currents. Default=NO (don’t compact).
NOOPITER	Skip directly to GMIN stepping algorithm. Default=NO (don’t skip).

GMINSTEP	Sets the number of steps in the GMIN stepping algorithm. When set to 0, GMIN stepping is disabled, making source stepping the simulator's default DC (operating point) convergence algorithm. Default=10 steps.
SRCSTEP	Sets the number of steps in the source stepping algorithm for DC (operating point) convergence. Default=10 steps.
ACCT	Causes accounting and run-time statistics to be displayed. Default=NO (no display).
LIST	Displays a comprehensive list of all elements in the circuit with connectivity and values. Default=NO (no list).
OPTS	Displays a list of all standard SPICE3 Option parameter settings. Default=NO (no list).
BYPASS	Enables the device bypass scheme for nonlinear model evaluation. Default=1 (on).
MINBREAK	Sets the minimum time between breakpoints. Default=0 seconds (sets the time automatically).
MAXOPALTER	Sets the maximum number of analog/event alternations for DC (operating point) convergence. Default=0.
MAXEVTITER	Sets the maximum number of event iterations for DC (operating point) convergence. Default=0.
NOOPALTER	Enables DC (operating point) alternations. Default=NO.
RAMPTIME	Controls turn-on time of independent sources and capacitor and inductor initial conditions from zero to their final value during the time period specified. Default=0.0 seconds.
CONVLIMIT	Disables convergence algorithm used in some built-in component models. Default=NO.
CONVSTEP	Sets the limit of the relative step size in solving for the DC operating point convergence for code model inputs. Default=0.25.
CONVABSSTEP	Sets the limit of the absolute step size in solving for the DC operating point convergence for code model inputs. Default=0.1.
AUTOPARTIAL	Enables the automatic computation of partial derivatives for XSPICE code modules. Default=NO.
PROPMNS	Sets scale factor used to determine minimum propagation delay when actual value is not specified in SimCode model. Default=0.5 (50% of typical propagation delay).
PROPMXS	Sets scale factor used to determine maximum propagation delay when actual value is not specified in SimCode model. Default=1.5 (150% of typical propagation delay).
TRANMNS	Sets scale factor used to determine minimum transition time when actual value is not specified in SimCode model. Default=0.5 (50% of typical transition time).

Mixed-Signal Simulation

TRANMXS	Sets scale factor used to determine maximum transition time when actual value is not specified in SimCode model. Default=1.5 (150% of typical transition time).
LOADMNS	Sets scale factor used to determine minimum input loading (maximum input resistance) when actual value is not specified in SimCode model. Default=1.5 (150% of typical input resistance).
LOADMXS	Sets scale factor used to determine maximum input loading (minimum input resistance) when actual value is not specified in SimCode model. Default=0.5 (50% of typical input resistance).
DRIVEMNS	Sets scale factor used to determine minimum output drive capacity (maximum output resistance) when actual value is not specified in SimCode model. Default=1.5 (150% of typical output resistance).
DRIVEMXS	Sets scale factor used to determine maximum output drive capacity (minimum output resistance) when actual value is not specified in SimCode model. Default=0.5 (50% of typical output resistance).
CURRENTMNS	Sets scale factor used to determine minimum supply current (maximum internal resistance) when actual value is not specified in SimCode model. Default=1.5 (150% of typical internal resistance).
CURRENTMXS	Scale factor used to determine maximum supply current (minimum internal resistance) when actual value is not specified in SimCode model. Default=0.5 (50% of typical internal resistance).
TPMNTYMX	Temporary global override for propagation delay index on SimCode devices (0=default, 1=min, 2=typ, 3=max). Default=0.
TTMNTYMX	Temporary global override for transition time index on SimCode devices (0=default, 1=min, 2=typ, 3=max). Default=0.
LDMNTYMX	Temporary global override for input loading index on SimCode devices (0=default, 1=min, 2=typ, 3=max). Default=0.
DRVMNTYMX	Temporary global override for output drive capacity index on SimCode devices (0=default, 1=min, 2=typ, 3=max). Default=0.
IMNTYMX	Temporary global override for supply current index on SimCode devices (0=default, 1=min, 2=typ, 3=max). Default=0.
SIMWARN	A nonzero value indicates that SimCode warning messages can be displayed during run time. SimCode warnings may include information concerning timing violations (tsetup, thold, trec, tw, etc) or indicate supply voltage dropping below device specifications. Default=0.
RSHUNT	Value in ohms of resistors added between each circuit node and ground, helping to eliminate problems such as “singular matrix” errors. In general, the value of RSHUNT should be set to a very high resistance (1e+12). Default=0 (no shunt resistors).
ADCSTEP	The minimum step size required to register an event on the input of the internal A/D converters. Default=0.01 volts.

Suggested SPICE Reading

Newton, A.R., Pederson, D.O., Sangiovanni-Vincentelli, A., *SPICE3 Version 3f4 User's Manual*

The basic SPICE reference, written by the developers of SPICE, at the University of California, Berkeley.

Vladimirescu, A., *The SPICE Book*, John Wiley & Sons, Inc., N.Y., 1994, ISBN: 0-471-60926-9, Library: TK454.V58 1994, 621.319'2'028553-dc20

Written as a tutorial and reference for electrical engineering students and professionals just starting to use the SPICE program to analyze and design circuits. This book explains how to use the SPICE program and describes the differences and similarities between the most popular commercial versions of it, including SPICE3, the latest version from Berkeley, which is used by the Protel Circuit Simulator.

Kielkowski, R., *Inside SPICE*, McGraw-Hill, Inc., N.Y., 1994, ISBN: 0-070911525, Library: TK 454.K48 1994, 621.319'2'011353-dc20

Written as a tutorial and reference for electrical engineering students and professionals who are familiar with the SPICE program. This book goes beyond the basics and covers the internal operation of the SPICE program to give the reader a solid understanding of how SPICE works. It provides step-by-step coverage of how to overcome nonconvergence, numerical integration instabilities and timestep control errors. It also shows how to make simulations run faster and more efficiently by setting the .OPTION parameters.

Simulating Digital Designs

Protel's Circuit Simulator is a true mixed-signal simulator, which means it can analyze circuits that include both analog and digital devices. Creating a mixed analog and digital design is as simple as placing the components and wiring them up. Each analog component from the simulation-ready libraries references a SPICE model, and each digital component references a *SimCode* model. Once the design is complete it is simulated in the normal manner. There are two libraries of digital components in the \Program Files\Design Explorer99\Library\Sch\Sim.ddb Library Database, 74xx.lib and CMOS.lib.

Modeling Digital Components

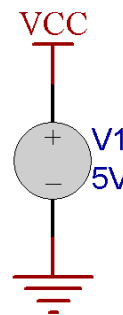
Due to the complexity of digital devices it is generally not practical to simulate them using standard, non-event-driven, SPICE instructions. For this reason, the simulator includes a special descriptive language that allows digital devices to be simulated using an extended version of the event-driven XSPICE. The digital devices included in the simulation library are modeled using the *Digital SimCode™* language.

Digital SimCode is a proprietary language – devices created with it are not compatible with other simulators, nor are digital components created for other simulators compatible with Protel's Circuit Simulator.

Digital Power and Ground

Digital devices include hidden power and ground power pins in each schematic symbol (VCC for 74xx devices, VDD for CMOS devices, and GND), and these pins are automatically connected during netlisting. The simulator uses these net names by default, so for a digital-only design it is not necessary to include sources to power these components. If the design includes any analog components that connect to the VCC (or VDD) power rail (perhaps a pullup), you must include an appropriate VCC or VDD source.

The value of VCC and VDD that are used during simulation, and the name of the reference net (the default is GND), are specified in the Analog Options dialog. Press the Advanced button in the Analyses Setup dialog to display this dialog.



powering the digital components in a mixed-mode design

Creating New SimCode Devices

SimCode is a “C like” description language. You use it to define the characteristics and behavior of the device you are modeling. It includes functions to define parameters such as propagation delays, load characteristics, strengths, and so on. The device behavior is defined using truth tables, math functions and conditional control statements, such as IF..THEN. Refer to the *SimCode Language Definition* topic later in this chapter for a summary of all the language items, and the *SimCode Language Syntax* topic for complete details of each language item.

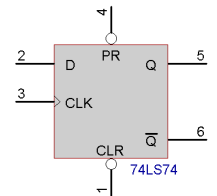
Use the following example to guide you through the process of creating your own SimCode model.

Example - Creating a 74LS74 Symbol and SimCode Model

Following is an example of the steps taken to create a new simulation-ready 74LS74 dual positive-edge triggered D flip-flop.

Step 1 – Create the schematic symbol

The first step is to create a schematic symbol for the device in the Schematic Library Editor. The 74LS74 has been drawn as a 2-part component. VCC (pin 14) and GND (pin 7) are defined as hidden pins.



First create the symbol

Refer to the *Schematic Components and Libraries* chapter in the *Schematic Capture* section for details on creating a schematic symbol. Often is easier to copy an existing symbol, and then modify it as required.

Step 2 – Define the simulation linking information

The next step is to add the information to the Library Text Fields that links the symbol to the SimCode model.

Refer to the topic *Creating your own Simulation-Ready Components* in the *Components and Models* chapter for an overall description of how each Library Text Field is used. Enter the component type, the model name, and the model file location in Text Fields 1, 2 and 3.

You are now ready to define the pin specifications (Text Field 4) and netlist information, which includes the pin mapping information (Text Field 5).

Text Field 4 - The symbol has been drawn as a 2-part component, so the pins= entry will have 2 sections, one for each part:

```
pins=1:[...]2:[...]
```

The order of the pins in Text Field 4 is not fixed, so for convenience they have been listed from left-to-right, top-to-bottom, for each part in the component.

```
pins=1:[2,3,4,1,5,6,7,14]2:[12,11,10,13,9,8]
```

Text Field 5 – The first entry on this line is the designator reference, %D. The last entry on this line is %M, the model reference.

Between the Designator and Model references is the pin mapping information, which is also known as the node list. The syntax for digital components is different than analog components; for digital components there are two sections, one for the input nodes, the second for the output nodes. Each section is delimited by square brackets.

The nodes must be listed in the *same order* as the pins in the *INPUTS* and *OUTPUTS* statements in the SimCode. If you refer to the example source file for the 74LS74, the inputs and outputs are specified as follows:

```
INPUTS VCC, GND, PRE, DATA, CLK, CLR;
```

```
OUTPUTS VCC_LD, PRE_LD, DATA_LD, CLK_LD, CLR_LD, QN, Q;
```

(The syntax of these statements is described later in the example.)

The easiest way to map the pins is to draw up a table. List the pins in the order they are required by the SimCode model, then write the position of this pin in the pin list (Text Field 4) next to it. These are shown below;

Input Pin	Position of this pin in the pin list
VCC	8
GND	7
PRE	3
DATA	1
CLK	2
CLR	4

Output Pin	Position of this pin in the pin list
VCC_LD	8
PRE_LD	3
DATA_LD	1
CLK_LD	2
CLR_LD	4
QN	6
Q	5

From this the node list can be written, adding an “i” after each input pin and an “o” after each output pin. Text Field 5 becomes:

```
netlist=%D [%8i %7i %3 %1 %2 %4][%8o %3o %1o %2o %4o %6o %5o] %M
```

Step 3 – Create the intermediate model linking file

The next step is to create a .MDL model linking file, that maps from the schematic symbol to the ASCII SimCode model file. Once the model has been compiled you change this mapping file to reference the compiled SimCode model file.

In a text editor create a new file, called 74LS74.MDL. Save it in the location specified in the model file location, in Text Field 3 of the symbol.

The MDL file includes one .MODEL line, and as many comment lines (starting with a “*” symbol) as required.

The syntax of the .MODEL line is:

```
.MODEL 74LS74 xsimcode(file="{MODEL_PATH}74LS74.TXT" func=ls74
```

.MODEL	Declares the model statement.
74LS74	Model name (as specified in Text Field 2).
xsimcode	Model type for a digital SimCode model.
file=	Points to file containing the device’s digital SimCode. {MODEL_PATH} is a shortcut to the Models directory specified in the <i>Advsim</i> .INI file.
func=	Specifies the device’s digital SimCode function.
data=	Contains ASCII data for the <i>READ_DATA</i> function (optional).
{mntymx}	Passes the device’s Digital Model Parameters into SimCode (this must appear exactly as shown).

Step 4 – Write the source code for SimCode model file

The last step is to create the digital SimCode model for the device. You can do this in any ASCII text editor. The file can be given any name and extension, as long as it matches the file name listed in the “file=” parameter in the .MDL file. Typically it is named the same as the device, 74LS74.TXT for this example. Multiple digital SimCode device models can be placed in the same file, each is referenced by the “func=” parameter.

There is an example of the SimCode for a 74LS74 later in this topic. This SimCode can be copied and pasted directly from the On-line help file if you do not want to type it in.

Test the new device by creating a simple circuit to test its functionality. It is advisable to only test one new device at a time.

When you run the simulation, the source code model is automatically compiled and written to an ASCII text file called *Simlist*.TXT, in the same directory as the schematic that you are simulating. This file also contains a listing of the execution order of the source code model. Refine the SimCode source model as needed, and continue testing until you have completely debugged the model.

Step 5 – Create the compiled SimCode model file

Once the SimCode has been successfully compiled you can extract the compiled model information from the *Simlist*.TXT file, and create a compiled model file (74LS74.SCB for this example). Again you can store multiple models in a single file, but you must set the “file=” parameter in the .MDL file to be the same as the file name of the compiled SimCode model library. The default location for compiled SimCode models

is in the root folder of the \Program Files\Design Explorer 99 SE\Library\Sim\Simulation Models.ddb database.

SimCode for the 74LS74

Following this description is an example of the SimCode that was written to describe a 74LS74 device. Following is a description of each section of the SimCode:

Section 1 - SimCode Function Identification

#ls74 source identifies the beginning of the SimCode source function for the 74LS74.

Section 2 - Data declarations

This section consists of pin and variable declarations.

The *INPUTS* statement declares the names of the input pins. VCC and GND pins are included in this statement. The order of these pins must match the order of their corresponding pin declarations in Library Text Field 5 of the schematic symbol.

The *OUTPUTS* statement declares the names of the output pins. Notice that the input pins are listed here as well, but with the suffix “_LD”. Input pins must also be declared as outputs so that the device can provide a load back on the driving circuitry. VCC pins are included in this statement, but not GND pins. The order of these pins must match the order of their corresponding pin declarations in Library Text Field 5 of the schematic symbol.

The *PWR_GND_PINS* statement declares which pins will be used for device power and ground, and samples their voltage levels for later use in the SimCode.

Section 3 - SimCode Function Initialization

The “*IF (init_sim) THEN*” section is executed only once, at the beginning of the simulation. In this section we set the device characteristics that are not subject to change due to outside influences, such as databook specifications. The outputs states should also be initialized here to their “most likely” state. The *EXIT* command should be placed at the end of this section.

Section 4 - LOAD and DRIVE Statements

These statements are used to declare the load and drive capabilities of the device pins.

Section 5 - Device Functionality

This section can vary dramatically from part to part. In this example an *EXT_TABLE* command has been used. Other device models use a variety of *IF...THEN*, *STATE_BIT*, *NUMBER*, and other statements to define the logical function of the device.

Section 6 - Tests for Device Setup Violations

These tests warn of device setup violations which, in the real world, may cause a device not to function properly. In the simulation, the device will generally still function, but warnings, if enabled, will be displayed.

Section 7 - Output Delays/Post Events

The *DELAY* statements occur at the end of the SimCode function. These statements actually post the events to the simulator to let it know that something has changed, and when these events are scheduled to occur, relative to the rest of the simulation. Timing (propagation delay) is assigned to each output based on the databook specifications, input stimulus and the functionality of the device.

```
//=====
①
# ls74 source
//1/2- 74LS74 D flip-flop Digital Simcode Model
//typical prop delay values from TI 1981 2nd edition data book
//=====

②
INPUTS VCC, GND, PRE, DATA, CLK, CLR;
OUTPUTS VCC_LD, PRE_LD, DATA_LD, CLK_LD, CLR_LD, QN, Q;
INTEGERS tblIndex;
REALS tplh_val, tphl_val, ts_val, th_val, trec_val, tt_val, temp_tp,
      clk_twl, clk_twh, pre_clr_twl, ril_val, rih_val, ricc_val;

PWR_GND_PINS(VCC,GND);          //set pwr_param and gnd_param values
SUPPLY_MIN_MAX(4.75,5.25);      //test for min supply=4.75 and max supply=5.25
VOL_VOH_MIN(0.2,-0.4,0.1);      //vol_param=gnd_param+0.2,voh_param=pwr_param-0.4
VIL_VIH_VALUE(1.25,1.35);      //set input threshold values: vil and vih
IO_PAIRS(PRE:PRE_LD, DATA:DATA_LD, CLK:CLK_LD, CLR:CLR_LD);

③
IF (init_sim) THEN
  BEGIN                      //select prop delay, setup, hold, and width times
    //MESSAGE("time\t\tPRE\t\tCLR\t\tCLK\t\tDATA\t\tQ\t\tQN"); //debug

    //NOTE: both ttlh and tthl are the same value
    tt_val= (MIN_TYP_MAX(tt_param: NULL, 5n,  NULL));

    temp_tp= (PWL_TABLE(sim_temp: -75, -5n, 125, 5n)); //tp temperature affect
    tplh_val= (MIN_TYP_MAX(tp_param: NULL, 14n, 25n)) + temp_tp;
    tphl_val= (MIN_TYP_MAX(tp_param: NULL, 20n, 40n)) + temp_tp;

    ts_val= (20n);
    th_val= (5n);
    trec_val= (5n);
    clk_twl= (25n);          //not specified - derived from fmax
    clk_twh= (25n);
    pre_clr_twl= (20n);

    //LS stdout drive IOL max=8mA @ VOL typ=0.35V: rol_param=0.35V/8mA=43.75
    //LS stdout drive IOL max=8mA @ VOL max=0.5V: rol_param=0.5V/8mA=62.5
    rol_param= (MIN_TYP_MAX(drv_param: 62.5, 43.75,  NULL));

    //LS stdout drive IOS min=20mA @ VCC max=5.25V: roh_param=5.25V/20mA=262.5
    //LS stdout drive IOS max=100mA @ VCC max=5.25V: roh_param=5.25V/100mA=52.5
    roh_param= (MIN_TYP_MAX(drv_param: 262.5, NULL,  52.5));
```

Mixed-Signal Simulation

```

//LS input load IIH max=20uA @ Vin=2.7V: ril= (2.7-vol_param)/20uA=125k
ril_val= (MIN_TYP_MAX(ld_param: NULL, NULL, 125k));
//LS input load IIL max=-0.4mA @ Vin=0.4V:rih= (voh_param-0.4)/0.4mA=10.5k
rih_val= (MIN_TYP_MAX(ld_param: NULL, NULL, 10.5k));

//Icc @ 5V: 2500= 4mA/2 typical, 1250= 8mA/2 max
ricc_val= (MIN_TYP_MAX(i_param: NULL, 2500, 1250));

STATE Q = ONE;           // initialize output states
STATE QN = ZERO;
EXIT;
END;

④
DRIVE Q QN = (v0=vol_param,v1=voh_param,ttlh=tt_val,tthl=tt_val);
LOAD PRE_LD DATA_LD CLK_LD CLR_LD =
(v0=vol_param,r0=ril_val,v1=voh_param,r1=rih_val,io=1e9,t=1p);

⑤
EXT_TABLE tblIndex
PRE CLR CLK DATA      Q      QN
0  1  X  X            H      L
1  0  X  X            L      H
0  0  X  X            H      H
1  1  ^  X            DATA  ~DATA
1  1  X  X            Q      ~Q;

//MESSAGE("%fs\t%d\t%d\t%d\t%d\t%d\t%d",present_time,PRE,CLR,CLK,DATA,Q,QN);
LOAD VCC_LD = (v0=gnd_param,r0=ricc_val,t=1p);

⑥
IF (warn_param) THEN
  BEGIN
    IF (PRE && CLR) THEN
      BEGIN
        SETUP_HOLD(CLK=LH DATA Ts=ts_val Th=th_val "CLK->DATA");
        RECOVER(CLK=LH PRE CLR Trec=trec_val "CLK->PRE or CLR");
        WIDTH(CLK Twl=clk_twl Twh=clk_twh "CLK");
        WIDTH(PRE CLR Twl= pre_clr_twl "PRE or CLR");
      END;
    END;

  DELAY Q QN =
    CASE (TRAN_LH) : tphl_val
    CASE (TRAN_HL) : tphl_val
  END;
  EXIT;

```

SimCode Language Definition

The following items make up the Digital SimCode language. The following pages describe each of these items in detail.

INPUTS	Input pins (pins that monitor the circuit)
OUTPUTS	Output pins (pins that drive or load the circuit)
INTEGERS	Integer variables and arrays
REALS	Real variables and arrays
PWR_GND_PINS	Power and ground pins and record supply voltage
IO_PAIRS	Input/output pin associations for input loading

Device Setup Functions

Use these functions to set certain characteristics of the device pins.

VIL_VIH_VALUE	Sets absolute VIL and VIH values
VIL_VIH_PERCENT	Sets VIL and VIH values to a percentage of supply voltage
VOL_VOH_MIN	Sets VOH and VOL relative to power and ground

Device Test Functions

Use these functions to test for any device setup violations which may occur in the circuit. These violations may not affect the simulation of the device's functionality (ie, the device may continue to function in simulation even with setup violations). In order to know if any of these setup violations have occurred, you must enable warnings.

SUPPLY_MIN_MAX	Tests supply pins for min/max supply voltage violations
RECOVER	Tests inputs for recovery time violations
SETUP_HOLD	Tests inputs for setup and hold time violations
WIDTH	Tests inputs for minimum pulse width violations
FREQUENCY(FMAX)	Tests inputs for minimum & maximum frequency violation

Output Pin Functions

Use these functions to program the output pins of a device.

STATE	Sets outputs to the declared logic state
STATE_BIT	Sets outputs to binary weighted logic states
LEVEL	Sets the level of the output state
STRENGTH	Sets the strength of the output state
TABLE	Sets output logic states based on truth table
EXT_TABLE	Sets output logic states based on extended truth table
LOAD	Declares loading characteristics of input pins
DRIVE	Declares drive characteristics of output pins
DELAY	Sets propagation delay to specified outputs
NO_CHANGE	Leaves output state of I/O pins unchanged
EVENT	Causes a digital event to be posted

Expression Operations

Use the operators and functions in expressions to manipulate data, and to make comparisons which control program flow. Expressions are always contained within parentheses (). Operator precedence is from left to right, starting with the inner most parentheses.

Operators	+, -, *, /, ~, !, &&, , ^, &, , >>, <<, >, <, =, !=, >=, <=
Math Functions	POW, ABS, SQRT, EXP, LOG, LOG10, SIN, COS, TAN, ASIN, ACOS, ATAN, HSIN, HCOS, HTAN

Expression Functions

PARAM_SET	Determines if a predefined SimCode param has been set
PWL_TABLE	Returns value from interpolative lookup table
SELECT_VALUE	Returns value from simple lookup table
MIN_TYP_MAX	Returns value from MIN_TYP_MAX lookup table
NUMBER	Returns number based on binary weighted pin states
VALUE	Returns state of the specified pin
CHANGE_TIME	Returns time when the specified pin last changed state
WIDTH_TIME	Returns last pulse width encountered on specified pin
INSTANCE	Checks to see if this is the specified device instance
CHANGED_xx	Checks to see if the specified pin has changed state
READ_DATA	Reads data from an ASCII file into arrays

Program Control

Use these statements to control the flow of the program.

# xxxx source	Identifies the beginning of the SimCode source function
IF ... THEN	Conditionally controls flow through the SimCode
WHILE ... DO	Conditionally controls looping in the SimCode
GOTO	Jumps to a new location in the SimCode
GOSUB	Jumps to a subroutine in the SimCode
RETURN	Returns from a subroutine in the SimCode
EXIT	Terminates SimCode execution

Output Text

Use these commands to display messages during simulation and debugging.

PROMPT	Pause simulation and display a message
MESSAGE	Display a message without pausing

Debug

Allow you to trace through the execution of the SimCode for debugging purposes.

STEP_ON	Turn on the SimCode trace mode
STEP_OFF	Turn off the SimCode trace mode

SimCode Language Syntax

This section describes each of the language items in detail. The following style is used in describing the syntax:

<i>italics</i>	reserved words or emphasis
< >	value/variable/pin/expression
[]	optional parameter
{ } { }	selections (you must choose ONE of these parameters)

xxxx source

Identifies the beginning of the SimCode source function.

General Form

<func name> source

Parameters

<func name> Name of the SimCode function.

Use

This statement identifies the SimCode function so that it can be called when it is time to simulate this device. It must be the first statement of each Digital SimCode device function.

Notes

The simulator's SPICE engine has the ability to read either source code models, or compiled code models. The keyword "source" identifies this as a source code model to the simulator, which automatically compiles the model when the simulation is run. The compiled code is placed in an ASCII text file called Simlist.TXT, in the same directory as the schematic that uses this device.

Example

```
//=====
# MyDevice source
//=====
INPUTS VCC, GND, IN1, IN2
OUTPUTS VCC_LD, IN1_LD, IN2_LD, OUT
.
.
.
EXIT
```

CHANGE_TIME

Returns time when the specified pin last changed state.

General Form

`CHANGE_TIME(<pin>)`

Parameters

<pin> Input or output pin name.

Use

This function returns a real value that indicates the last time the specified input or output pin changed states.

Example

```
T1 = (CHANGE_TIME(INA));
```

CHANGED_xx

Checks if the specified pin has changed state.

General Form

`CHANGED_xx(<pin> [{<>}|{<=}|{>}|{>=}] <var/time/value>]`

Parameters

<pin> Input or output pin name.
<var/time/value> Item to which <pin> is compared.

Use

The *CHANGED_xx* function is used to determine if the specified <pin> has changed state. The *_xx* that follows the keyword CHANGED can be eliminated (to indicate *any* type of change) or the *xx* can be set to:

LH, LX, HL, HX, XL, XH, LZ, ZL, ZH, ZX, HZ or XZ

to indicate a specific type of change. The optional compare operator (<, <=, >, >=) and <var/time/value> would be included to check for a more specific change. If they are not included, the function will return 1 if the pin has changed at the current simulation step.

Examples

```
IF (CHANGED_LH(CLK)) THEN ...
IF (CHANGED(DATA < 10n)) THEN ...
```

DELAY

Sets propagation delay to specified outputs.

General Form 1

```
DELAY <output> [<output> ...] = <delay>;
```

General Form 2

```
DELAY <output> [<output> ...] =
CASE (<conditional exp>) : <delay>
CASE (<conditional exp>) : <delay>
```

```
[CASE (<conditional exp>) : <delay> ...]
END;
```

Parameters

<output>	Name of, or variable index to the output pin.
<conditional exp>	Conditional expression that determines which delay is used.
<delay>	Propagation delay time to the output pin.

Use

The *DELAY* command is executed once for each pin listed and posts a propagation delay for each pin that has changed its level. The *CASE* option allows more than one <delay> to be specified. The <conditional exp> then determines which <delay> will be used. If a delay is set for a pin that has not changed then the pin will be flagged as NO-CHANGE and the delay will *not* be posted. The <delay> can be a real constant, a real variable or a real expression.

Notes

The *DELAY* command must be executed exactly once for each output pin, that is, for each pin declared in the *OUTPUTS* statement which is *not* listed in the *LOAD* or *NO CHANGE* statements. The order in which the delays are set is based on the order in which these pins are listed in the *DELAY* command (i.e. first pin listed is set first). Each <conditional expression> is evaluated in the order it is listed until one expression evaluates TRUE. When this occurs, the <delay> value associated with the TRUE expression is posted for the output being set. When using the *CASE* option, at least one <conditional exp> should evaluate as TRUE for each output pin listed. If no <conditional exp> evaluates to TRUE, the <delay> associated with the last *CASE* statement is posted.

In addition to the standard expression functions, the following terms apply *only* to the output pin being set and can be used in the <conditional exp> :

TRAN_LH	low-to-high
TRAN_LX	low-to-other
TRAN_HL	high-to-low
TRAN_HX	high-to-other
TRAN_HZ	high-to-tristate
TRAN_XL	other-to-low
TRAN_XH	other-to-high
TRAN_LZ	low-to-tristate
TRAN_ZL	tristate-to-low
TRAN_ZH	tristate-to-high
TRAN_ZX	tristate-to-other
TRAN_XZ	other-to-tristate
TRAN_XX	other-to-different

If the <delay> value is less than or equal to 0.0 a run-time error message will be displayed. Output pins can be specified by using the output pin name or by an integer variable that contains the *index* of an output pin. Pin names and variables *cannot* be mixed in the same *DELAY* statement. References to outputs must be either all pin names or all variable names.

Examples

```
DELAY Q1 Q2 Q3 Q4 = 10n;
DELAY Q QN =
    CASE (TRAN_LH) : tphl_val
    CASE (TRAN_HL) : tphl_val
END;
data = (E0_1 && (CHANGED(D0) || CHANGED(D1)));
DELAY Q1 Q0 =
    CASE (data && TRAN_LH) : tphl_D_Q
    CASE (data && TRAN_HL) : tphl_D_Q
    CASE (TRAN_LH) : tphl_E_Q
    CASE (TRAN_HL) : tphl_E_Q
END;
```

In this example, if data is nonzero and Q1 is changing from High to Low, the tphl_D_Q delay will be posted for Q1. Then, if Q0 is changing from Low to High, the tphl_D_Q delay will be posted for Q0.

DRIVE

Declares drive characteristics of output pins.

General Form

```
DRIVE <output> [<output> ...] =
    (v0=<value> v1=<value> ttlh=<value> tthl=<value>);
```

Parameters

<output>	Name of, or variable index to the output pin.
<value>	Real value or variable.
v0	VOL for the output pin.
v1	VOH for the output pin.
ttlh	Low-to-high transition time for the output pin.
tthl	High-to-low transition time for the output pin.

Use

The *DRIVE* command is used to declare the output pin's DRIVE characteristics. When the output is set to a LOW state, the output pin is connected to voltage value *v0* through resistance *rol_param*. When the output is set to a HIGH state, the output pin is connected to voltage value *v1* through resistance *roh_param*. The low-to-high transition time is set by *ttlh* and the high-to-low transition time is set by *tthl*.

Notes

Pin names and variables *cannot* be mixed in the same *DRIVE* statement. References to outputs must be either all pin names or all variable names.

The values used for *rol_param* should be derived using the databook specs for VOL. This value represents the total saturation resistance of the pull-down structure of the device's output. A standard LS output in the LOW state, for example, sinking 8mA will not exceed 0.5V, typically closer to 0.35V. Therefore:

for typ LOW state drive: $rol\_param = VOL_{typ} / IOL_{max}$
 $rol\_param = 0.35V / 8mA$
 $rol\_param = 43.75\text{ ohms}$
 for min LOW state drive: $rol\_param = VOL_{max} / IOL_{max}$
 $rol\_param = 0.5V / 8mA$
 $rol\_param = 62.5\text{ ohms}$

The values used for *roh_param* should be derived using the databook specs for IOS, if available. This value represents the total saturation resistance of the pull-up structure of the device's output. A standard LS output in the HIGH state with the output shorted to ground and Vcc=5.25V will source at least 20mA but not more than 100mA. Therefore:

for min HIGH state drive: $roh\_param = VCC_{max} / IOS_{min}$
 $roh\_param = 5.25V / 20mA$
 $roh\_param = 262.5\text{ ohms}$
 for max HIGH state drive: $roh\_param = VCC_{max} / IOS_{max}$
 $roh\_param = 5.25V / 100mA$
 $roh\_param = 52.5\text{ ohms}$

Example

```
rol_param = (MIN_TYP_MAX(drv_param: 62.5, 43.75, NULL));
roh_param = (MIN_TYP_MAX(drv_param: 262.5, NULL, 52.5));
DRIVE Q QN = (v0=vol_param,v1=voh_param,ttlh=ttlh_val,
ttl=ttl_val);
```

See Also

LOAD

EVENT

Causes a digital event to be posted.

General Form

```
EVENT = ({<time>}|{<expression>})
```

Parameters

<time>	Time at which event should occur.
<expression>	Expression indicating time at which event should occur.

Use

In most cases a digital event is posted when one or more INPUT pins for a simcode model changes state. When the event is processed, the simcode for the specified event is called and run. This instruction allows a simcode model to post a digital event at a specified <time>. If the specified EVENT time is greater than the simulation time (indicated by present_time), then a digital event will be posted. If more than one EVENT is posted in a single call to a simcode model, only the longest EVENT <time> will be used. This function allows the creation of one-shots and other similar device models.

Notes

If a digital event for a specific simcode model occurs before an EVENT <time> posted by that simcode, the EVENT <time> must be posted again. For example, if 1) the present simulation time is 1us, 2) a simcode model sets EVENT = 2us and 3) an INPUT pin in the simcode model changes state at 1.5us, then the 2us event must be posted again.

Example

```
EVENT = (present_time + 1e-6); //return in 1us
```

EXIT

Terminates SimCode execution.

General Form

```
EXIT;
```

Use

The *EXIT* statement is used to terminate SimCode execution.

Notes

This is the last line of a SimCode model, but it may also be placed at other locations to abort execution of remaining SimCode.

EXT_TABLE

Sets output logic states based on extended truth table.

General Form

```
EXT_TABLE <line>
<input pin> [<input pin> ...] <output pin> [<output pin> ...]
<input state> [<input state> ...]
    <output state> [<output state> ...];
```

Parameters

<line>	Variable into which the line number used in the table is placed
<input pin>	Name of the input pin
<output pin>	Name of the output pin

<input state> State of the individual inputs
 <output state> State of the individual outputs based on input conditions

Use

The *EXT_TABLE* statement is an extended truth table function used to set the level and strength of the specified outputs. Valid input states are:

0 low (input voltage is $\leq$ *vil_param*)
 1 high (input voltage is $\geq$ *vih_param*)
 ^ low-to-high-transition
 v high-to-low-transition
 X don't care what input voltage is

Valid output states are:

L ZERO (set output level to *vol_param*).
 H ONE (set output level to *voh_param*).
 Z UNKNOWN (set output level to *v3s_param*).

It also allows INPUT and/or OUTPUT pin names with optional prefixes to specify the output states. Prefixes are:

- State is the previous state.
 ~ State is the inverse of the state.
 ~~ State is the inverse of the previous state.

Output state letters can be followed by a colon and a letter to indicate strength:

s STRONG (set output to *rol_param* for L and *roh_param* for H).
 z HI_IMPEDANCE (set output to *r3s_param*).

If a strength character is *not* specified after an output state then STRONG will be used for L and H states and HI_IMPEDANCE will be used for Z states.

Notes

Each row is tested sequentially from top to bottom until the input conditions are met. The outputs are set for the *first* row to meet the input conditions. <line> is set to the line number in the table that was used. If no match was made then <line> is set to 0. Pin names used to specify output states do not need to be in the table heading. Unlike the *TABLE* statement, input variables are not allowed.

Example

```
EXT_TABLE tblIndex
PRE CLR CLK DATA Q QN
0 1 X X H L
1 0 X X L H
0 0 X X H H
1 1 ^ X DATA ~DATA
1 1 X X Q ~Q;
```

This example is representative of 1/2 of a 7474 D type flip-flop. If input pins PRE, CLR, and DATA are all high ($\geq v_{ih\_param}$) and CLK has a low-to-high transition, Q is set to high (v_{oh_param}) and STRONG (roh_param), QN is set to low (vol_param) and STRONG (rol_param) and tblIndex is set to 4.

See Also

REALS, STATE, STATE_BIT, TABLE

FREQUENCY (FMAX)

Tests inputs for minimum and maximum frequency violation.

General Form

```
FREQUENCY(<input> [<input>...] MIN=<frequency> MAX=<frequency>
["<message>"])
```

Parameters

<input>	Name of, or variable index to the input pin under test.
MIN	Minimum frequency allowed on the pin under test.
MAX	Maximum frequency allowed on the pin under test.
<message>	Text string that will be displayed if a warning occurs.

Use

The *FREQUENCY* function compares the <input> period (the time from one low-to-high edge to the next low-to-high edge) with the reciprocal of the specified <frequency> (1/freq). If the time period for the <input> is smaller than the reciprocal of the specified MAX<frequency> or the time period for the <input> is greater than the reciprocal of the specified MIN<frequency>, then a WARNING will be displayed. An optional <message> string can be included in the *FREQUENCY* statement which will be output if a WARNING is displayed.

Notes

Databook specifications should be used with this function. Pin and variable names *can* be mixed in the same *FREQUENCY* statement. Only the first FREQUENCY failure for each pin listed will be reported.

Example

```
FREQUENCY(CLK MAX=10MEG "CLK"); //check fmax only
```

GOSUB

Jumps to a subroutine in the SimCode.

General Form

```
GOSUB <label>;
```

Parameters

<label>	Location in SimCode where program flow resumes.

Use

The *GOSUB* instruction is used to perform non-sequential execution of the SimCode. However, unlike the *GOTO* statement, SimCode execution will continue on the instruction following the *GOSUB* instruction when a *RETURN* instruction is encountered in the SimCode being run.

Example

```
GOSUB Shift_Left;
.
.
.
Exit;
Shift_Left:
.
.
.
RETURN;
```

See Also

RETURN

GOTO

Jumps to a new location in the SimCode.

General Form

```
GOTO <label>;
```

Parameters

<label> Location in SimCode where program flow resumes.

Use

The *GOTO* instruction is used to perform non-sequential execution of the SimCode.

Notes

Program flow resumes from the location where <label>: appears in the SimCode. <label> must begin with an alpha character, followed by any number of alpha-numeric characters or the underscore (`_`) character. Where <label> appears in the code, it must be followed *immediately* by a colon (`:`).

Example

```
GOTO Shutdown;
.
.
.
Shutdown:
.
.
```

```
.
Exit;
```

See Also

GOSUB, IF ... THEN

IF ... THEN

Conditionally controls flow through the SimCode.

General Form 1

```
IF (<expression>) THEN BEGIN ... [ELSE ...] END;
```

General Form 2

```
IF (<expression>) THEN GOTO <label>;
```

Parameters

<expression>	Any expression that can be evaluated as true or false.
<label>	Location in SimCode where program flow resumes.

Use

The *IF ... THEN* statement is used to control the flow of the program, based on whether <expression> evaluates to true or false. Multiple *IF ... THEN* statements may be nested.

Notes

When the BEGIN...ELSE...END form of this statement is used and <expression> evaluates to true, program flow resumes from the BEGIN statement and skips any optional SimCode between the ELSE and END statements. If <expression> evaluates to false, program flow resumes from the optional ELSE statement if it exists or after the END statement if it does not exist.

When the GOTO form of this statement is used and <expression> evaluates to true, program flow resumes from the location where <label>: appears in the SimCode. <label> must begin with an alpha character, followed by any number of alpha-numeric characters or the underscore (_) character. Where <label> appears in the code, it must be followed *immediately* by a colon (:).

Examples

```
IF (EN) THEN
  BEGIN
    STATE Q0 = UNKNOWN
  ELSE
    IF (IN2) THEN
      BEGIN
        STATE Y2 = ONE
      ELSE
        STATE Y2 = ZERO
      END
    END
  END
```

```
IF (x = -2) THEN GOTO Do_Neg2
.  
.  
.  
Do_Neg2:  
.  
.  
.
```

See Also

GOTO, WHILE ... DO

INPUTS

Declares input pins (pins that monitor the circuit).

General Form

```
INPUTS <input pin>[, <input pin>,
```

Parameters

<input pin> Name of the input pin.

Use

The *INPUTS* data type is used to define the pins which monitor stimulus external to the device. These generally include input, i/o, power and ground pins.

Notes

Input pin names *must* begin with a letter and be defined before they are used.

Example

```
INPUTS VCC, GND, PRE, DATA, CLK, CLR
```

See Also

OUTPUTS, IO_PAIRS, PWR_GND_PINS

INSTANCE

Checks if this is the specified device instance.

General Form

```
INSTANCE("<instance name>")
```

Parameters

<instance name> Text string indicating instance name.

Use

The *INSTANCE* function returns 1 if the present instance of the SimCode device matches the <instance name> specified. Otherwise it returns 0;

Notes

A circuit may contain more than one of any given device. During simulation it may be important to know if the device being simulated at this moment is the one you are interested in. This would allow you, for example, to print messages for one specific NAND gate without having to wade through messages for all the other NAND gates as well. The instance name is the device Designation preceded by its SPICE Prefix Character (the letter A).

Example

```
IF (INSTANCE("AU23")) THEN
  BEGIN
    MESSAGE("U23-Q0 = %d", Q0);
  END;
```

INTEGERS

Declares integer variables and arrays.

General Form

```
INTEGERS <var>[, <var>, ...];
```

Parameters

<var> Name of the variable.

Use

The *INTEGERS* data type is used to define integer variables and arrays.

Notes

Integer variables and arrays *must* begin with a letter and be defined before they are used. Integer arrays are defined by following the array name with a left bracket ([), an integer number which defines the size of the array, and a right bracket (]). Integer arrays can be set and/or used in expressions.

The following are reserved SimCode integer variables which do not need to be declared:

Variable	Use	Digital Mode Parameter	SPICE Option
tp_param	tplh/hl index	Propagation Delays	TPMNTYMX
tt_param	ttlh/hl index	Transition Times	TTMNTYMX
ld_param	LOAD index	Input Loading	LDMNTYMX
drv_param	DRIVE index	Output Drive	DRVMNTYMX
i_param	ICC index	Device Current	IMNTYMX
user_param	USER index	User Defined	USERMNTYMX
warn_param	Warning messages	WARN flag	SIMWARN

Mixed-Signal Simulation

init_sim	1 during simcode init	N/A	N/A
tran_pin	TRAN_xx pin index	N/A	N/A

The first six variables in this list are expected to have a value of 1, 2 or 3. These values represent an index into the min/typ/max arrays:

Value	Represents

1	Index to minimum value.

2	Index to typical value.

3	Index to maximum value.

The Digital Model Parameter can be set independently for each digital device in the Digital Model Parameters dialog box. If a SPICE Option parameter is set in the Analog Options dialog box, that setting will globally override the Digital Model Parameter settings for all digital devices. If the variable is set explicitly in the SimCode, that setting will override all other settings.

warn_param can be set to any positive value to conditionally display warning messages for the device. Different levels of warning could be created by the device programmer, accessed by entering different positive values. The value of *init_sim* is set to 1 during SimCode initialization, otherwise it is set to 0. The value of *tran_pin* is set to the index of the pin being set during a *DELAY CASE* statement. This index is used to determine which pin the *TRAN_xx* instruction is applied to.

Example

```
INTEGERS tblIndex, count, data[64];
```

See Also

DELAY, MIN_TYP_MAX

IO_PAIRS

Declares input/output pin associations for input loading.

General Form

```
IO_PAIRS (<ipin:opin>[, <ipin:opin>,
```

Parameters

<ipin:opin> Pin names of associated input and output pins.

Use

The *IO PAIRS* statement defines which of the *INPUTS* pins are associated with which of the *OUTPUTS* pins. This association is used by the *LOAD* statement.

Notes

Each physical input pin on a device consists of both an ipin and an opin in SimCode. The opin is required to provide input loading characteristics. This statement can only be used once in the SimCode. Power pins are not listed in the *IO_PAIRS* statement.

Example

```
IO_PAIRS (IN1:IN1_LD, IN2:IN2_LD
```

In this example, IN1 and IN2 are *INPUTS* and IN1_LD and IN2_LD are *OUTPUTS*. IN1 and IN1_LD both refer to the same physical pin on the device.

See Also

INPUTS, OUTPUTS, LOAD

LEVEL

Sets the level of the output state.

General Form 1

```
LEVEL <output> [<output> ...] = (<expression>
```

General Form 2

```
LEVEL <output> [<output> ...] = {ZERO}|{ONE}|{UNKNOWN
```

Parameters

<output> Name of, or variable index to the output.
<expression> Any expression compared to VOL or VOH.

Use

The state of an output pin is determined by its level and its strength. Use the *LEVEL* command to set the level of one or more output pins.

<expression>	State	Level
<= <i>vil_param</i>	ZERO	<i>vil_param</i>
>= <i>voh_param</i>	ONE	<i>voh_param</i>
other	UNKNOWN	<i>v3s_param</i>

Notes

Output pins can be specified by using the output pin name, or by an integer variable that contains the INDEX of an output pin. Pin and variable names *cannot* be mixed in the same *LEVEL* statement. References to outputs must be either all pin names or all variable names.

Examples

```
LEVEL Q = ONE;
LEVEL Q1 Q2 Q3 Q4 = ZERO;
LEVEL OUT = ((1+2)/3);
```

In the last example, OUT will be:

ZERO if *vil_param* > 1

UNKNOWN if $vil\_param < 1$ and $vih\_param > 1$

ONE if $vih\_param < 1$

See Also

REALS, STATE, STATE_BIT, STRENGTH

LOAD

Declares loading characteristics of input pins.

General Form

```
LOAD <output> [<output> ...] =
    (v0=<value> r0=<value> [v1=<value> r1=<value>] [io=<value>] t=<value>);
```

Parameters

<output>	Name of, or variable index to the output pin.
<value>	Real value or variable.
<i>v0</i>	Load voltage for HIGH state input.
<i>r0</i>	Load resistance for HIGH state input.
<i>v1</i>	Load voltage for LOW state input.
<i>r1</i>	Load resistance for LOW state input.
<i>io</i>	Off-state load resistance for unused load.
<i>t</i>	Time delay before the load will be applied.

Use

The *LOAD* command is typically used with input or power pins to provide loading for the driving circuit. Since only output pins can provide a load, each input must have a corresponding output. These are assigned using the *IO_PAIRS* statement.

If different loads are required for different inputs, multiple *LOAD* statements may be used. Power pins should be placed in a separate *LOAD* statement which does not include the *v1/r1* load or *io*. Power pins are not included in the *IO_PAIRS* statement. The *IO_PAIRS* statement must be entered before any *LOAD* statements that contain *io*.

Notes

An input load consists of a voltage and a resistance (*v0/r0* or *v1/r1*). The voltage level of the incoming signal determines which load will be used. If the voltage level goes below *VIL* and remains below *VIH*, then the input is considered to be in the LOW state and the *v1/r1* is applied. If the voltage level goes above *VIH* and remains above *VIL*, then the input is considered to be in the HIGH state and the *v0/r0* is applied. *io* is the input state off resistance. The unused load is essentially removed from the circuit by changing its *r* value to the value specified for *io*.

The values for *v0*, *r0*, *v1* and *r1* can be either real constants or real variables. The values for *io* and *t* must be real constants. Pin names and pin variables *cannot* be mixed

in the same *LOAD* statement. References to outputs must be either all pin names or all variable names.

For input pins, the values used for *r0* should be derived using the databook specs for I_{IH}. A standard LS input, for example, will sink a maximum of 20uA at V_{in}=2.7V. Therefore, if *vol_param* = 0.2V, then:

$$\begin{aligned}\text{for max HIGH state load: } r0 &= (V_{in} - vol\_param) / I_{IHmax} \\ r0 &= (2.7V - 0.2V) / 20uA \\ r0 &= 125k\text{ ohms}\end{aligned}$$

The values used for *r1* should be derived using the databook specs for I_{IL}. A standard LS input, for example, will source a maximum of 400uA at V_{in}=0.4V. Therefore, if *voh_param* = 4.6V then:

$$\begin{aligned}\text{for max LOW state load: } r1 &= (voh\_param - V_{in}) / I_{ILmax} \\ r1 &= (4.6V - 0.4V) / 400uA \\ r1 &= 10.5k\text{ ohms}\end{aligned}$$

For power pins, the value used for *r0* should be derived using the databook specs for I_{CC}. For a 74LS151, I_{cc typ} is 6mA at V_{cc}=5V and I_{cc max} is 10mA at V_{cc}=5.25V. Therefore:

$$\begin{aligned}\text{for Icc typ: } r0 &= 5V / 6mA = 833\text{ ohms} \\ \text{for Icc max: } r0 &= 5.25V / 10mA = 525\text{ ohms}\end{aligned}$$

If creating a multiple-parts-per-package device, such as a 74LS00 quad NAND gate, you should adjust the I_{cc} load for the individual parts accordingly.

Examples

```
r0_val = (MIN_TYP_MAX(ld_param: NULL, NULL, 125k);
r1_val = (MIN_TYP_MAX(ld_param: NULL, NULL, 10.5k);
ricc_val = (MIN_TYP_MAX(ld_param: NULL, 833, 525);
LOAD PRE_LD DATA_LD CLK_LD CLR_LD = (v0=vol_param, r0=r0_val,
    v1=voh_param, r1=r1_val, io=1e9, t=1p);
LOAD VCC_LD = (v0=gnd_param, r0=ricc_val, t=1p);
```

See Also

IO_PAIRS, DRIVE

MATH FUNCTIONS

Function	Description	Example
POW	power	X= (12 POW(3));
ABS	absolute value	X= (ABS(-12));
SQRT	square-root	X= (SQRT(2));
EXP	exponent	X= (EXP(10));
LOG	natural log	X= (LOG(0.1));
LOG10	log base 10	X= (LOG10(0.1));
SIN	sine	X= (SIN(0.1));
COS	cosine	X= (COS(0.1));
TAN	tangent	X= (TAN(0.1));
ASIN	arc sine	X= (ASIN(0.1));
ACOS	arc cosine	X= (ACOS(0.1));
ATAN	arc tangent	X= (ATAN(0.1));
HSIN	hyperbolic sine	X= (HSIN(0.1));
HCOS	hyperbolic cosine	X= (HCOS(0.1));
HTAN	hyperbolic tangent	X= (HTAN(0.1));

See Also

OPERATORS

MESSAGE

Displays a message without pausing.

General Form

```
MESSAGE ("<message>" [, <value/pin>...]);
```

Parameters

<message> Message string including formatting characters as needed.
 <value> Variable or constant value.
 <pin> Pin name or index to pin variable.

Use

The *MESSAGE* statement is used to output the information specified by the <message> string. It does *not* interrupt the simulation. The message is displayed in the status window during simulation.

Notes

A format string in *MESSAGE* is similar to a format that may be used in a printf statement in C. Valid formatting characters include (but are not limited to):

\t	tab
\n	new line
\r	Carriage return
%d	Decimal display for short variable or current input/output state
%D	Decimal display for short variable or old input/output state
%x	Hex display for short variable or current input/output state
%X	Hex display for short variable or old input/output state
%c	Character display for short variable or current input/output state
%C	Character display for short variable or old input/output state
%e	Exponential display for real variable
%f	Floating point engineering display for real variable
%g	Short display (%e or %f) for real variable
%s	String constant display

Note: The *only* valid string constants are:

INSTANCE	The present SimCode device instance name.
FUNC	The present SimCode device function name.
FILE	The present SimCode device file name.

Examples

```
MESSAGE("time\t\tCLK\tDATA\tQ\tQN");
MESSAGE("device instance= %s",INSTANCE);
MESSAGE("%.3e\t%d\t%d\t%d\t%d",present_time,CLK,DATA,Q,QN);
```

See Also

PROMPT

MIN_TYP_MAX

Returns value from MIN_TYP_MAX look-up table.

General Form

```
MIN_TYP_MAX(<index>: <min>, <typ>, <max>);
```

Parameters

<index>	Input variable (index to select min, typ or max values).
<min>	Minimum databook value.
<typ>	Typical databook value.
<max>	Maximum databook value.

Use

The *MIN_TYP_MAX* function is similar to the *SELECT_VALUE* function except that three (3) values/variables *must* be entered. The keyword “NULL” can be substituted for one or two unknown values. If a predefined integer variable (see *INTEGERS*) is used as the <index>, unknown (NULL) values are calculated from the known values as follows:

Known Values	Formula
<min>, <max>	typical = (<max> + <min>) / 2;
<min> only	typical = (<min> / <min scale factor>) maximum = (<min> / <min scale factor>) * <max scale factor>
<typ> only	minimum = (<typ> * <min scale factor>) maximum = (<typ> * <max scale factor>)
<max> only	minimum = (<max> / <max scale factor>) * <min scale factor> typical = (<max> / <max scale factor>)

Notes

If <index> is not one of the predefined variables listed below, then <min scale factor> = 0.5 and <max scale factor> = 1.5. The <min scale factor> and <max scale factor> for each of these predefined variables can be changed in the Analog Options dialog. The <min scale factor> and <max scale factors> are reversed for *ld_param*, *drv_param* and *i_param* because these parameters control a resistance value rather than a current value (i.e., maximum load equates to minimum resistance).

Variable	SPICE Option	Parameter	Default
<i>tp_param</i>	PROPMNS	<min scale factor>	0.5
<i>tp_param</i>	PROPMXS	<max scale factor>	1.5
<i>tt_param</i>	TRANMNS	<min scale factor>	0.5
<i>tt_param</i>	TRANMXS	<max scale factor>	1.5
<i>ld_param</i>	LOADMNS	<min scale factor>	1.5
<i>ld_param</i>	LOADMXS	<max scale factor>	0.5
<i>drv_param</i>	DRIVEMNS	<min scale factor>	1.5
<i>drv_param</i>	DRIVEMXS	<max scale factor>	0.5
<i>i_param</i>	CURRENTMNS	<min scale factor>	1.5
<i>i_param</i>	CURRENTMXS	<max scale factor>	0.5
<i>vth_param</i>	VTHMNS	<min scale factor>	0.5
<i>vth_param</i>	VTHMXS	<max scale factor>	1.5
<i>user_param</i>	USERMNS	<min scale factor>	0.5
<i>user_param</i>	USERMXS	<max scale factor>	1.5

Examples

```
tplh_val = (MIN_TYP_MAX(tp_param: NULL, 5n, NULL));
```

In this example, if we assume that PROPMNS and PROPMXS are set to their default values, then:

```
if tp_param = 1, then tplh_val = 2.5n
if tp_param = 2, then tplh_val = 5n
if tp_param = 3, then tplh_val = 10n
```

```
ricch_val = (MIN_TYP_MAX(i_param: NULL, 2500, 1250));
```

In this example, if we assume that CURRENTMNS and CURRENTMXS are set to their default values, then:

```
if i_param = 1, then ricch_val = 5000
if i_param = 2, then ricch_val = 2500
if i_param = 3, then ricch_val = 1250
```

See Also

INTEGERS, SELECT_VALUE

NO_CHANGE

Leaves output state of I/O pins unchanged.

General Form

```
NO_CHANGE <output> [<output> ...];
```

Parameters

<output> Name of, or variable index to the output pin.

Use

Use the *NO_CHANGE* function to indicate no-change for specified output pins. Use this statement on bi-directional pins when the bi-directional pin is being treated as an input.

Notes

Pin names and variables *cannot* be mixed in the same *NO_CHANGE* statement. References to outputs must be either all pin names or all variable names.

Example

```
NO_CHANGE Q1 Q2 Q3 Q4;
```

NUMBER

Returns number based on binary weighted pin states.

General Form

```
NUMBER(<MSB pin>, [<pin>, ...] <LSB pin> );
```

Mixed-Signal Simulation

Parameters

<pin> Name of or index to a pin.

Use

The *NUMBER* function returns a short integer that represents the decimal value of the binary number represented by the list of <pin>. Each bit (represented by a <pin>) is set to 1 if the <pin> is non-zero, otherwise it is set to 0.

Notes

The first <pin> in the list represents the most-significant-bit (MSB) and the last <pin> in the list represents the least-significant-bit (LSB).

Example

```
A = (NUMBER(D3,D2,D1,D0)) ;
```

In this example, if D3 is HIGH, and D2, D1 and D0 are LOW (1000_2), then $A = 8$.

OPERATORS

Assignment Operator

= Equals (sets a variable or output pin to a value or state).

Math Operators

+ Add

- Subtract

* Multiply

/ Divide

Unary Operators

~ Logical not

! Bitwise complement

Logical Operators

&& AND

|| OR

^^ XOR

Bitwise Operators

& AND

| OR

^ XOR

<< Shift left

>> Shift right

Relative Comparators

= Equal
 != Not equal
 < Less than
 <= Less than or equal to
 > Greater than
 >= Greater than or equal to

Use

Operators are used to set and manipulate variables and expressions.

Notes

Expressions must be enclosed within parentheses (). Expressions are *always* evaluated from left to right within parentheses. You should use parentheses to set precedence within an expression. When using the Unary Operators on values, variables, expressions, etc, the values, variables, expressions, etc, must be in parentheses ().

Examples

```
clk_twl = (25n);
reg = (reg + 1);
vx = (vol_param - 10m);
C = (A * B);
val = (xval / 2);
X = (A && ~(B));
Y = (! (X));           //if X=1 then Y=FFFFFFFE
A = (X & 1);           //if X=1 then A=1, if X=2 then A=0
B = (X | 8);           //if X=1 then B=9, if X=2 then B=10
C = (X >> 2);          //if X=1 then C=0, if X=2 then C=0
D = (2 >> X);          //if X=1 then D=1, if X=2 then D=0
E = (X << 2);          //if X=1 then E=4, if X=2 then E=8
F = (2 << X);          //if X=1 then F=4, if X=2 then F=8
IF (A >= B) THEN ...
IF ((A < 2) && (B > 3)) THEN ...
IF ((C < 2) || (X > 4)) THEN ...
```

See Also

MATH FUNCTIONS

OUTPUTS

Declares output pins (pins that drive or load the circuit).

General Form

```
OUTPUTS <output pin>[, <output pin>, ...];
```

Parameters

<output pin> Name of the output pin.

Use

The *OUTPUTS* data type is used to define the pins which affect the operation of circuitry external to the device. These generally include input, output, I/O and power pins. Input and power pins are included in this list because their presence constitutes a load on the driving circuitry.

Notes

Output pin names *must* begin with a letter and be defined before they are used.

Example

```
OUTPUTS VCC_LD, PRE_LD, DATA_LD, CLK_LD, CLR_LD, QN, Q;
```

See Also

INPUTS, IO_PAIRS, PWR_GND_PINS

PARAM_SET

Determines if a predefined SimCode param has been set.

General Form

```
PARAM_SET(<param var>)
```

Parameters

<param var> SimCode model definition parameter.

Use

The *PARAM_SET* function is used to determine if a parameter in the SimCode model definition has been set. It returns 1 if the specified parameter was set (e.g., *vil_param*=0.8) otherwise it returns 0.

Notes

See *INTEGER* and *REAL* declarations for a list of SimCode model definition parameters and their associated variable names.

Example

```
A = PARAM_SET(ld_param);  
IF (PARAM_SET(voh_param)) THEN ...
```

See Also

INTEGERS, REALS

PROMPT

Pauses simulation and displays a message.

General Form

```
PROMPT("<message>"[, <value/pin>...]);
```

Parameters

<message> Message string including formatting characters as needed.
 <value> Variable or constant value.
 <pin> Pin name or index to pin variable.

Use

The *PROMPT* statement is used to stop simulation and display the information specified by the <message> string. The message is displayed in the status window during simulation. The user must click on a button to continue execution of the SimCode.

Notes

A format string in *PROMPT* is similar to a format that may be used in a printf statement in C. Valid formatting characters include (but are not limited to):

\t	tab
\n	new line
\r	Carriage return
%d	Decimal display for short variable or current input/output state
%D	Decimal display for short variable or old input/output state
%x	Hex display for short variable or current input/output state
%X	Hex display for short variable or old input/output state
%c	Character display for short variable or current input/output state
%C	Character display for short variable or old input/output state
%e	Exponential display for real variable
%f	Floating point engineering display for real variable
%g	Short display (%e or %f) for real variable
%s	String constant display

Note: The *only* valid string constants are:

INSTANCE The present SimCode device instance name.
 FUNC The present SimCode device function name.
 FILE The present SimCode device file name.

Example

```
PROMPT("input=%d time=%f device=%s", D1, t1, INSTANCE);
```

See Also

MESSAGE

PWL_TABLE

Returns value from interpolative look-up table.

General Form

```
PWL_TABLE (<IN var>: <IN1>,<OUT1>,<IN2>,<OUT2>[,...<OUTn>,<OUTn>])
```

Parameters

<IN var> input variable (integer or real)
 <INx> input compare value
 <OUTx> output value at <INx>

Use

This piece-wise-linear function is essentially a look-up table. The value of <IN var> is used to look up an entry in the table which consists of pairs of values. The first value in each pair is an input compare value and the second value is the corresponding output value. If the <IN var> value is less than the first <IN> value the first <OUT> value is returned. If the <IN var> value is greater than the last <INn> value then the last <OUTn> value is returned. Linear interpolation is done between entries according to the formula:

$$\text{value} = (((\text{OUTA} - \text{OUTB}) / (\text{INA} - \text{INB})) * (\text{IN var} - \text{INA}) + \text{OUTA})$$

where <IN var> falls between the input compare values INA and INB. The actual output value will fall between output values OUTA and OUTB.

Notes

Two or more IN/OUT data value pairs must be entered and the IN values must be entered in ascending order. There is no limit to the maximum number of IN/OUT data pairs that can be entered.

Example

```
twh = (PWL_TABLE(var: 5,180n,10,120n,15,80n));
```

In this example, if var = 10 then twh = 120n and if var = 12 then twh = 104.

See Also

SELECT_VALUE

PWR_GND_PINS

Declares power and ground pins; records supply voltage.

General Form

```
PWR_GND_PINS (<pwrpin>, <gndpin>);
```

Parameters

<pwrpin> name of the power pin
 <gndpin> name of the ground pin

Use

The *PWR_GND_PINS* statement defines which of the *INPUTS* pins are power and ground and sets the Power and Ground parameters of the device to absolute voltages as follows:

pwr_param = voltage on <pwrpin>
gnd_param = voltage on <gndpin>

Notes

This statement can only be used once in the SimCode. Only one pin can be defined for power and one for ground.

Example

```
PWR_GND_PINS (VCC, GND) ;
```

See Also

INPUTS, OUTPUTS, REALS, VIL_VIH_PERCENT, SUPPLY_MIN_MAX

READ_DATA

Reads data from an ASCII file into arrays.

General Form

```
READ_DATA(<array>[, <array>, ...])
```

Parameters

<array> Name of the array into which the value is placed.

Use

The *READ_DATA* function opens the file specified by the “data=” parameter in the device’s .MODEL statement and reads ASCII text data. The number and type (integer/real) of the values per line that will be read is based on the number and type of array variables that are specified in the function call. The number of data lines read is determined by the number of data lines in the specified file and/or the size of the smallest array in the function call. The *READ_DATA* function returns the number of lines read. A negative number is returned if an error is encountered:

- 1 Invalid file name
- 2 Can’t find file
- 3 Invalid array
- 4 Illegal array access
- 5 Data type
- 6 Expected data value

Notes

Multiple values per line in the data file must be separated by commas. The real values in the data file *must* be in scientific notation. The device’s .MODEL statement which contains the “data=” parameter must be placed in the device symbol’s .MDL file.

Example

MYDEVICE.MDL file:

```
.MODEL AMYDEVICE XSIMCODE(file="{MODEL_PATH}MYDEVICES.SCB" +
func=MyDevice data="{MODEL_PATH}MYDEVICE.DAT" {mntymx})
```

MYDEVICE.DAT file:

```
8, 8E-6
9, 9E-6
10, 1E-5
11, 1.1E-5
```

MyDevice SimCode:

```
nlines = READ_DATA(int_array, real_array);
```

This example opens a file called MYDEVICE.DAT in the Models directory. It reads two columns of data from the file where the first column contains integer values and the second column contains real values. If the arrays are declared as int_array[3] and real_array[5] then only the first 3 data lines will be read and nlines will be set to 3.

REALS

Declares real variables and arrays.

General Form

```
REALS <var>[, <var>, ...];
```

Parameters

<var> name of the variable

Use

The *REALS* data type is used to define real variables and arrays.

Notes

Real variables and arrays *must* begin with a letter and be defined before they are used. Real arrays are defined by following the array name with a left bracket ([), an integer number which defines the size of the array, and a right bracket (]). Real arrays can be set and/or used in expressions.

The following are reserved SimCode real variables which do not need to be declared:

Variable	Use	Digital Model Parameter
vil_param	low input state value	VIL value
vih_param	high input state value	VIH value
vol_param	low output state value	VOL value
voh_param	high output state value	VOH value
v3s_param	tri-state output state value	N/A
rol_param	low output strength value	N/A

roh_param	high output strength value	N/A
r3s_param	tri-state output strength value	N/A
pwr_param	Voltage on power pin	PWR value
gnd_param	Voltage on ground pin	GND value
present_time	Present simulation time	N/A
previous_time	Previous simulation time	N/A
sim_temp	Circuit operating temperature	N/A (SPICE Option: TEMP)

The Digital Model Parameter can be set independently for each digital device in the Digital Model Parameters dialog box. If the variable is set explicitly in the SimCode, that setting will override all other settings.

The values of *pwr_param* and *gnd_param* are set each time the *PWR GND PINS* statement is executed. The value of *present_time* and *previous_time* are set each time the time step changes. The value of *sim_temp* is the current operating temperature of the circuit which can be set from the SPICE Option “TEMP”.

Example

```
REALS tplh_val, tphl_val, ricc_val, vbias, values[64];
```

See Also

PWR_GND_PIN, *VIL_VIH_VALUE*, *VIL_VIH_PERCENT*, *VOL_VOH_MIN*

RECOVER

Tests inputs for recovery time violations.

General Form

```
RECOVER(<clk input> = {LH}|{HL} <mr input> [<mr input> ...]
```

Parameters

<clk input>	Name of, or index to the input clock/reference pin under test
<mr input>	Name of, or index to the input set/reset pin under test.
<i>TREC</i>	Recovery time for both low and high going <mr pin>.
<i>TRECL</i>	Recovery time for low going <mr pin>.
<i>TRECH</i>	Recovery time for high going <mr pin>.
<message>	Text string that will be displayed if a warning occurs.

Use

The *RECOVER* function compares the time difference between a level change (LH or HL) on the <clk input> and a level change on the <mr input> to a specified test time. RECOVER test times are specified jointly using *TREC*=<time> (which sets *TRECL* and *TRECH* to the same value) or individually using *TRECL*=<time> and *TRECH*=<time>. If the compare time is less than the specified <time> a warning will

be displayed during simulation. An optional <message> string can be included in the *RECOVER* statement which will be output if a warning is displayed.

Notes

Databook specifications should be used with this function. TRECL=<time> and TRECH=<time> can be entered in the same *RECOVER* test. The *RECOVER* test will be made only if the state of the <mr input> matches the time parameter (TRECL=LOW, TRECH=HIGH) when the <clk input> makes the specified transition (LH or HL). For example, if <clk input> = LH and TRECL is specified then the <mr input> must be LOW when the <clk input> goes from LOW to HIGH for a *RECOVER* test to be made. Pin names and variables *can* be mixed in the same *RECOVER* statement.

Example

```
RECOVER(CLK=LH PRE CLR TREC=trec_val
        "CLK->PRE or CLR");
```

RETURN

Returns from a subroutine in the SimCode.

General Form

```
RETURN;
```

Use

The *RETURN* instruction is used to return program flow to the instruction that followed the last *GOSUB* instruction.

See Also

GOSUB

SELECT_VALUE

Returns value from simple look-up table.

General Form

```
SELECT_VALUE (<index>: <val/pin/var>,
              <val/pin/var>[, <val/pin/var>, ...]);
```

Parameters

<index>	Input variable (index to <val/pin/var>)
<val/pin/var>	Output value, pin or variable

Use

The *SELECT_VALUE* function returns the value of the number or variable indicated by the value of the index variable.

Notes

The number of values and/or variables used is not limited.

Example

```
A = (SELECT_VALUE(B: 16, 8, 4, 2, 1));
```

In this example, if B = 2 then A = 8 (the 2nd value).

See Also

PWL_TABLE, MIN_TYP_MAX

SETUP_HOLD

Tests inputs for setup and hold time violations

General Form

```
SETUP_HOLD(<clk input> = {LH}|{HL}
    <data input> [<data input> ...]
    {TS=<time>}|{TSL=<time> TSH=<time>}
    {TH=<time>}|{THL=<time> THH=<time>} ["<message>"];
```

Parameters

<clk input>	Name of, or index to the input clock/reference pin under test.
<data input>	Name of, or index to the input data pin under test.
<i>TS</i>	Setup time for both low and high going <data input>.
<i>TSL</i>	Setup time for low going <data input>.
<i>TSH</i>	Setup time for high going <data input>.
<i>TH</i>	Hold time for both low and high going <data input>.
<i>THL</i>	Hold time for high going <data input>.
<i>THH</i>	Hold time for low going <data input>.
<message>	Text string that will be displayed if a warning occurs.

Use

The *SETUP_HOLD* function compares the time difference between a level change (LH or HL) on the <clk input> and a level change on the <data input> to a specified test time. SETUP test times are specified jointly using TS=<time> (which sets TSL and TSH to the same value) or individually using TSL=<time> and TSH=<time>. HOLD test times are specified jointly using TH=<time> (which sets THL and THH to the same value) or individually using THL=<time> and THH=<time>. If the compare time is less than the specified <time> a WARNING will be displayed. An optional <message> string can be included in a *SETUP_HOLD* statement which will be output if a WARNING is displayed.

Notes

Databook specifications should be used with this function. TSL=<time>, TSH=<time>, THL=<time> and THH=<time> can be entered in the same *SETUP_HOLD* statement. The SETUP and/or HOLD test will be made only if the state of the <data input> matches the time parameter (TSL or THL=LOW, TSH or THH=HIGH) when the <clk input> makes the specified transition (LH or HL). For example, if <clk input>=LH and

TSL is specified, then the <data input> must be LOW when the <clk input> goes from LOW to HIGH for a SETUP test to be made. Pin names and variables *can* be mixed in the same *SETUP_HOLD* statement.

Example

```
SETUP_HOLD(CLK=LH DATA Ts=ts_val Th=th_val "CLK->DATA");
```

STATE

Sets outputs to the declared logic state.

General Form 1

```
STATE <output> [<output>...] = (<expression>);
```

General Form 2

```
STATE <output> [<output>...] = {ZERO} | {ONE} | {UNKNOWN};
```

Parameters

<output> Name of, or variable index to the output pin.
<expression> Any expression to be compared to VIL or VIH.

Use

The state of an output pin is determined by its level and its strength. The *STATE* command sets the level and strength for one or more output pins or variables. If <expression> is less than or equal to *vil_param*, the output will be set to ZERO. If <expression> is greater than or equal to *vih_param*, the output will be set to ONE. Otherwise, the output will be set to UNKNOWN. The level and strength values are set according to the state:

<expression>	State	Level	Strength
<= <i>vil_param</i>	ZERO	<i>vil_param</i>	<i>rol_param</i>
>= <i>voh_param</i>	ONE	<i>voh_param</i>	<i>roh_param</i>
other	UNKNOWN	<i>v3s_param</i>	<i>r3s_param</i>

Notes

Output pins can be specified by using the output pin name or by an integer variable that contains the *index* of an output pin. Pin and variable names *cannot* be mixed in the same *STATE* command. References to outputs must be either all pin names or all variable names.

Examples

```
STATE Q = ONE;
STATE Q1 Q2 Q3 Q4 = ZERO;
STATE OUT = ((1+2)/3);
```

In the last example, OUT will be:

ZERO if *vil_param* > 1

UNKNOWN if *vil_param* < 1 and *vih_param* > 1

ONE if $v_{ih\_param} < 1$

See Also

REALS, STATE_BIT, LEVEL, STRENGTH, TABLE, EXT_TABLE

STATE_BIT

Sets outputs to binary weighted logic states.

General Form

STATE_BIT <output> [<output> ...] = (<expression>);

Parameters

<output> Name of, or variable index to the output pin
<expression> Any expression which can be bitwise matched with the outputs

Use

The state of an output pin is determined by its level and its strength. The *STATE_BIT* command is used to set the level and strength for one or more output pins based on the value of the <expression>. The state of the first pin listed is set according to the first (least-significant-bit) of the expression's value, the state of the second pin listed is set according to second bit of the expression's value, and so on. The level and strength values are set by the bit's value:

Bit Value	State	Level	Strength
0	ZERO	vol_param	rol_param
1	ONE	voh_param	roh_param

Notes

Output pins can be specified by using the output pin name or by an integer variable that contains the *index* of an output pin. Pin and variable names *cannot* be mixed in the same *STATE_BIT* statement. References to outputs must be either all pin names or all variable names. The maximum number of output pins/vars is limited to 16.

Example

```
STATE_BIT Q1 Q2 Q3 Q4 = (internal_reg);
```

In this example, if `internal_reg = 11` (1011 binary) then Q1 (LSB) = ONE, Q2 = ONE, Q3 = ZERO and Q4 (MSB) = ONE.

See Also

REALS, STATE, LEVEL, STRENGTH, TABLE, EXT_TABLE

STEP_OFF

Turns off the SimCode trace mode.

General Form

`STEP_OFF`

Use

The *STEP_OFF* statement turns off the SimCode TRACE mode.

See Also

`STEP_ON`

STEP_ON

Turns on the SimCode trace mode.

General Form

`STEP_ON`

Use

The *STEP_ON* statement turns on the SimCode TRACE mode. This causes the SimCode to display the Program Counter (PC) number and each SimCode instruction before it is executed.

See Also

STEP_OFF

STRENGTH

Sets the strength of the output state.

General Form 1

`STRENGTH <output> [<output> ...] = (<expression>);`

General Form 2

`STRENGTH <output> [<output> ...] = {STRONG}|{HI_IMPEDANCE};`

Parameters

<code><output></code>	Name of, or variable index to the output pin.
<code><expression></code>	Any expression to be used directly as a strength.

Use

The state of an output pin is determined by its level and its strength. Use the *STRENGTH* command to set the strength of one or more output pins.

Value	State	Strength
STRONG	ZERO	rol_param
STRONG	ONE	roh_param
HI_IMPEDANCE	N/A	r3s_param
<expression>	N/A	<expression>

Notes

Output pins can be specified by using the output pin name or by an integer variable that contains the *index* of an output pin. Pin and variable names *cannot* be mixed in the same *STATE* statement. References to outputs must be either all pin names or all variable names.

See Also

REALS, STATE, STATE_BIT, LEVEL

SUPPLY_MIN_MAX

Tests supply pins for min and max supply voltage violations.

General Form

```
SUPPLY_MIN_MAX(<min value>, <max value>);
```

Parameters

<min value>	Minimum recommended power supply voltage.
<max value>	Maximum recommended power supply voltage.

Use

The *SUPPLY_MIN_MAX* function checks the voltage difference between the power and ground pins defined in *PWR_GND_PINS*. If the “WARN flag” is set in the Digital Model Parameters dialog box and the voltage difference (*pwr_param* - *gnd_param*) is less than <min value> or greater than <max value> a warning will be displayed during simulation.

Notes

Databook specifications should be used with this function. *PWR_GND_PINS* must be defined to use this function.

Example

```
SUPPLY_MIN_MAX(4.75, 5.25);
```

See Also

INTEGERS, PWR_GND_PINS

TABLE

Sets output logic states based on truth table.

General Form

```
TABLE <line>
<input> [<input> ...] <output pin> [<output pin> ...]
<input state> [<input state> ...] <output state> [<output state> ...];
```

Parameters

<line>	Variable into which the table line number is placed.

Mixed-Signal Simulation

<input>	Name of the input pin or variable index to the input pin.
<output pin>	Name of the output pin.
<input state>	State of the individual inputs.
<output state>	State of the individual outputs based on input conditions.

Use

The *TABLE* statement operates like a truth table to set the level and strength of the specified outputs. Valid input states are:

0	low (input voltage is $\leq$ <i>vil_param</i>).
1	high (input voltage is $\geq$ <i>vih_param</i>).
X	don't care what input voltage is.

Valid output states are:

L	ZERO (set output level to <i>vol_param</i>)
H	ONE (set output level to <i>voh_param</i>).
Z	UNKNOWN (set output level to <i>v3s_param</i>).

Output state letters can be followed by a colon and a letter to indicate strength:

s	STRONG (set output to <i>rol_param</i> for L and <i>roh_param</i> for H).
z	HI_IMPEDANCE (set output to <i>r3s_param</i>).

If a strength character is *not* specified after an output state then STRONG will be used for L and H states and HI_IMPEDANCE will be used for Z states.

Notes

Each row is tested sequentially from top to bottom until the input conditions are met. The outputs are set for the *first* row to meet the input conditions. The <line> is set to the line number in the table that was used. If no match was made then <line> is set to 0. Input pin and variable names *cannot* be mixed in the same *TABLE* statement. References to inputs must be either all pin names or all variable names.

Example

```
TABLE tblIndex
INA  INB  OUT
0  0  H
0  1  H
1  0  H
1  1  L;
```

This example is representative of 1/4 of a 7400 2-input NAND gate. If input pins INA and INB are both high ($\geq$ *vih_param*), OUT is set to ZERO (*vol_param*) and STRONG (*rol_param*) and tblIndex is set to 4.

See Also

REALS, *STATE*, *STATE_BIT*, *EXT_TABLE*

VALUE

Returns the value of the specified pin.

General Form

```
VALUE(<pin>)
```

Parameters

<pin> Name of or index to a pin.

Use

The *VALUE* function returns a real number that indicates the voltage level of the specified pin.

Example

```
v = (VALUE(D3));
```

VIL_VIH_PERCENT

Sets VIL and VIH values to a percentage of supply voltage.

General Form

```
VIL_VIH_PERCENT (<vil %>, <vih %>);
```

Parameters

<vil %> Percentage of the supply voltage which defines vil.

<vih %> Percentage of the supply voltage which defines vih.

Use

VIL and VIH do not use a min/typ/max array to select their values, but must be declared explicitly for each digital device. The *VIL_VIH_PERCENT* statement sets the VIL and VIH parameters of the device to a percentage of the supply voltage as follows:

$$vil\_param = (pwr\_param - gnd\_param) * <vil \%>$$

$$vih\_param = (pwr\_param - gnd\_param) * <vih \%>$$
Notes

PWR_GND_PINS must be defined to use this function. The % values must be greater than 0 and less than 100. The *vil_param* and *vih_param* values set by *VIL_VIH_PERCENT* are overridden by any values set for “VIL value” and “VIH value” in the Digital Model Parameters dialog box.

Example

```
VIL_VIH_PERCENT(33, 67);
```

See Also

REALS, *PWR_GND_PINS*, *VIL_VIH_VALUE*

VIL_VIH_VALUE

Sets absolute VIL and VIH values.

General Form

```
VIL_VIH_VALUE (<vil>, <vih>);
```

Parameters

<vil> Absolute voltage level which defines vil.

<vih> Absolute voltage level which defines vih.

Use

VIL and VIH do not use a min/typ/max array to select their values, but must be declared explicitly for each digital device. The *VIL_VIH_VALUE* statement sets the VIL and VIH parameters of the device to absolute voltages as follows:

$$vil\_param = \text{<vil>}$$
$$vih\_param = \text{<vih>}$$

Notes

In order to more accurately model the actual switching characteristics of a digital input, VIL and VIH are *not* generally set to their specified databook values. The exception is the case of devices with a specified hysteresis such as the 74LS14. Typically, the hysteresis of a digital device is small, in the order of 100mV, but never 0V.

The *vil_param* and *vih_param* values set by *VIL_VIH_VALUE* are overridden by any values set for “VIL value” and “VIH value” in the Digital Model Parameters dialog box.

Example

```
VIL_VIH_VALUE(1.25, 1.35);
```

See Also

REALS, VIL_VIH_PERCENT

VOL_VOH_MIN

Sets VOH and VOL relative to power and ground.

General Form

```
VOL_VOH_MIN (<vol offset>, <voh offset>, <min voh-vol>);
```

Parameters

<vol offset> Voltage offset which must be applied to ground pin voltage to get vol.

<voh offset> Voltage offset which must be applied to power pin voltage to get voh.

<min voh-vol> Minimum allowed difference between voh and vol.

Use

VOL and VOH do not use a min/typ/max array to select their values, but must be declared explicitly for each digital device. The *VOL_VOH_MIN* statement sets the VOL and VOH parameters of the device as follows:

$$vol\_param = gnd\_param + \text{<vol offset>}$$
$$voh\_param = pwr\_param + \text{<voh offset>}$$

Notes

In order to more accurately model the actual characteristics of a digital output, VOH is *not* generally set to its specified databook value. The reason for this deviation is that databook values for VOH are specified for maximum IOH load. In digital SimCode, VOL and VOH represent an *unloaded* output voltage.

PWR_GND_PINS must be defined to use this function. The *vol_param* and *voh_param* values set by *VOL_VOH_MIN* are overridden by any values set for “VOL value” and “VOH value” in the Digital Model Parameters dialog box. These are offset values rather than absolute voltages. The <voh offset> is negative so that when added to *pwr_param*, the resulting VOH will not be greater than *pwr_param*. If the difference between the resulting *vol_param* and *voh_param* is less than <min voh-vol>, then *vol_param* will be set to the value of *gnd_param* and *voh_param* will be set to *gnd_param* + <min voh-vol>.

Example

```
VOL_VOH_MIN(0.2, -0.4, 0.1);
```

In this example:

- 1 If *gnd_param* = 0V and *pwr_param* = 5.0V, then
 vol_param = 0.2V and *voh_param* = 4.6V
- 2 If *gnd_param* = 0V and *pwr_param* = 0.5V, then
 vol_param = 0.0V and *voh_param* = 0.1V

See Also

REALS, *PWR_GND_PINS*

WHILE ... DO

Conditionally controls looping in the SimCode.

General Form

```
WHILE (<expression>) DO BEGIN ... END;
```

Parameters

<expression> Any expression that can be evaluated as true or false

Use

The *WHILE ... DO* statement is used to loop through a section of SimCode until <expression> evaluates to false.

Notes

Program flow will remain in a loop between the BEGIN and END statements until <expression> evaluates to false, then program flow resumes after the END statement.

Examples

```
i = 1;
WHILE (i <= 5) DO
  BEGIN
    data[i] = data[i + 1];
    i = i + 1;
```

END;

See Also

IF ... THEN

WIDTH

Tests inputs for minimum pulse width violations.

General Form

```
WIDTH(<input> [<input>...] {TWL=<time>}|{TWH=<time>} ["<message>"];
```

Parameters

<input>	Name of or variable index to the input pin under test.
<i>TWL</i>	Width of a low going pulse.
<i>TWH</i>	Width of a high going pulse.
<message>	Text string that will be displayed if a warning occurs.

Use

The *WIDTH* function compares the pulse width on each <ipin> to the specified test *WIDTH* times. A low level test time is specified using *TWL*=<time> while a high level test time is specified using *TWH*=<time>. If the compare time is less than the specified <time> a WARNING will be displayed. An optional <message> string can be included in the *WIDTH* statement which will be output if a WARNING is displayed.

Notes

Databook specifications should be used with this function. The input pins can be input pin names and/or integer variables that contain an index value to an input pin. Pin names and variables *can* be mixed in the same *WIDTH* statement.

Examples

```
WIDTH(CLK TWL=clk_twl TWH=clk_twh "CLK");
WIDTH(PRE CLR TWL= pre_clr_twl "PRE or CLR");
```

WIDTH_TIME

Returns last pulse width encountered on specified pin.

General Form

```
WIDTH_TIME(<input>)
```

Parameters

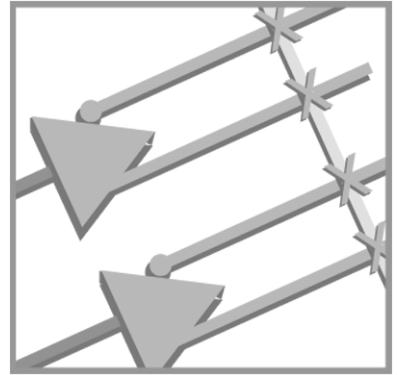
<input>	Name of or index to an input pin.

Use

This function returns a real value that indicates the last pulse width encountered on the specified <input>.

Example

```
PW = (WIDTH_TIME(CP2));
```

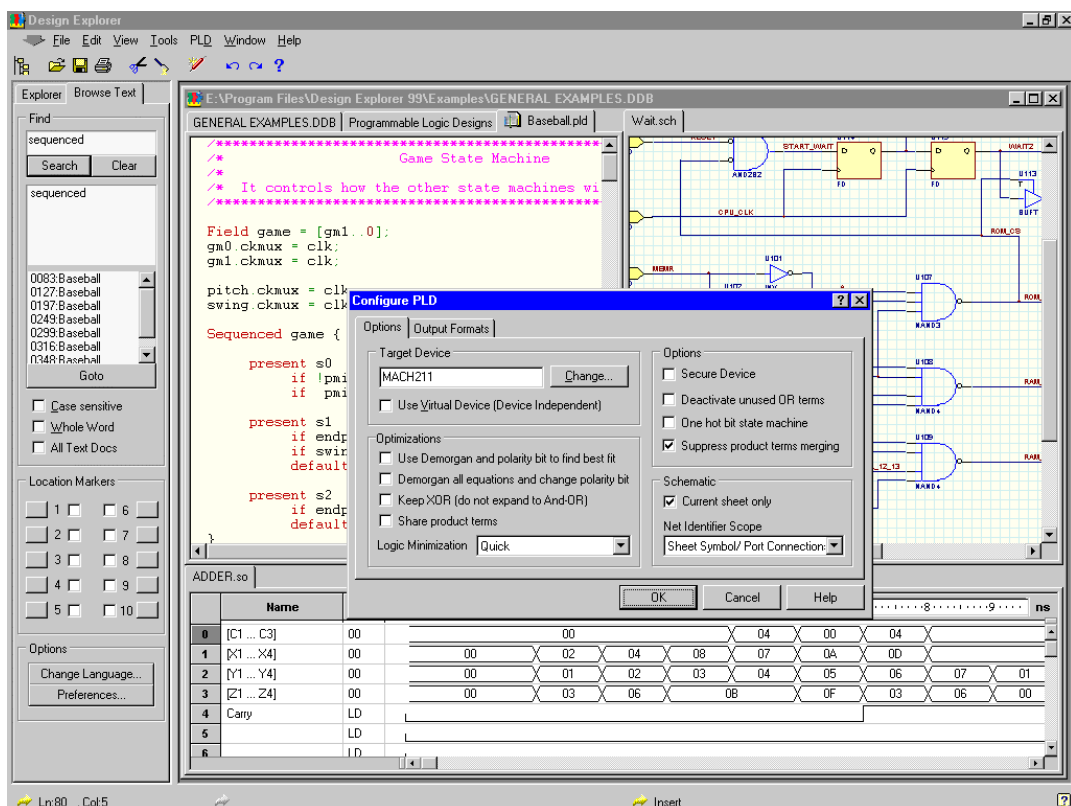


Section 5 _____

PLD Design

<i>PLD Design – Feature Highlights</i>	313
<i>An Overview of the PLD Design Process</i>	316
<i>Schematic-based PLD Design Entry</i>	319
<i>CUPL HDL Design Entry</i>	336
<i>Compiling the Programmable Logic Design</i>	351
<i>Simulating the Programmable Logic Design</i>	359
<i>Sample PLD Design Session</i>	389
<i>PLD Design Examples</i>	411

PLD Design – Feature Highlights



This section of the Protel Designer's Handbook will guide you through the process of designing for programmable logic. As you read this section you will find all the information you need to design a PLD; either as a schematic, or using the CUPL™ Hardware Description Language.

The CUPL HDL allows you to design a PLD using Boolean equations, state machines and truth tables. Use the sample design session and the design examples to guide you through the design process.

Extensive device libraries are included, supporting devices from all the major manufacturers of programmable logic. The CUPL HDL is manufacture-independent, providing you with both design and packaging freedom.

Schematic-based PLD Design

Capture your PLD as a schematic, from the special PLD Symbols library. This library includes all the functional blocks required to create a schematic-based PLD design. Your schematic-based PLD design can be anything from a single sheet, through to a multi-sheet hierarchical design of any depth. PLD Symbols.lib is in the Pld.Ddb Library Database.

Designing a PLD with CUPL

CUPL HDL logic description files and simulation listing files are entered using the integrated, syntax-aware Text Editor. The PLD design tools incorporates integrated error reporting. When an error is detected during compilation or simulation it is automatically displayed and highlighted, along with a description of the error condition.

The PLD Compiler

The PLD Compiler contains the fastest and most powerful minimizer offered for programmable logic equation reduction, with four levels of minimization. The Compiler simplifies Boolean expressions by the distributive property and DeMorgan's Theorem.

The Compiler produces an industry standard JEDEC file. This is a download file format that is compatible with any logic programmer that supports the JEDEC format. Devices from all the major manufacturers of programmable logic are supported, providing both design and packaging freedom.

The PLD Simulator

Use the PLD Simulator to simulate your design before it is put into production. Create a simulation listing file which describes the expected functionality of the PLD in terms of input and output values. The Simulator compares the expected values to actual values calculated during compilation. Test vectors verified by the Simulator can be included in the JEDEC file and downloaded to the logic programmer.

Viewing the Simulation Waveforms

Simulation results can be examined with the Waveform Editor. This Editor reads the simulation listing file (*filename.SO*) and displays the waveform in a "spreadsheet" style window.

CUPL High-level Language

The PLD Compiler supports the CUPL Hardware Description Language. Throughout this guide it is referred to as ‘CUPL’ or ‘the CUPL language’.

To save design time, the CUPL language contains expression substitution for equations and shorthand notation for lists, address ranges, and bit fields.

State machine syntax provides a means of implementing any synchronous application using either Mealy or Moore state machine models. Truth table syntax provides a way to clearly express certain logic descriptions.

A comprehensive CUPL Language Reference section is included in the on-line Help.

Universal Device Support

Devices from all the major manufacturers of programmable logic are supported, giving two distinct advantages. The first is that you only need to learn one development environment and language. You can design anything from a simple address decoder, destined for a GAL16V8; through to a proprietary design destined for a Xilinx 5000 series device. The second advantage is that you can package the same functional logic into physically different parts, giving you the freedom to choose the device manufacturer.

An Overview of the PLD Design Process

This chapter gives an overview of the process of PLD design. It outlines the sequence you follow – to go from a design concept, through to a programmed device.

Entering the Design

Your design can be entered either as a schematic, or in the CUPL HDL. The approach you choose will depend on the nature of the design, and your design preferences.

Schematic-based Design

To create a schematic-based PLD design use the components from the special schematic-to-PLD library, PLD Symbols.lib, which is in the \Program Files\Design Explorer 99 SE\Library\Sch\Pld.ddb Library Database.

Once the logic design is complete you select the target device, configure the compiler, and enable the output options in the Configure dialog. When the compiler detects that the design is schematic-based it automatically translates the design into CUPL HDL code. This CUPL code is then compiled in the normal way.

◆ The pins from the physical target device are mapped to the internal logic using the special IPAD and OPAD components.

CUPL HDL Design

The CUPL Hardware Description Language allows you to design a PLD using a variety of methods, including Boolean equations, state machines and truth tables. To save design time, the CUPL language contains expression substitution for equations and shorthand notation for lists, address ranges, and bit fields.

State machine syntax provides a means of implementing any synchronous application using either Mealy or Moore state machine models, and truth table syntax provides a way to clearly express certain logic descriptions.

◆ A comprehensive CUPL Language Reference section is included in the on-line Help.

Compiling the Design

The following steps are carried out to compile the CUPL code.

Language Preprocessing

The input file is scanned, processing the preprocessor directives, such as \$DEFINE. An intermediate file is generated with all preprocessor directives expanded.

Parsing And State Machine Translation

The intermediate file is parsed using a table driven parser, generating a symbol table and expanded equations. It expands state machine, truth table, and user-defined function syntax into Boolean equations, and also performs simple logic reduction while processing range statements.

Device Compatibility Checking and DeMorganizing

The pinouts selected in the PLD source file are checked against the allowable pinouts in the device library (.DL) file, verifying that all variables in the source file are used correctly based on the architecture of the device.

This module also performs DeMorgan's theorem on an output variable when the polarity of the variable does not match the polarity of the pin in the device. This allows active high designs to be implemented in active low devices, and vice-versa. Also at this stage of compiling the first level of minimization is performed. It will eliminate 0's and 1's, redundant terms, and product terms that are "contained" within other product terms. This pass builds the final symbol table, containing device model links and a bitmap representation of the logic design.

Logic Minimizing

The Logic Minimizing module executes logic minimization algorithms on the bitmap representation of the logic generated in the previous pass. It processes only the equations for which reduction has been requested.

The minimizer performs the selected minimization algorithm on the source file equations. The minimizer algorithms are; Quick, Quine-McCluskey, Presto, and Espresso. The best algorithm for PAL architectures is Quine-McCluskey, and the best algorithm for PLA architectures is Espresso (it does very well with product term sharing). In general, Espresso is fast and gives almost the same results as Quine-McCluskey.

Fitter And Output File Generation

The fitter module fits the design into the desired device. It uses built-in algorithms, or calls a third party fitter to achieve a device fit. The fitter module manipulates the

macrocells and multiplexers of complex devices to fit the Boolean logic into the device. If the design cannot fit into the device an error message is displayed. The fitter module also creates all the output files specified in the Configure dialog.

The fitter module determines if the design fits the target device architecture and builds a fuse map. The fuse map and symbol table are used to generate the documentation and JEDEC files.

Simulating the Design

To simulate your design you must first create a simulation input file (*filename.SI*), and compile the design with the ABS output option enabled.

The simulation input file describes the expected functionality of the device in terms of input and output values. When you run a simulation the expected values are compared to the actual values calculated during compilation (these are in the ABS file). Test vectors verified by the Simulator can also be included in the JEDEC file and downloaded to the logic programmer.

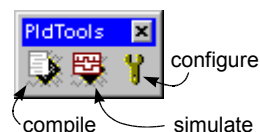
The simulator performs unit-delay simulation that mostly reflects how the design will behave in the actual device. Certain device idiosyncrasies and internal delays are not simulated. However, the simulator will show race conditions and true “Don’t Care” usage.

Where to Set-up and Run the PLD Compiler

Device selection, compiler options and output options are configured in the Configure PLD dialog. Select **PLD » Configure** from the Text Editor or Schematic menus to display this dialog. You can also configure, compile and simulate from the PLDTools toolbar. Select **View » Toolbars » PLD Toolbar** from the menus to display the toolbar.

The compiler is launched by selecting **PLD » Compile** from the menus, or by clicking on the Compile button. The simulator is launched by selecting **PLD » Simulate** from the menus, or by clicking on the Simulate button.

Refer to the chapter, *Compiling the Programmable Logic Design*, for more details on configuring the compiler.



Schematic-based PLD Design Entry

Schematic-based design for programmable logic is done in the same way that you capture the schematic for the PCB. The design can be single or multi-sheet, with an unlimited depth to the design hierarchy. The only difference is that you must use symbols from the special PLD Symbols library.

When you compile a schematic-based programmable logic design that is targeted for a PLD, the design is first translated from a schematic to a CUPL compiler source file. The resulting CUPL file is then compiled to produce the selected output files.

Using the Symbol Library

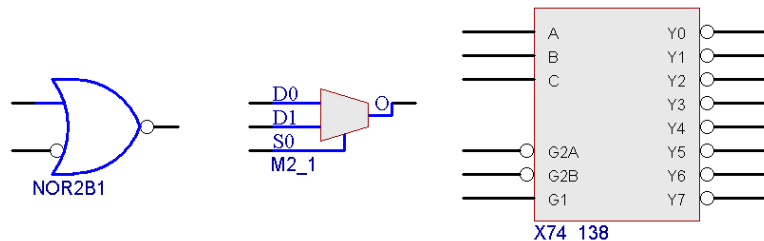
There is a special library of symbols that is used for designing programmable logic, PLD Symbols.lib, which is in the \Program Files\Design Explorer 99 SE\Library\Sch\Pld.ddb Library Database.

This library covers all the standard functional groups, such as buffers, comparators, flip-flops, and so on.

It is important that you are familiar with the architecture of the device that your design is targeted for, to ensure that you only use symbols that can be implemented in that device.

Symbol Naming Conventions

The symbols are named in one of two ways. Most are named using a set of descriptors, with different letters to indicate the various functions supported by that symbol. The others are named using standard TTL 74 series logic names. These are preceded with the letter X, for example X74_138.



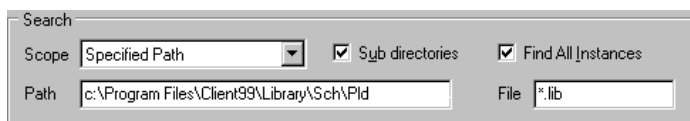
A 2 input NOR, a 2-to-1 multiplexer, and a 3-to-8 decoder from the PLD Symbols library

Finding Symbols in the Library

There are over 360 symbols in the programmable logic Symbols library. They are all listed, with a description, in the following topic, *Symbols Grouped by Functional Category*.

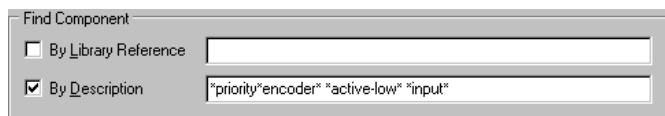
You can also use the Schematic Editor's Find feature to search for a symbol. Select **Tools » Find Component** from the menus to display the Find Component dialog. Each symbol in the programmable logic symbol library includes a comprehensive description, making it easy to search by description. For example, perhaps you would like to know if there is a symbol for a priority encoder with active-low inputs in the library.

Before you begin the search make sure that the Search Scope and Search Path are set up correctly. The figure shows how the \PLD folder has been included in the path to restrict the search to only the programmable logic symbol libraries.



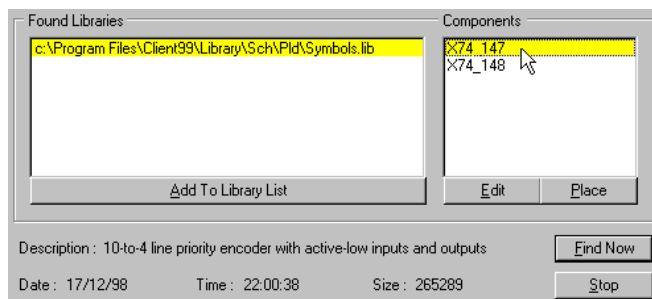
Set the search scope and path to restrict the search

Rather than trying to type the search string in exactly as it might be written in the library, type in each word (or sets of words) that you want to search for, enclosed by the wildcard characters.



Searching for a component by description

Any libraries that include components with these words in their description will be listed in the Found Libraries section of the dialog. Click on a component in the list to display a description of the component at the bottom of the dialog.



Use the Find feature to search for a symbol

Symbols Grouped by Functional Category

Arithmetic Functions

There are three types of arithmetic functions: accumulators (ACC), adders (ADD), and adder/subtractors (ADSU).

ACC1	1-Bit loadable cascadable accumulator with carry-in, carry-out, and synchronous reset
ACC16	16-Bit loadable cascadable accumulator with carry-in, carry-out, and synchronous reset
ACC4	4-Bit loadable cascadable accumulator with carry-in, carry-out, and synchronous reset
ACC8	8-Bit loadable cascadable accumulator with carry-in, carry-out, and synchronous reset
ADD1	1-Bit full adder with carry-in and carry-out
ADD16	16-Bit full adder with carry-in and carry-out
ADD4	4-Bit full adder with carry-in and carry-out
ADD8	8-Bit full adder with carry-in and carry-out
ADSU1	1-Bit cascadable adder/subtractor with carry-in and carry-out
ADSU16	16-Bit cascadable adder/subtractor with carry-in and carry-out
ADSU4	4-Bit cascadable adder/subtractor with carry-in and carry-out
ADSU8	8-Bit cascadable adder/subtractor with carry-in, carry-out and Overflow
X74_280	9-Bit Odd/Even Parity generator/checker
X74_283	4-Bit full adder with carry-in and carry-out

Buffers

Buffers route high fan-out signals, 3-state signals, and clocks inside a PLD device.

BUF	General purpose buffers
BUF16	General purpose buffers
BUF4	General purpose buffers
BUF8	General purpose buffers
BUFE	Internal 3-State buffers
BUFE16	Internal 3-State buffers
BUFE4	Internal 3-State buffers
BUFE8	Internal 3-State buffers
BUFG	Global clock buffer
BUFGP	Primary Global buffer for driving clocks or longlines (Four per PLD device)
BUFGS	Secondary buffer for driving clocks or longlines (Four per PLD device)
BUFT	Internal 3-State buffers
BUFT16	Internal 3-State buffers
BUFT4	Internal 3-State buffers

BUFT8	Internal 3-State buffers

Comparators

There are two types of comparators, identity (COMP) and magnitude (COMPM).

COMP16	16-Bit Identity comparator
COMP2	2-Bit Identity comparator
COMP4	4-Bit Identity comparator
COMP8	8-Bit Identity Comparator
COMPM16	16-Bit Magnitude Comparator
COMPM2	2-Bit Magnitude Comparator
COMPM4	4-Bit Magnitude Comparator
COMPM8	8-Bit Magnitude Comparator
COMPMC16	16-Bit Magnitude Comparator
COMPMC8	8-Bit Magnitude Comparator
X74_518	8-Bit identity comparator with active-low enable
X74_521	8-Bit identity comparator with active-low enable and output
X74_L85	4-Bit expandable magnitude comparator

Counters

There are six types of counters, with a variety of synchronous and asynchronous inputs. Use the naming information shown below to determine exactly what functions each counter includes.

CB16CE	16-Bit cascadable binary counter with clock enable and asynchronous Clear
CB16CLE	16-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CB16CLED	16-Bit loadable cascadable bi-directional binary counter with clock enable and asynchronous Clear
CB16RE	16-Bit cascadable binary counter with clock enable and synchronous reset
CB2CE	2-Bit cascadable binary counter with clock enable and asynchronous Clear
CB2CLE	2-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CB2CLED	2-Bit loadable cascadable bi-directional binary counter with clock enable and asynchronous Clear
CB2RE	2-Bit cascadable binary counter with clock enable and synchronous reset
CB4CE	4-Bit cascadable binary counter with clock enable and asynchronous Clear
CB4CLE	4-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CB4CLED	4-Bit loadable cascadable bi-directional binary counter with clock enable and asynchronous Clear
CB4RE	4-Bit cascadable binary counter with clock enable and synchronous reset
CB8CE	8-Bit cascadable binary counter with clock enable and asynchronous Clear

CB8CLE	8-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CB8CLED	8-Bit loadable cascadable bi-directional binary counter with clock enable and asynchronous Clear
CB8RE	8-Bit cascadable binary counter with clock enable and synchronous reset
CC16CE	16-Bit cascadable binary counter with clock enable and asynchronous Clear
CC16CLE	16-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CC16CLED	16-Bit loadable cascadable bidirectional binary counter with clock enable and asynchronous Clear
CC16RE	16-Bit cascadable binary counter with clock enable and synchronous reset
CC8CE	8-Bit cascadable binary counter with clock enable and asynchronous Clear
CC8CLE	8-Bit loadable cascadable binary counter with clock enable and asynchronous Clear
CC8CLED	8-Bit loadable cascadable bidirectional binary counter with clock enable and asynchronous Clear
CC8RE	8-Bit cascadable binary counter with clock enable and synchronous reset
CD4CE	4-Bit cascadable BCD counter with clock enable and asynchronous Clear
CD4CLE	4-Bit loadable cascadable BCD counter with clock enable and asynchronous Clear
CD4RE	4-Bit cascadable BCD counter with clock enable and synchronous reset
CD4RLE	4-Bit loadable cascadable BCD counter with clock enable and synchronous reset
CJ4CE	4-Bit Johnson counter with clock enable and asynchronous Clear
CJ4RE	4-Bit Johnson counter with clock enable and synchronous reset
CJ5CE	5-Bit Johnson counter with clock enable and asynchronous Clear
CJ5RE	5-Bit Johnson counter with clock enable and synchronous reset
CJ8CE	8-Bit Johnson counter with clock enable and asynchronous Clear
CJ8RE	8-Bit Johnson counter with clock enable and synchronous reset
CR16CE	16-Bit Negative-Edge binary ripple counter with clock enable and asynchronous Clear
CR8CE	8-Bit Negative-Edge binary ripple counter with clock enable and asynchronous Clear
X74_160	4-Bit BCD counter with parallel and trickle enables, active-low load enable and synchronous reset
X74_161	4-Bit binary counter with parallel and trickle enables, active-low load enable and synchronous reset
X74_162	4-Bit binary counter with parallel and trickle enables, active-low load enable and synchronous reset
X74_163	4-Bit binary counter with parallel and trickle enables, active-low load enable and synchronous reset

X74_168	4-Bit BCD bi-directional counter with parallel and trickle clock enables and active-low load enable
X74_390	4-Bit BCD/Bi-Quinary counter with negative-edge clock and asynchronous clear

Data Registers

The 3 types of data registers function exactly the same as the equivalent TTL components.

X74_174	6-Bit Data register with active-low asynchronous clear
X74_273	8-Bit Data register with active-low asynchronous clear
X74_377	8-Bit Data register with active-low clock enable

Decoders

Decoders with an enable can be used as multiplexers.

D2_4E	2-to-4 Decode/Demultiplexer with enable
D3_8E	3-to-8 Decode/Demultiplexer with enable
D4_16E	4-to-16 Decode/Demultiplexer with enable
X74_42	4-line to 10-line BCD-to-Decimal decode with active-low outputs
X74_138	3-to-8 line decoder/demultiplexer with active-low outputs and three enables
X74_139	2-to-4 line decoder/demultiplexer with active-low outputs and active-low enable
X74_154	4-to-16 line decoder/demultiplexer with active-low outputs and two enables

Encoders

There is a 10 to 4-line BCD encoder, and an 8 to 3-line binary encoder.

X74_147	10-to-4 line priority encoder with active-low inputs and outputs
X74_148	8-to-3 line cascadable priority encoder with active-low inputs and outputs

Flip-Flops

There are three types of flip-flops (D, J-K, toggle) with various synchronous and asynchronous inputs. Some of the flip-flops have inverted clock inputs, and/or the ability to be set in response to a global set/reset, rather than reset.

FD	D Flip-Flop
FD_1	D Flip-Flop with negative-edge clock
FD16CE	16-Bit data register with clock enable and asynchronous clear
FD16RE	16-Bit data register with clock enable and synchronous reset
FD4CE	4-Bit data register with clock enable and asynchronous clear
FD4RE	4-Bit data register with clock enable and synchronous reset
FD8CE	8-Bit data register with clock enable and asynchronous clear
FD8RE	8-Bit data register with clock enable and synchronous reset

FDC	D Flip-Flop with asynchronous clear
FDC_1	D Flip-Flop with negative-edge clock and asynchronous clear
FDCE	D Flip-Flop with clock enable and asynchronous clear
FDCE_1	D Flip-Flop with negative-edge clock, clock enable and asynchronous clear
FDCP	D Flip-Flop with asynchronous preset and clear
FDCPE	D Flip-Flop with clock enable and asynchronous preset and clear
FDCS	D Flip-Flop with asynchronous set and clear
FDP	D Flip-Flop with asynchronous preset
FDP_1	D Flip-Flop with negative-edge clock and asynchronous preset
FDPE	D Flip-Flop with clock enable and asynchronous preset
FDPE_1	D Flip-Flop with negative-edge clock, clock enable and asynchronous preset
FDR	D Flip-Flop with synchronous reset
FDRE	D Flip-Flop with clock enable and synchronous reset
FDRS	D Flip-Flop with synchronous reset and synchronous set
FDRSE	D Flip-Flop with synchronous reset and set and clock enable
FDS	D Flip-Flop with synchronous set
FDSE	D Flip-Flop with clock enable and synchronous set
FDSR	D Flip-Flop with synchronous set and synchronous reset
FDSRE	D Flip-Flop with synchronous set and reset and clock enable
FJKC	J-K Flip-Flop with asynchronous clear
FJKCE	J-K Flip-Flop with clock enable and asynchronous clear
FJKCP	J-K Flip-Flop with asynchronous clear and preset
FJKCPE	J-K Flip-Flop with clock enable and asynchronous clear and preset
FJKCS	J-K Flip-Flop with asynchronous clear and set
FJKP	J-K Flip-Flop with asynchronous preset
FJKPE	J-K Flip-Flop with clock enable and asynchronous preset
FJKRSE	J-K Flip-Flop with clock enable and synchronous reset and set
FJKSRE	J-K Flip-Flop with clock enable and synchronous reset and set
FTC	Toggle Flip-Flop with toggle enable and asynchronous clear
FTCE	Toggle Flip-Flop with toggle enable, clock enable and asynchronous clear
FTCLE	Toggle/loadable Flip-Flop with toggle enable, clock enable and asynchronous clear
FTCP	Toggle Flip-Flop with toggle enable and asynchronous clear and preset
FTCPE	Toggle Flip-Flop with toggle enable, clock enable and asynchronous clear and preset
FTCPLE	Toggle/loadable Flip-Flop with toggle enable, clock enable and asynchronous clear and preset
FTCS	Toggle Flip-Flop with toggle enable and asynchronous clear and set
FTP	Toggle Flip-Flop with toggle enable and asynchronous preset
FTPE	Toggle Flip-Flop with toggle enable, clock enable and asynchronous preset

FTPLE	Toggle/loadable Flip-Flop with toggle enable, clock enable and asynchronous preset
FTRSE	Toggle Flip-Flop with toggle enable, clock enable and synchronous reset and set
FTRSLE	Toggle/loadable Flip-Flop with toggle enable, clock enable and synchronous reset and set
FTSRE	Toggle Flip-Flop with toggle enable, clock enable and synchronous reset and set
FTSRLE	Toggle/loadable Flip-Flop with toggle enable, clock enable and synchronous reset and set

Input/Output Flip-Flops

Input/Output flip-flops are configured in IOBs. They include flip-flops whose outputs are enabled by 3-state buffers, flip-flops that can be set upon global set/reset rather than reset, and flip-flops with inverted clock inputs.

IFD	Input D Flip-Flop
IFD_1	Input D Flip-Flop with inverted clock
IFD16	16 x Input D Flip-Flops
IFD4	4 x Input D Flip-Flops
IFD8	8 x Input D Flip-Flops
OFD	Output D Flip-Flop
OFD_1	Output D Flip-Flop with inverted clock
OFD16	16 x Output D Flip-Flop
OFD4	4 x Output D Flip-Flop
OFD8	8 x Output D Flip-Flop
OFDE	Output D Flip-Flop with active-high enable output buffer
OFDE_1	Output D Flip-Flop with active-high enable output buffer and inverted clock
OFDE16	16 x Output D Flip-Flop with active-high enable output buffer
OFDE4	4 x Output D Flip-Flop with active-high enable output buffer
OFDE8	8 x Output D Flip-Flop with active-high enable output buffer
OFDT	Output D Flip-Flop with active-high 3-state and active-low enable output buffer
OFDT_1	Output D Flip-Flop with active-high 3-state and active-low enable output buffer and inverted clock
OFDT16	16 x Output D Flip-Flop with active-high 3-state and active-low enable output buffer
OFDT4	4 x Output D Flip-Flop with active-high 3-state and active-low enable output buffer
OFDT8	8 x Output D Flip-Flop with active-high 3-state and active-low enable output buffer

Input/Output Buffers

Input and output buffers, with various output enable options.

IBUF	Input buffer
IBUF16	16 x Input buffers
IBUF4	4 x Input buffers
IBUF8	8 x Input buffers
OBUF	Output buffer
OBUF16	16 x Output buffer
OBUF4	4 x Output buffer
OBUF8	8 x Output buffer
OBUE	3-state Output buffer with active-high output enable
OBUE16	16 x 3-state Output buffer with active-high output enable
OBUE4	4 x 3-state Output buffer with active-high output enable
OBUE8	8 x 3-state Output buffer with active-high output enable
OBUT	3-state Output buffer with active-low output enable
OBUT16	16 x 3-state Output buffer with active-low output enable
OBUT4	4 x 3-state Output buffer with active-low output enable
OBUT8	8 x 3-state Output buffer with active-low output enable

Input/Output Pads

Input, Output, and Input/Output pads are used to connect to the device pins.

IOPAD	Input/Output Pad
IOPAD16	16 x Input Pads
IOPAD4	4 x Input/Output Pads
IOPAD8	8 x Input/Output Pads
IPAD	Input Pad
IPAD16	16 x Input Pads
IPAD4	4 x Input Pads
IPAD8	8 x Input Pads
OPAD	Output Pad
OPAD16	16 x Output Pads
OPAD4	4 x Output Pads
OPAD8	8 x Output Pads

Input Latches

Input latches are used to hold transient data entering the device.

ILD	Input transparent Data Latch
ILD_1	Input transparent Data Latch with inverted gate
ILD16	16 x Input Transparent Data Latches
ILD4	4 x Input Transparent Data Latches
ILD8	8 x Input Transparent Data Latches

Inverters

Logic inverters.

INV	Inverter
INV16	16 x Inverters
INV4	4 x Inverters
INV8	8 x Inverters

Latches

Transparent data latches.

LD	Transparent Data Latch
LD_1	Transparent Data Latch with inverted gate
LD16	16 x Transparent Data Latch
LD4	4 x Transparent Data Latch
LD8	8 x Transparent Data Latch

Logic Primitives

Combinatorial logic gates that implement the basic Boolean functions. Up to 5 inputs with a combination of inverted and non-inverted inputs, and up to 9 inputs non-inverted.

AND2	2-Input and gate
AND2B1	2-Input and gate with one input inverted
AND2B2	2-Input and gate with two inputs inverted
AND3	3-Input AND gate
AND3B1	3-Input AND gate with one input inverted
AND3B2	3-Input AND gate with two inputs inverted
AND3B3	3-Input AND gate with three inputs inverted
AND4	4-Input AND gate
AND4B1	4-Input AND gate with one input inverted
AND4B2	4-Input AND gate with two inputs inverted
AND4B3	4-Input AND gate with three inputs inverted
AND4B4	4-Input AND gate with four inputs inverted

AND5	5-Input AND gate
AND5B1	5-Input AND gate with one input inverted
AND5B2	5-Input AND gate with two inputs inverted
AND5B3	5-Input AND gate with three inputs inverted
AND5B4	5-Input AND gate with four inputs inverted
AND5B5	5-Input AND gate with five inputs inverted
AND6	6-Input AND gate
AND7	7-Input AND gate
AND8	8-Input AND gate
AND9	9-Input AND gate
NAND2	2-Input NAND gate
NAND2B1	2-Input NAND gate with one input inverted
NAND2B2	2-Input NAND gate with two inputs inverted
NAND3	3-Input NAND gate
NAND3B1	3-Input NAND gate with one input inverted
NAND3B2	3-Input NAND gate with two inputs inverted
NAND3B3	3-Input NAND gate with three inputs inverted
NAND4	4-Input NAND gate
NAND4B1	4-Input NAND gate with one input inverted
NAND4B2	4-Input NAND gate with two inputs inverted
NAND4B3	4-Input NAND gate with three inputs inverted
NAND4B4	4-Input NAND gate with four inputs inverted
NAND5	5-Input NAND gate
NAND5B1	5-Input NAND gate with one input inverted
NAND5B2	5-Input NAND gate with two inputs inverted
NAND5B3	5-Input NAND gate with three inputs inverted
NAND5B4	5-Input NAND gate with four inputs inverted
NAND5B5	5-Input NAND gate with five inputs inverted
NAND6	6-Input NAND gate
NAND7	7-Input NAND gate
NAND8	8-Input NAND gate
NAND9	9-Input NAND gate
NOR2	2-Input NOR gate
NOR2B1	2-Input NOR gate with one input inverted
NOR2B2	2-Input NOR gate with two inputs inverted
NOR3	3-Input NOR gate
NOR3B1	3-Input NOR gate with one input inverted
NOR3B2	3-Input NOR gate with two inputs inverted
NOR3B3	3-Input NOR gate with three inputs inverted

NOR4	4-Input NOR gate
NOR4B1	4-Input NOR gate with one input inverted
NOR4B2	4-Input NOR gate with two inputs inverted
NOR4B3	4-Input NOR gate with three inputs inverted
NOR4B4	4-Input NOR gate with four inputs inverted
NOR5	5-Input NOR gate
NOR5B1	5-Input NOR gate with one input inverted
NOR5B2	5-Input NOR gate with two inputs inverted
NOR5B3	5-Input NOR gate with three inputs inverted
NOR5B4	5-Input NOR gate with four inputs inverted
NOR5B5	5-Input NOR gate with five inputs inverted
NOR6	6-Input NOR gate
NOR7	7-Input NOR gate
NOR8	8-Input NOR gate
NOR9	9-Input NOR gate
OR2	2-Input OR gate
OR2B1	2-Input OR gate with one input inverted
OR2B2	2-Input OR gate with two inputs inverted
OR3	3-Input OR gate
OR3B1	3-Input OR gate with one input inverted
OR3B2	3-Input OR gate with two inputs inverted
OR3B3	3-Input OR gate with three inputs inverted
OR4	4-Input OR gate
OR4B1	4-Input OR gate with one input inverted
OR4B2	4-Input OR gate with two inputs inverted
OR4B3	4-Input OR gate with three inputs inverted
OR4B4	4-Input OR gate with four inputs inverted
OR5	5-Input OR gate
OR5B1	5-Input OR gate with one input inverted
OR5B2	5-Input OR gate with two inputs inverted
OR5B3	5-Input OR gate with three inputs inverted
OR5B4	5-Input OR gate with four inputs inverted
OR5B5	5-Input OR gate with five inputs inverted
OR6	6-Input OR gate
OR7	7-Input OR gate
OR8	8-Input OR gate
OR9	9-Input OR gate
SOP3	3-Input sum-of-products function
SOP3B1A	3-Input sum-of-products with one AND input inverted

SOP3B1B	3-Input sum-of-products function with one OR input inverted
SOP3B2A	3-Input sum-of-products with two AND inputs inverted
SOP3B2B	3-Input sum-of-products function with one OR and one AND input inverted
SOP3B3	3-Input sum-of-products function with all inputs inverted
SOP4	4-Input sum-of-products function
SOP4B1	4-Input sum-of-products with one AND input inverted
SOP4B2A	4-Input sum-of-products function with both inputs on one AND gate inverted
SOP4B2B	4-Input sum-of-products with one AND input on each AND gate inverted
SOP4B3	4-Input sum-of-products function with three inputs inverted
SOP4B4	4-Input sum-of-products function with all inputs inverted
XNOR2	2-input XNOR Gates with non-inverted inputs
XNOR3	3-input XNOR Gates with non-inverted inputs
XNOR4	4-input XNOR Gates with non-inverted inputs
XNOR5	5-input XNOR Gates with non-inverted inputs
XNOR6	6-input XNOR Gates with non-inverted inputs
XNOR7	7-input XNOR Gates with non-inverted inputs
XNOR8	8-input XNOR Gates with non-inverted inputs
XNOR9	9-input XNOR Gates with non-inverted inputs
XOR2	2-input XOR Gates with non-inverted inputs
XOR3	3-input XOR Gates with non-inverted inputs
XOR4	4-input XOR Gates with non-inverted inputs
XOR5	5-input XOR Gates with non-inverted inputs
XOR6	6-input XOR Gates with non-inverted inputs
XOR7	7-input XOR Gates with non-inverted inputs
XOR8	8-input XOR Gates with non-inverted inputs
XOR9	9-input XOR Gates with non-inverted inputs

Multiplexers

Up to 16 input multiplexers, with various enable and output options.

M16_1E	16-to-1 Multiplexer with enable
M2_1	2-to-1 Multiplexer
M2_1B1	2-to-1 Multiplexer with D0 inverted
M2_1B2	2-to-1 Multiplexer with D0 and D1 inverted
M2_1E	2-to-1 Multiplexer with enable
M4_1E	4-to-1 Multiplexer with enable
M8_1E	8-to-1 Multiplexer with enable
X74_150	16-to-1 line multiplexer with active-low enable and output
X74_151	8-to-1 line multiplexer with active-low enable and complementary outputs

X74_152	8-to-1 line multiplexer with active-low output
X74_153	Dual 4-to-1 line multiplexer with active-low enables and common select input
X74_154	4-to-16 line decoder/demultiplexer with active-low outputs and two enables
X74_157	Quadruple 2-to-1 multiplexer with common select and active-low enable
X74_158	Quadruple 2-to-1 multiplexer with common select and active-low enable, and active-low outputs
X74_298	Quadruple 2-input multiplexer with storage and negative-edge clock
X74_352	Dual 4-to-1 multiplexer with active-low enables and outputs

Shift Registers

Various size shift registers, with different enable and clear options.

SR4CE	4-Bit Serial-in Parallel-out shift register with clock enable and asynchronous clear
SR4CLE	4-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR4CLED	4-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR4RE	4-Bit Serial-in Parallel-out shift register with clock enable and synchronous reset
SR4RLE	4-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
SR4RLED	4-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
SR8CE	8-Bit Serial-in Parallel-out shift register with clock enable and asynchronous clear
SR8CLE	8-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR8CLED	8-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR8RE	8-Bit Serial-in Parallel-out shift register with clock enable and synchronous reset
SR8RLE	8-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
SR8RLED	8-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
SR16CE	16-Bit Serial-in Parallel-out shift register with clock enable and asynchronous clear
SR16CLE	16-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR16CLED	16-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and asynchronous clear
SR16RE	16-Bit Serial-in Parallel-out shift register with clock enable and synchronous

	reset
SR16RLE	16-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
SR16RLED	16-Bit loadable Serial/Parallel-in Parallel-out shift register with clock enable and synchronous reset
X74_164	8-Bit serial-in parallel-out shift register with active-low asynchronous clear
X74_165S	8-Bit loadable serial/parallel-in parallel-out shift register with clock enable
X74_194	4-Bit loadable bi-directional serial/parallel-in parallel-out shift register
X74_195	4-Bit loadable serial/parallel-in parallel-out shift register

Shifters

BRLSHFT4	4-Bit barrel shifter
BRLSHFT8	8-Bit barrel shifter

Supply and Pullup Symbols

GND	Ground-Connection Signal Tag
VCC	VCC-Connection signal tag

RAM

RAM16X1	16-Deep by 1-Wide Static RAM

How the Symbol Behavior is Defined

For each of the symbols in the PLD Symbol library, there is a description of its behavior. The descriptions are stored in the Symbol Definition File, PldNet.SDF. The syntax used in this file is described in the file header.

Designing the Internal Logic

The internal logic of a programmable device is designed using standard, digital design methodologies. Only use components from the PLD Symbols library, then wire them up in the normal way.

When you compile a schematic-based programmable logic design that is targeted for a PLD, the design is first translated from a schematic to a CUPL compiler source file. The resulting CUPL file is then compiled to produce the selected output files.

General Design Notes:

- Each symbol in the design must have a unique designator, including the input and output pads. There are no special requirements or restrictions on designators.
- Assigning net names to all internal nets can assist in debugging.
- Internal net names must be unique, and cannot be the negated value of any other net name.

- Buses can be used, they must include the standard bus-format net labels, the range can be either ascending or descending.
- Use symbols that are appropriate for the target device.

Creating a Multi-Sheet Design

Multi-sheet design is done in the same way as single sheet design. The Schematic Editor supports a number of inter-sheet connection methodologies, all of which are available for a programmable logic design. Refer to the chapter, *Multi-Sheet Design and Project Management*, in the *Schematic Capture* section of this Handbook for details on how to use each of the inter-sheet connection methods.

The inter-sheet connection method is specified by the Net Identifier Scope that is used during netlisting. When you are compiling the design with the CUPL compiler you must select the Net Identifier Scope in the Configure PLD dialog (select **PLD » Configure** from the menus).

The simplest way of remembering what each scope does is – Sheet Symbols/Port Connections creates inter-sheet connections “vertically” (between ports and matching sheet entries); whereas the other two options create inter-sheet connections “horizontally” (directly between matching ports and matching net labels). If you plan to use multiple levels of sheet hierarchy in the design you must use the Sheet Symbols/Port Connections option.

Interfacing from Internal Logic to Component Pins

Special input (IPAD), output (OPAD), and input/output (IOPAD) symbols are used to interface from the internal logic to the pins of the target programmable device.

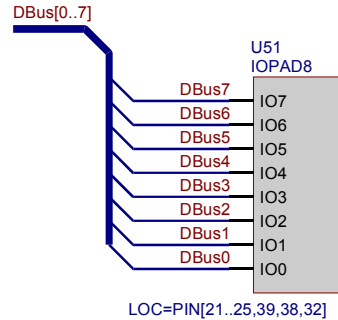
Place one of the appropriate pads at each point that connects to a component pin, and give each PAD symbol a unique designator. The number of the component pin that the PAD connects to is specified in Part Field 1 of the pad symbol, using the syntax:

```
LOC=PIN[component pin_number]
```

Using Multi-pin PAD Components

There are a number of multi-pin IPAD, OPAD and IOPAD components in the library.

net name	Component pin
DBus0	21
DBus1	22
DBus2	23
DBus3	24
DBus4	25
DBus5	39
DBus6	38
DBus7	32



These can be used to connect an internal bus to the component pins. The syntax used for mapping the PAD pins to physical component pins in Part Field 1 is to either specify the pins as a range, separated by 2 dots (ascending or descending); or specify them individually, separated by a comma. The mapping that would apply for the IOPAD8 shown in the figure is shown in the table.

Displaying all the Pin Numbers

The Pin Numbers can be displayed on the sheet by selecting **Simulate » Display Pin LOCs** from the Schematic Editor menus. This is a toggle feature, select it again to hide them.

Compiling a Schematic-based Programmable Logic Design

Once the design is complete the CUPL compiler is configured by selecting **PLD » Configure** from the menus. Refer to the chapter *Compiling the Design* for information on setting the compiler optimizations, options, and output formats.

If the design is a multi-sheet design you must also disable the Current sheet only option and set the Net Identifier Scope option, in the Configure PLD dialog.

When you initiate the Compile process the design is first translated from a schematic to a CUPL compiler source file. The resulting CUPL file (*filename.PLD*) is then compiled to produce the selected output files.

- ◆ When you compile a design using the CUPL compiler you can use the Virtual Device option. This allows you to verify that the design will compile, as well as examine the maximum number of product terms it will require (check the DOC file). The number of product terms is often the reason a larger device must be used – experiment with the various compiler options and optimizations to reduce this number.

CUPL HDL Design Entry

A PLD compiler source file is a text file that contains a logic description of the PLD design in the CUPL Hardware Description Language. This file serves as an input to the PLD compiler, which compiles the design and generates a file suitable for downloading to a device programmer.

Use the following steps to guide you through the process of designing your PLD.

Examine the Design Task: As you consider the design requirements remember that state-machine, Boolean equations, and truth tables are available for the design. Try to determine which type of syntax best suits the design task that is being worked on.

Create the Compiler Source File: Follow the syntax requirements specified in this chapter. Remember to edit the header to reflect the new file that is being created.

Formulate the Equations: Equations must be written in CUPL to specify the logic desired. This can be in Boolean, state-machine, or truth table format.

Choose a Target Device: Make sure that there is a sufficient number of input pins. Check that the number of registered and non-registered output pins is sufficient for the design. Ensure that the device has three-state output control, if needed. Check that the device can adequately handle the number of product terms required. Select the target device in the Configure PLD dialog.

Make Pin Assignments: Assign the inputs and outputs of the design to the pins of the device. Make sure that the device is being used properly by consulting the reference material available from the manufacturer.

Ready to Compile: Decide which file formats will be needed for downloading and simulation. A choice of four minimizers are available. You are now ready to compile your design. Refer to the chapter *Compiling the Design* for information on how to set up the Configure PLD dialog.

Overview of CUPL Source File Syntax

The creation of Compiler source file should adhere to a general outline described here. The header information must always be at the beginning of a source file, followed by the pin/node definitions. The other areas that are described are used as required, and may be in any order. A comprehensive CUPL Language Reference section is included in the on-line Help.

Header Information

The header contains fundamental information about the design. The type of information included is the name of the file, a part number for the device, a starting date, a design revision number, a company name, a designer's name, the assembly for the device, and the location of the device in the assembly. The header may also specify the device to be used for the design and the type of output to be generated by the compiler. The header information can appear in any order and all of the information is optional except for the name field. When a header item is missing, a warning message will appear, but the compilation will be uninterrupted

```
Name      XXXXXX;
Partno    XXXXXX;
Date      XX/XX/XX;
Revision  XX;
Designer  XXXXXX;
Company   XXXXXX;
Assembly  XXXXXX;
Location  XXXXXX;
Format    XXXXXX;
Device    XXXXXX;
```

Header information section

Title Block

The title block is a comment area provided for the designer to place a title and description of the design. The actual device name and manufacturer can be placed here for future use or second source possibilities.

```
/* **** */
/*                                           */
/*                                           */
/*                                           */
/* **** */
/* Allowable Target Device Types:          */
/* **** */
```

Title Block Section

Pin/node Definition

The pin/node section provides a comment label for the section and pin declaration statements that the designer can fill in with the numbers and labels for his design. The pin definitions can be in any order but it is easier to understand the design when the pins are grouped according to usage. Empty comments are provided after each pin definition to better describe the pins' role in the design

```

/** Inputs **/
Pin      =      ;      /*
Pin      =      ;      /*
Pin      =      ;      /*

/** Outputs **/
Pin      =      ;      /*
Pin      =      ;      /*
Pin      =      ;      /*

Pinnode   =      ;      /*
Pinnode   =      ;      /*

```

*Pin/Node Definition Section***Intermediate Variables**

Intermediate variables are used to generate logic equations or additional intermediate variables, which can then be used in other expressions. Writing logic equations in this “top down” manner yields a logic description file that is easier to read and comprehend

Logic Equations

This section is like the body of a letter. It contains all the important parts of the design like state machines, truth tables, and Boolean equations

Source File Directives

Most of the source file directives begin with a dollar sign (\$). These directives are commonly referred to as preprocessor commands. When used they must start in column one of the source file. Any combination of uppercase and lowercase letters may be used to type these commands. All source file directives may be placed on any line within the source file.

Defining And Undefined Symbolic Constants

To define a symbolic constant use the `$define` command. This command replaces a character string by another specified operator, number, or symbol. The replacement is a literal string substitution made on the input file before being processed by the Compiler. To undefine a symbolic constant use the `$undef` command. This command cancels the `$define` command.

Including Other Files

The inclusion of another file into a source file can be achieved by using the `$include` command. File inclusion allows the designer to standardize on a portion of commonly used CUPL code. It is also useful for keeping a separate parameter file that defines

constants that are commonly used in many source specifications. The files that are included may also contain \$include commands, allowing for “nested” include files.

Conditional Compilation

Conditional compilation allows the user to compile sections of source code based on the existence or nonexistence of defined symbolic constants. These commands are \$ifdef, \$ifndef, \$else, and \$endif. When a symbolic constant has previously been defined, the source statements following the \$ifdef command are compiled until the occurrence of an \$else or \$endif command. When the argument has not previously been defined, the source statements following the \$ifdef command are ignored. The \$ifndef command works in the opposite manner of the \$ifdef command. If the tested condition of the \$ifdef or \$ifndef commands is false, then any source statements between an \$else and \$endif command are compiled, otherwise they are ignored. The \$endif command is used to end a conditional compilation started with the \$ifdef or \$ifndef commands. Conditional compilation may be nested, and for each level of nesting of the \$ifdef or \$ifndef command, an associated \$endif must be used.

```
$DEFINE Prototype X /* define Prototype*/
$IFDEF Prototype
pin 1 = memreq;      /* memory request on */
                    /* pin 1 of prototype*/
pin 2 = ioreq;        /* I/O request on*/
                    /* pin 2 of prototype*/
$ELSE
pin 1 = ioreq;        /* I/O request on*/
                    /* pin 1 of PCB*/
pin 2 = memreq;       /* memory request on */
                    /* pin 2 of PCB*/
$ENDIF
```

Using Conditional Compilation

Repeating Statements

To repeat a set of CUPL language statements use the \$repeat and \$repnd commands. This command is similar to the FOR statement in C language and the DO statement in FORTRAN language. The statements between the \$repeat and \$repnd will be repeated as many times as the index in the \$repeat statement allows. This can make defining state machine based counters very easy.

CUPL Source Code	Results after preprocessing
FIELD sel = [in2..0]	FIELD sel = [in2..0];
\$REPEAT i = [0..7]	!out0 = sel:'h'0 & enable;
!out{i} = sel:'h'{i} & enable;	!out1 = sel:'h'1 & enable;
\$REPND	!out2 = sel:'h'2 & enable;
	!out3 = sel:'h'3 & enable;

```
!out4 = sel:'h'4 & enable;
!out5 = sel:'h'5 & enable;
!out6 = sel:'h'6 & enable;
!out7 = sel:'h'7 & enable;
```

Using \$repeat and \$repnd Commands

Using Macros

A macro is a user-defined command that allows the user to substitute a series of commands for a single word. This is accomplished by using the \$macro and \$mend commands. The statements between the \$macro and \$mend commands will not be compiled until the macro name is called. The macro is called by stating the macro name in the source file and passing the parameters to the macro. Macros can be used to create a library of decoders, counters, etc. A macro expansion (.MX) file can be created to see how the preprocessor processed the macro definition.

```
$MACRO decoder bits MY_X MY_Y MY_enable;
    FIELD select = [MY_Y{bits-1}..0];
    $REPEAT i = [0..{2**(bits-1)}]
        !MY_X{i} = select:'h'{i} & MY_enable;
    $REPEND
$MEND
_/* Other CUPL statements */
decoder(3, out, in, enable); /*macro function call*/
```

Using \$macro and \$mend Commands

Individually Minimizing Outputs

Outputs can be minimized on a pin by pin basis using the MIN statement. The minimization levels for the MIN statement are 0 to 4, and these correspond to the options in the Configure PLD dialog. This statement allows you to specify different minimization levels for different outputs in the same design.

The MIN statement numbers, and their corresponding option in the Configure PLD dialog are:

Number	Corresponding Minimization
0	None
1	Quick
2	Quine-McCluskey
3	Presto
4	Expresso

Following are examples of individually minimizing outputs:

```
MIN async_out= 0;           /* no reduction          */
MIN [outa, outb]    = 2;     /* Quine-McCluskey reduction */
MIN count.d= 4;        /* Espresso reduction        */
```

Blowing Technology-Dependent Fuses

Certain devices have fuses that can change the behavior of the whole device. Currently, these fuses are referred to as MISER and TURBO bits. Depending on how you want the device to behave, you need to either blow or not blow these fuses. This is accomplished by using the FUSE statement. State the fuse number and the blow value and the Compiler will set that fuse to the specified value. This statement should be used with utmost care, as it can lead to unpredictable results if used incorrectly.

```
FUSE(101,1);           /* Blow the Turbo bit */
FUSE(102,0);           /* Do not blow the Miser bit */
```

Using the FUSE statement

Language Elements

This section describes the elements that comprise the CUPL logic description language.

Pin/node Definition

Since the PIN definitions must be declared at the beginning of the source file, their definition is a natural starting point for a design. Nodes and pinnodes, used to define buried registers, should also be declared at the beginning of the source file. Pin assignment needs to be done if you already know the device you want to use. However, when creating a VIRTUAL design only the variable names that will later be assigned to pins need to be filled in. The area that normally contains the pin number is left blank.

Defining Intermediate Variables

Intermediate variables are variables that are assigned to an equation, but are not assigned to a PIN or NODE. These are used to define equations that are used by many variables or to provide an easier understanding of the design.

Using Indexed Variables

Variable names that end in a decimal number from 0 to 31 are referred to as indexed variables. They can be used to represent a group of address lines, data lines, or other sequentially numbered items. When indexed variables are used in bit field operations the variable with index number 0 is always the lowest order bit.

Using Number Bases

All operations involving numbers in the Compiler are done with 32-bit accuracy. Therefore, the numbers may have a value from 0 to $2^{32} - 1$. Numbers may be represented in any one of the four common bases: binary, octal, decimal, or hexadecimal. The default base for all numbers used in the source file is hexadecimal, except for device pin numbers and indexed variables, which are always decimal. Binary, octal, and hexadecimal numbers can have don't care ("X") values intermixed with numerical values.

Number	Base	Decimal Value
'b'0	Binary	0
'B'1101	Binary	13
'O'663	Octal	435
'D'92	Decimal	92
'h'BA	Hexadecimal	186
'O'[300..477]	Octal (range)	192..314
'H'7FXX	Hexadecimal (range)	32512..32767

Using List Notation

A list is a shorthand method of defining groups of variables. It is commonly used in pin and node declarations, bit field declarations, logic equations, and set operations. Square brackets are used to delimit items in the list.

Using Bit Fields

A bit field declaration assigns a single variable name to a group of bits. After making a bit field assignment using the FIELD keyword, the name can be used in an expression; the operation specified in the expression is then applied to each bit in the group. When a FIELD statement is used, the compiler generates a single 32-bit field internally. This is used to represent the variables in the bit field. Each bit represents one member of the bit field. The bit number which represents a member of a bit field is the same as the index number if indexed variables are used. This means that A0 will always occupy bit 0 in the bit field. This is mainly used for defining and manipulating address and data buses.

```
FIELD ADDRESS = [A7, A6, A5, A4, A3, A2, A1, A0];  
    or  
FIELD ADDRESS = [A7..0];  
  
FIELD Mode = [Up, Down, Hold];
```

Using The FIELD Statement

Language Syntax

This section introduces the logic and arithmetic operators and functions that are needed to create a design using the CUPL language.

Logical Operators

Four standard logical operators are available for use: NOT, AND, OR, and XOR. The following table lists the operators and their order of precedence, from highest to lowest.

Operator	Example	Description	Precedence
!	!A	NOT	1
&	A & B	AND	2
#	A # B	OR	3
\$	A \$ B	XOR	4

Arithmetic Operators And Functions

Six standard arithmetic operators are available for use in \$repeat and \$macro commands. The following table lists these operators and their order of precedence, from highest to lowest.

Operator	Example	Description	Precedence
**	2**3	Exponentiation	1
*	2*i	Multiplication	2
/	4/2	Division	2
%	9%8	Modulus	2
+	2+4	Addition	3
-	4-i	Subtraction	3

One arithmetic function is available to use in expressions being used in \$repeat and \$macro commands. The following table shows the arithmetic function and its bases.

Function	Base
LOG2	Binary
LOG8	Octal
LOG16	Hexadecimal
LOG	Decimal

The LOG function returns an integer value. For example:

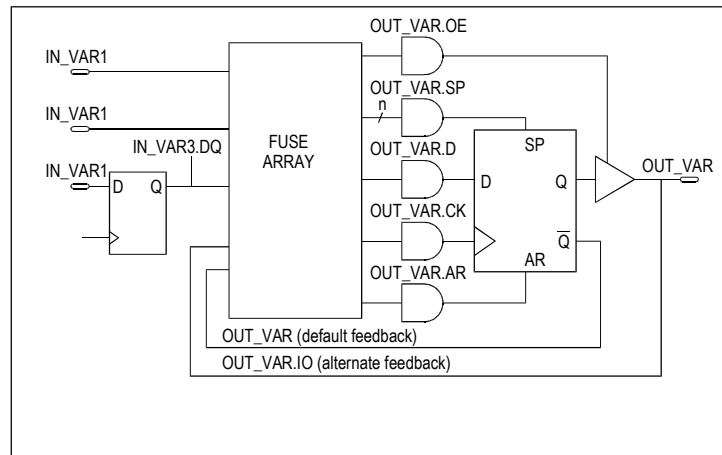
```
LOG2(32) = 5 <==> 2**5 = 32
LOG2(33) = ceil(5.0444) = 6 <==> 2**6 = 64
```

Ceil(x) returns the smallest integer not less than x.

Variable Extensions

Extensions can be added to variable names to indicate specific functions associated with the major nodes inside a programmable device, including such capabilities as flip-flop description and programmable tri-state enables. The Compiler checks the usage of the extension to determine whether it is valid for the specified device and whether its usage conflicts with some other extension used. The Compiler uses these extensions to configure the macrocells within a device. This way the designer does not have to know which fuses control what in the macrocells.

The figure below shows the use of extensions. Note that this figure does not represent an actual circuit, but shows how to use extensions to write equations for different functions in a circuit.



Circuit Illustrating Extensions

Logic Equations

Logic equations are the building blocks of the CUPL language. The form for logic equations is as follows:

```
[!] var [.ext] = exp ;
```

where:

var is a single variable or a list of indexed or non-indexed variables defined according to the rules for list notation. When a variable list is used, the expression is assigned to each variable in the list.

.ext is an optional variable extension to assign a function to the major nodes inside a programmable device.

exp is an expression; that is, a combination of variables and operators.

= is the assignment operator; it assigns the value of an expression to a variable or set of variables.

! is the complement operator.

In standard logic equations, normally only one expression is assigned to a variable. The APPEND statement enables multiple expressions to be assigned to a single variable. The APPENDED logic equation is logically ORed to the original equation for that variable. The format for using the APPEND statement is identical to defining a logic equation except the keyword APPEND appears before the logic equation begins.

Place logic equations in the “Logic Equations” section of the source file provided by the template file.

Using Set Operations

All operations that are performed on a single bit of information, for example, an input pin, a register, or an output pin, may be applied to multiple bits of information grouped into sets. Set operations can be performed between a set and a variable or expression, or between two sets.

The result of an operation between a set and a single variable (or expression) is a new set in which the operation is performed between each element of the set and the variable (or expression).

When an operation is performed on two sets, the sets must be the same size (that is, contain the same number of elements). The result of an operation between two sets is a new set in which the operation is performed between elements of each set.

When numbers are used in set operations, they are treated as sets of binary digits. A single octal number represents a set of three binary digits, and a single decimal or hexadecimal number represents a set of four binary digits.

Equality Operations

Unlike other set operations, the equality operation evaluates to a single Boolean expression. It checks for bit equality between a set of variables and a constant. The bit positions of the constant number are checked against the corresponding positions in the set. Where the bit position is a binary 1, the set element is unchanged. Where the bit position is a binary 0, the set element is negated. Where the bit position is a binary X, the set element is removed. The resulting elements are then ANDed together to create a single expression.

The equality operator can also be used with a set of variables that are to be operated upon identically. For example, the following three expressions:

```
[A3 , A2 , A1 , A0] : &
[B3 , B2 , B1 , B0] : #
[C3 , C2 , C1 , C0] : $
```

are equivalent respectively to:


```

A3 & A2 & A1 & A0
B3 # B2 # B1 # B0
C3 $ C2 $ C1 $ C0

```

Range Operations

The range operation is similar to the equality operation except that the constant field is a range of values instead of a single value. The check for bit equality is made for each constant value in the range.

First, define the address bus, as follows:

```
FIELD address = [A3..A0]
```

Then write the RANGE equation:

```
select = address:[C..F] ;
```

This is equivalent to the following equation:

```
select = address:C # address:D # address:E # address:F;
```

Truth Tables

Sometimes the clearest way to express logic descriptions is in tables of information. CUPL provides the TABLE keyword to create tables of information. First, define relevant input and output variable lists, and then specify one-to-one assignments between decoded values of the input and output variable lists. Don't-care values are supported for the input decode value, but not for the output decode value.

A list of input values can be specified to make multiple assignments in a single statement. The following block describes a simple hex-to-BCD code converter:

```

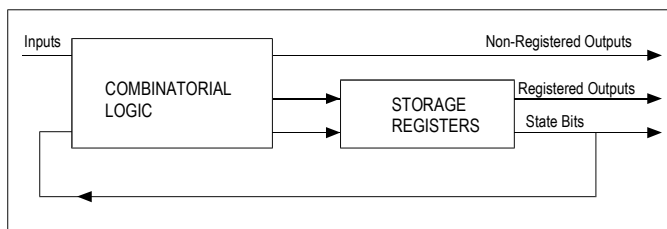
FIELD input = [in3..0] ;
FIELD output = [out4..0] ;
TABLE input => output {
0=>00; 1=>01; 2=>02; 3=>03;
4=>04; 5=>05; 6=>06; 7=>07;
8=>08; 9=>09; A=>10; B=>11;
C=>12; D=>13; E=>14; F=>15;
}

```

State Machines

A state machine, according to AMD/MMI, is “a digital device which traverses through a predetermined sequence of states in an orderly fashion”. A synchronous state machine is a logic circuit with flip-flops. Because its output can be fed back to its own or some other flip-flop's input, a flip-flop's input value may depend on both its own output and that of other flip-flops; consequently, its final output value depends on its own previous values, as well as those of other flip-flops.

The CUPL state-machine model, as shown below, uses six components: inputs, combinatorial logic, storage registers, state bits, registered outputs, and non-registered outputs.



State Machine Model

Inputs

Signals entering the device that originate in some other device.

Combinatorial Logic

Any combination of logic gates (usually AND-OR) that produces an output signal that is valid T_{pd} (propagation delay time) nsec after any of the signals that drive these gates changes. T_{pd} is the delay between the initiation of an input or feedback event and the occurrence of a non-registered output.

State Bits

Storage register outputs that are fed back to drive the combinatorial logic. They contain the present-state information.

Storage Registers

Any flip-flop elements that receive their inputs from the state machine's combinatorial logic. Some registers are used for state bits, others are used for registered outputs. The registered output is valid T_{co} (clock to out time) nsec after the clock pulse occurs. T_{co} is the time delay between the initiation of a clock signal and the occurrence of a valid flip-flop output.

To implement a state machine, CUPL supplies a syntax that allows the describing of any function in the state machine. The SEQUENCE keyword identifies the outputs of a state machine and is followed by statements that define the function of the state machine. The SEQUENCE keyword causes the storage registers and registered output types generated to be the default type for the target device. Along with the SEQUENCE keyword are the SEQUENCED, SEQUENCEJK, SEQUENCERS, and SEQUENCET keywords. Respectively, they force the state registers and registered outputs to be generated as D, J-K, S-R, and T-type flip-flops. The format for the SEQUENCE syntax is as follows:

```

SEQUENCE state_var_list {
PRESENT  state_n0
    IF (condition1) NEXT state_n1;
    IF (condition2) NEXT state_n2    OUT out_n0;

```

```

        DEFAULT          NEXT  state_n0;
PRESENT  state_n1
        NEXT  state_n2;
.
.
.
PRESENT  state_nn statements ;
}

```

where

- state_var_list is a list of the state bit variables used in the state machine block. The variable list can be represented by a field variable.
- state_n is the state number and is a decoded value of the state_variable_list and must be unique for each PRESENT statement.
- statements are any of the conditional, next, or output statements described in the following subsections of this section.

Multiple State Machines

The CUPL syntax allows for more than one state machine to be defined within the same PLD design. When multiple state machines are defined, occasionally the designer would like to have the state machines communicate with each other. That is, when one state machine reaches a certain state another state machine may begin. There are two ways of accomplishing state machine communication: using set operations on the state bits or defining a “global” register that can be accessed by both state machines.

In one state machine a conditional statement can contain another state machine’s name followed by a state number or range of state numbers. The conditional statement will become TRUE when the other state machine reaches that particular state or states. The same case is true when using a register that is accessed by multiple state machines. However, this way needs to use one of the devices output or buried registers. Depending on the situation, the global register could also be combinatorial which may make a difference as to when the state machine receives the information from another state machine.

Condition Statement

The CONDITION syntax provides a higher-level approach to specifying logic functions than does writing standard Boolean logic equations for combinatorial logic. The format is as follows:

```

CONDITION {
  IF expr0 OUT  var ;
  .
  .
  IF exprn OUT  var ;
}

```

```
    DEFAULT OUT var ;  
}
```

The CONDITION syntax is equivalent to the asynchronous conditional output statements of the state machine syntax, except that there is no reference to any particular state. The variable is logically asserted whenever the expression or DEFAULT condition is met.

Defining a Function

The FUNCTION keyword permits the creating of personal keywords by encapsulating some logic as a function and giving it a name. This name can then be used in a logic equation to represent the function. The format for user-defined functions is as follows:

```
FUNCTION name ([parameter0, ..., parametern])  
{  
    body  
}
```

The statements in the body may assign an expression to the function, or may be unrelated equations.

When using optional parameters, the number of parameters in the function definition and in the reference must be identical. The parameters defined in the body of the function are substituted for the parameters referenced in the logic equation. The function invocation variable is assigned an expression according to the body of the function. If no assignment is made in the body statements, the function invocation variable is assigned the value of 'h'o.

Compiling the Programmable Logic Design

This chapter describes how to; select the target device, configure the PLD Compiler options, and select the required output formats.

Selecting the Target Device

The PLD Compiler is like any compiler, in that it processes a source file and produces output based on a target architecture. This compiler, however, has hundreds of different target architectures. This allows a design to be implemented in many different architectures without making significant changes to the source file.

The target device is specified in the Configure PLD dialog. The Options Tab displays the current Target Device. To select a different target device click on the Change button, the Target Device dialog will pop up.

- ◆ When you compile you can also use the Virtual Device option. This allows you to verify that the design will compile, as well as examine the maximum number of product terms it will require (check the DOC file). The number of product terms is often the reason a larger device must be used – experiment with the various compiler options and optimizations to reduce this number.

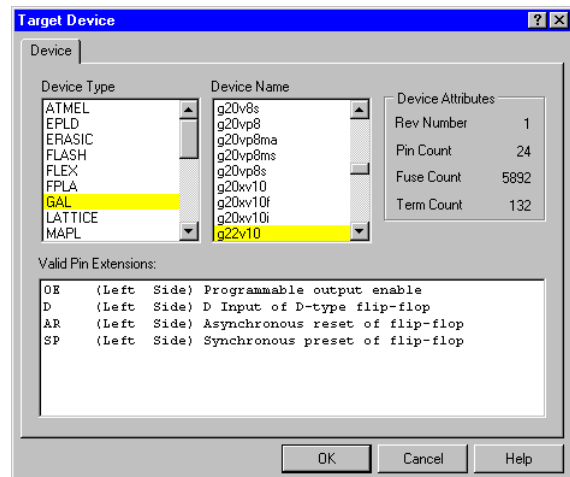
Specifying a Device in the Compiler Source File

The target device is also specified in the Compiler source file, in the “DEVICE” field in the header section of the PLD file. This field is automatically updated whenever you select a different device in the Configure PLD dialog.

Selecting a Device in the Target Device Dialog

To access the Target Device dialog press the Change button in the Configure PLD dialog. The Target Device dialog gives a description of each supported target device.

To change the target device, select the Device Type, the Device Name, and press OK.



The Device Library

The information displayed in the Target Device dialog is contained in the device libraries. There is a device library (*filename.DL*) for each type shown in the Device Type field. These binary files contain a description of each of the target devices supported by the Compiler. The library describes the physical characteristics of each device including; internal architecture, number of pins, valid input and output pins, and also describes the logical characteristics, including registered and non-registered pins, number of product terms, fuse map information, and download format information.

Device Mnemonics (Device Names)

There is a naming system used to identify the various device architectures, referred to as *device mnemonics*. It is important to note that this naming system is for architectures, not for the devices themselves. For example, Altera EP512 and Intel 5AC312 both have the same architecture. Therefore, if you were doing a design with either of these devices from these manufacturers, you would tell the Compiler that your target architecture is EP312.

◆ A device mnemonic can reference multiple manufacturers. Locate the device you require in the *devhelp.TXT* file to find its mnemonic, then refer to mnemonic vs. manufacturer cross reference file, *devices.TXT* to see which manufacturer this mnemonic is listed under in the Target Devices dialog. You can type this mnemonic directly into the Target Device field of the Configure PLD dialog.

The mnemonic is composed of a device family prefix and industry standard part number suffix. Following is a list of the device mnemonic prefixes:

Symbol	Meaning
EP	Erasable Programmable Logic Device (EPLD)
G	Generic Array Logic (GAL)
F	Field Programmable Logic Array (FPLA)
F	Field Programmable Gate Array (FPGA)
F	Field Programmable Logic Sequencer (FPLS)
F	Field Programmable Sequence Generator (FPSG)
P	Programmable Logic Array (PAL)
P	Programmable Logic Device (PLD)
P	Programmable Electrically Erasable Logic (PEEL)
PLD	Pseudo Logical Device
RA	Bipolar Programmable Read Only Memory (PROM)

For example, the device mnemonic for a PAL10L8 is P10L8; for an 82S100 the device mnemonic is F100. For bipolar PROMs, the suffix is the array size; for example, the device mnemonic for a 1024 x 8 bipolar PROM is RA10P8, since there are 10 address input pins and 8 data output pins.

Virtual Device

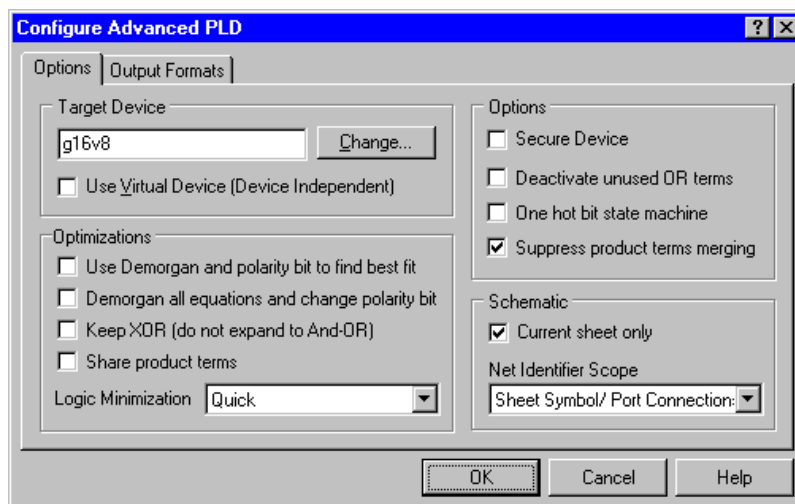
The virtual device option allows the user to create digital designs for programmable logic without regard to the target architecture. The virtual device is not a device at all. It is simply a removal of the restrictions by the Compiler, allowing the design to contain unlimited product terms and pins. With the virtual device, the Compiler allows all register types to be used. The virtual device is useful for determining the resources needed to implement your design.

When using the Virtual device the Compiler ignores the polarity in the pin declaration. The pin number can be left out.

Compiler Options

The Options Tab of the Configure PLD dialog allows you to:

- Select the Target Device.
- Enable the Optimization Options.
- Enable the Compile Options.
- Set the Logic Minimization level.
- Specify the schematic-based PLD compilation scope.



Following is a description of each option:

Option	Description
Use DeMorgan and polarity bit	Optimize product term usage for pin or pinnode variables. This overrides the DEMORGAN statement if it appears in the source file.
DeMorgan all equations	DeMorganize all pin and pinnode variables. This overrides the DEMORGAN statement if it appears in the source file.
Keep XOR	Do not expand XOR to AND-OR equations. This is used for device-independent designs or designs targeted for fitter-supported devices where the fitter supports XOR gates.
Share Product Terms	Force product term sharing during minimization. This is also referred to as group reduction.

Secure Device	Adds the necessary code in the JEDEC download file to automatically allow the device programmer to blow the security fuse when programming, making the device non-erasable. Not all programmers support this option.
Deactivate unused OR terms	In IFL devices, the OR-gate output array is driven by each of the AND-gate product terms. Normally, unused OR-gate inputs are left connected to the product term array so that new terms may be added. However, with this option, the unused OR-gate inputs are removed (deactivated) from the product term array. The result is reduced propagation delay from input to output.
Suppress product terms merging	In IFL devices, each product term from the AND-gate array may be shared among any number of OR-gate outputs. This option defeats this capability, forcing identical product terms to be generated for each output OR-array when required. The result is reduced propagation delay from input to output. This option will also force minimization to be performed on each output individually (as opposed to minimization on all outputs at once) when Quine-McCluskey or Espresso minimization is chosen.
One-hot-bit State Machines	This is an option for FPGA designers. Using this option will cause the compiler to generate state machine equations as “one-hot-bit”. This has some distinct advantages in register-rich architectures such as XILINX devices. The fanning is reduced making routing much easier and timing problems associated with variable length feedback paths from register to register are eliminated. Try compiling with and without this option to see the difference. Currently, the Compiler treats all state machines in the design as “one-hot-bit” if the option is used.
Net Identifier Scope	Specifies how the sheet-to-sheet connections are created in a multi-sheet design. There are effectively 2 ways of creating connections: horizontally, (Net Labels and Ports global and Only Ports Global) where connections are created directly from one sub-sheet to another; or vertically (Sheet Symbol/Port Connections), where connections are created from a sub-sheet port to the matching sheet entry.
Current Sheet Only	Use this to compile the current sheet only.

Minimization Methods

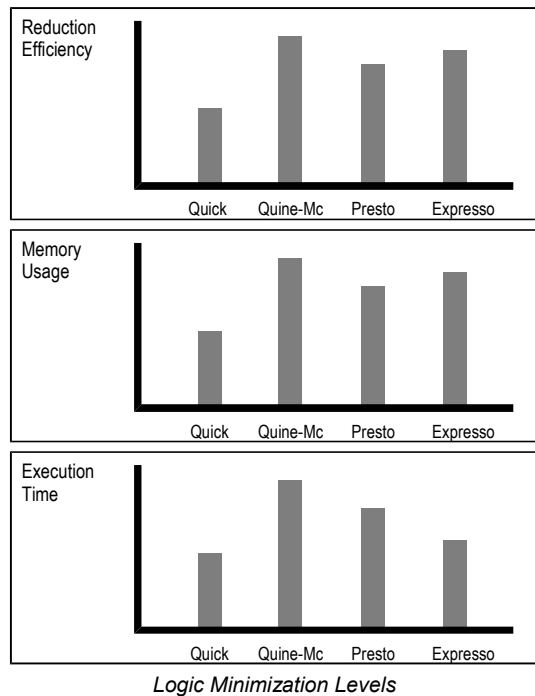
The Minimization Methods are the logic reduction algorithms the compiler will use. There are four minimization methods available:

- Quick
- Quine-McCluskey
- Presto
- Espresso

Minimization levels Quine-McCluskey and Presto will perform multiple output minimization in IFL devices. This maximizes product term sharing in these types of devices.

Selecting None defeats all logic minimization during the compilation. It is useful when working with PROMs, to keep contained product terms from being eliminated.

The diagram shows the relative memory usage, speed, and efficiency of the four minimization levels.



Boolean Logic

The table shows the Boolean Logic rules for eliminating excess product terms from the expanded equations, used by the logic reduction algorithms built into the compiler.

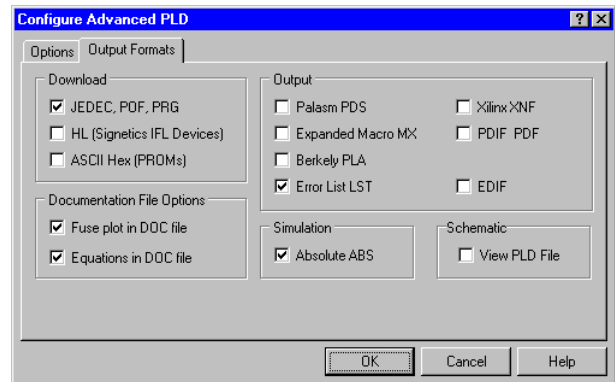
Expression	Result
!0	1
!1	0
A & 0	0
A & 1	A
A & A	A
A & !A	0
A # 0	A
A # 1	1
A # A	A
A # !A	1
A & (A # B)	A
A # (A & B)	A

Output Formats

The PLD Compiler can produce the following types of output files:

- Download formats, suitable for downloading to a programmer.
- Output formats, which include formats for interfacing to third-party fitters and error listings.
- Documentation formats, which can include fuse plots and equations.
- A simulation format, which is essentially a programmed virtual device. The simulator then applies the test vectors from your test specification source file (*filename.SI*) to this virtual device, and records the results in the simulator listing file (*filename.SO*).

To configure the Compiler Output Formats select the check boxes on the Output Formats Tab of the Configure PLD dialog.



Following is a description of each Output Format:

Output Formats	Description
JEDEC, POF, PRG	Generates a JEDEC-compatible ASCII download file (<i>filename.JED</i>). The filename is specified by the NAME statement in the header section of the logic description file.
HL	Generates an HL download file (<i>filename.HL</i>). This format is available only for the Signetics IFL devices. The filename is specified by the NAME statement in the header section of the logic description file.
ASCII Hex	Generates an ASCII-hex download file (<i>filename.HEX</i>). This format is available only for PROMs. The filename is specified by the NAME statement in the header section of the logic description file.
PALASM PDS	Generates a PALASM format file (<i>filename.PDS</i>), to the standard set by Monolithic Memories in the PAL Handbook (Third Edition).

Expanded Macro MX	Generates an expanded macro definition file (<i>filename.MX</i>) which contains an expanded listing of all macros used in the source file. It also contains the expanded expressions that use the REPEAT command.
Berkeley PLA	Generates a Berkeley PLA file (<i>filename.PLA</i>) for use by the Berkeley PLA tools, such as PLEASURE, or other PLA layout tools which use the Berkeley PLA format. This format is also used as an input to a number of the external fitters.
Error list LST	Generates an error listing file (<i>filename.LST</i>). Each line in the original source file is numbered. Error messages are listed at the end of the file and use the line numbers for reference.
XNF Xilinx	This can be used to create an input file for other logic design tools and gate array fitters such as PDS2XNF from XILINX.
PDIF PDF	Generates a PDIF (P-CAD Database Interchange Format) file (<i>filename.PDF</i>) which can be translated by the PDIFIN program into a symbol for the PC-CAPS (P-CAD Schematic Capture) program. The generated symbol will contain packaging information for the PLD.
EDIF	Generates an EDIF format file.
Equations in DOC File	Generates a documentation file (<i>filename.DOC</i>) which contains an expanded listing of the logic terms in sum-of-products format and a symbol table of all variables used in the source file. It includes the total number of product terms and the number available for each output.
Fuse Plot in DOC File	Generates a fuse plot in the documentation file. For PAL devices, each output pin is listed and the associated product term rows are shown with the starting JEDEC fuse number. Fuses present are denoted with “x”. Fuses blown are denoted with “-”. For IFL devices, the HL download format is used, showing JEDEC fuse numbers with input terms denoted as “H,” “L,” “0,” or “-”.
Absolute ABS	Generates an absolute file (<i>filename.ABS</i>) used by the PLD logic Simulator.
View PLD File	Automatically open the intermediate PLD file produced during a schematic-based PLD compilation.

Simulating the Programmable Logic Design

This chapter explains how to create a test specification source file to simulate the programmable logic device under design. Test vectors specify the expected functional operation of a PLD by defining the outputs as a function of the inputs. Test vectors are used both for simulation of the device logic before programming and for functional testing of the device once it has been programmed. The PLD Simulator can generate JEDEC-compatible downloadable test vectors, which are appended to the .JED file created during compilation.

Input to the Simulator

A test specification source file (*filename.SI*) is the input to the Simulator. It contains a functional description of the requirements of the device in the circuit.

The input pin stimuli and output pin test values entered in the source file are compared to the actual values calculated from the logic equations in the PLD source file. The calculated values are contained in the absolute file (*filename.ABS*), which is created during compilation when the absolute ABS format is specified. The absolute file must be created during compilation before running the Simulator.

Output from the Simulator

The Simulator outputs are;

- a simulation listing file
- vectors appended to the JEDEC downloadable fuse link file

The simulation listing file (*filename.SO*) contains the results of the simulation. It has the same filename as the input test specification file.

The simulation listing is an ASCII file. All header information is displayed in the listing file with any header errors marked appropriately. Each complete vector is assigned a number. Any output tests that failed are flagged with the actual (simulator-determined) output value displayed. Each variable in error is listed along with the expected (user-supplied) value. Any invalid or unexpected test values are listed along with an appropriate error message.

The simulation results can be displayed as a series of waveforms, by double-clicking to open the .SO file with the PLD Waveform Editor.

The Simulator appends the test vectors to an existing JEDEC downloadable fuse link file (*filename.JED*) created during compilation. Enable the JEDEC option in the Formats Tab of the Configure PLD dialog to create this file.

Creating a Test Specification Source File

The test specification file (*filename.SI*) is an ASCII file, created with the Text Editor. The filename is the same as the corresponding CUPL logic description source file, except with the .SI extension. The test specification file should include the following:

- Header information
- Comments
- Variable ordering
- Base sets
- Test vectors
- Simulator directives

♦ For an example of how to create a simulation test specification file, refer to the *Sample PLD Design Session* chapter.

Header Information

Header information which is entered must be identical to the information in the corresponding CUPL logic description file (.PLD). If any header information is different, a warning message appears, stating that the status of the logic equations could be inconsistent with the current test vectors in the test specification file.

Following is a list of the keywords used for header information:

```

PARTNO
NAME
REVISION
DATE
DESIGNER
COMPANY
ASSEMBLY
LOCATION
DEVICE
FORMAT

```

The easiest way to begin creating a test specification file is to copy the header information from the corresponding CUPL source file.

Specifying a Device

The Simulator accesses device information in the device libraries (*LibraryName.DL*). The library describes the physical characteristics of each device including; internal architecture, number of pins, type of registers available, the logical characteristics (including registered and non-registered pins), feedback capabilities, register power-on state and register control features.

Reference the target device using device mnemonics. Each mnemonic is composed of a device family prefix and industry-standard part number suffix. The table below lists the device mnemonic prefixes.

Prefix	Device Family
EP	Erasable Programmable Logic Device (EPLD)
G	Generic Array Logic (GAL)
F	Field Programmable Logic Array (FPLA)
F	Field Programmable Gate Array (FPGA)
F	Field Programmable Logic Sequencer (FPLS)
F	Field Programmable Sequence Generator (FPSG)
P	Programmable Logic Array (PAL)
P	Programmable Logic Device (PLD)
P	Programmable Electrically Erasable Logic (PEEL)
PLD	Pseudo Logical Device
RA	Bipolar Programmable Read-Only Memory (PROM)

For example, the device mnemonic for a PAL10L8 is P10L8; for an 82S100 the device mnemonic is F100. For bipolar PROMs, the suffix is the array size. For example, the device mnemonic for a 1024 x 8 bipolar PROM is RA10P8, since there are 10 address input pins and 8 data output pins.

The target device can be specified either in the test specification source file (*filename.SI*), or selected in the Options Tab of the Configure PLD dialog.

Comments

Comments can be placed anywhere within the test specification file. Comments can be used to explain the contents of the specification file or the function of certain test vectors. A comment begins with a slash-asterisk (/*) and ends with an asterisk-slash (*). Comments can span multiple lines and are not terminated by the end of a line. However, comments cannot be nested.

Statements

The Simulator provides the keywords, ORDER, BASE, and VECTORS to write statements in the source file that determine the simulation output and how it is displayed. The following sections describe how to write statements with the CUPL keywords.

ORDER Statement

Use the ORDER keyword to list the variables to be used in the simulation table, and to define how they are displayed. Typically, the variable names are the same as those in the corresponding CUPL logic description file. Place a colon after ORDER, separate each variable in the list with a comma, and terminate the list with a semicolon. The following is an example of an ORDER statement:

```
ORDER: inputA, inputB, output ;
```

Only those variables that are actually used in the simulation must be listed.

The polarity of the variable name can be different than was declared in the CUPL logic description file, allowing simulation of active-LO outputs with an active-HI simulation vector. The variable names can be entered in any order; the Simulator automatically creates the proper order and polarity of the resulting vector to match the requirements of the JEDEC download format for the device. When indexed variables are used in the ORDER statement, they can be expressed in list notation format. However, since the ORDER statement is already in list form, square brackets are not needed to delimit the ORDER set. The following is an example of two equivalent ORDER statements; the first statement lists all the variables, and the second is written in list form.

```
ORDER: A0, A1, A2, A3, SELECT, !OUT0, !OUT1;
ORDER: A0..3, SELECT, !OUT0..1 ;
```

In list notation format, the polarity of the first indexed variable (!OUT0 in the above example) determines the polarity for the entire list. Bit fields that are declared in the CUPL logic description file can be referenced by their single variable name. Bit fields can also be declared in the test specification file for the Simulator, using FIELD declaration statements. The FIELD statement must appear before the ORDER statement.

The ORDER statement can be used to specify the format of the vector results in the simulator listing file. By default, variable values are displayed without spaces between columns. For example, the following ORDER statement

```
ORDER: clock, input, output ;
```

generates the following display in the output file (using sample values):

```
0001: C0H
0002: C1L
```

Spaces can be inserted between columns by using the % symbol and a decimal value between 1 and 80. For example, the following ORDER statement

```
ORDER: clock, %2, input, %2, output ;
```

generates the following display in the output file:

```
0001: C 0 H
0002: C 1 L
```

Note that the ORDER statement must be terminated by a semicolon.

Text can be inserted into the output file by putting a character string, enclosed by double quotes (" ") into the ORDER statement. For example, the following ORDER statement


```
ORDER: "Clock is ", clock,
      " and input is ", input,
      " output goes ", output ;
```

produces the following result in the output file:

```
0001: Clock is C and input is 0 output goes H
0002: Clock is C and input is 1 output goes L
```

Multiple ORDER statements

Several ORDER statements can be defined in a single SI file. For example, if the file TEST.SI has the following contents:

```
Name          test;
Partno         XXXXX;
Date           XX/XX/XX;
Revision       XX;
Designer       XXXXX;
Company        XXXXX;
Assembly       XXXXX;
Location       XXXXX;
Device         gl6v8;

Order: A, %1, B, %1, X, %1, Y;
Vectors:
  0 0 H L
  0 1 H H
  1 0 H H
  1 1 L L
  0 X H X
  X 0 H X
  1 X X X
  X 1 X X
Order: A, B, X;
Vectors:
  0 0 H
  0 1 H
  1 0 H
  1 1 L
  0 X H
  X 0 H
  1 X X
  X 1 X
```

The file TEST.SO will look like this:

```
CSIM: CUPL Simulation Program
Version 4.2a Serial# ...
CREATED Wed Dec 04 02:14:12 1991
LISTING FOR SIMULATION FILE: test.si
1: Name test;
2: Partno XXXXX;
3: Date XX/XX/XX;
4: RevisionXX;
5: DesignerXXXXX;
6: Company XXXXX;
7: AssemblyXXXXX;
8: LocationXXXXX;
9: Device gl6v8;
10:
11: Order: A, %1, B, %1, X, %1, Y;
12:
=====
      A B X Y
=====
0001: 0 0 H L
0002: 0 1 H H
0003: 1 0 H H
0004: 1 1 L L
0005: 0 X H X
0006: X 0 H X
0007: 1 X X X
0008: X 1 X X
25: Order: A, B, X; 26:
=====
      ABX
=====
0010: 00H
0011: 01H
0012: 10H
0013: 11L
0014: 0XH
0015: X0H
0016: 1XX
0017: X1X
```

BASE Statement

In most cases, each variable in the ORDER statement (except for FIELD variables) has a corresponding single character test value that appears in the test vector table of the output file. Multiple test vector values can be represented with quoted numbers. Use single quotes for input values and double quotes for output values. Enter a BASE statement to specify how each quoted number is expanded. The format for the BASE statement is:

```
BASE: name;
```

where

name is either octal, decimal or hex.

Follow BASE with a colon. Note that the base statement must be terminated by a semicolon.

The default base for quoted test values is hexadecimal. The BASE statement must appear in the file before the ORDER statement.

If the base is decimal or hexadecimal, quoted numbers expand to four digits; if the base is octal, they expand to three digits. For example, a test vector entered as '7' is interpreted as follows:

```
1 1 1          Base is octal
```

or

```
0 1 1 1       Base is decimal
```

or

```
0 1 1 1       Base is hex
```

More than one hexadecimal or octal digit may be entered between quotes. For example, '563' expands to the following:

```
1 0 1 1 1 0 0 1 1      Base is octal
```

or

```
0 1 0 1 0 1 1 0 0 0 1 1  Base is decimal
```

or

```
0 1 0 1 0 1 1 0 0 0 1 1  Base is hex
```

Quoted values may also be used with all other test values. For example, if the base is set to octal

```
"XX"      expands to X X X X X X
```

```
"LL"      expands to L L L L L L
```

```
"45"      expands to H L L H L H
```

Note that quoted values cannot contain *.

Test values for FIELD variables can be expressed either individually (for example, 001, HHLL) or with quoted values (for example, '1', "C"). When quoted values are

used, the value is automatically expanded to the number of variables in the field. For example, for the following address field

```
FIELD address = [A0..5] ;
```

A test value of

```
/*
A      A      A      A      A      A
5      4      3      2      1      0
-----*/
1      1      1      0      0      1
```

could be written using single test values, or '39' using quoted test values.

VECTORS Statement

Use the VECTORS keyword to prefix the test vector table. Following the keyword, include test vectors made up of single test values or quoted test. Each vector must be contained on a single line. No semicolons follow the vector. The following table lists allowable test vector values:

Test Value	Description
0	Drive input LO (0 volts) (negate active-HI input)
1	Drive input HI (+5 volts) (assert active-HI input)
C	Drive (clock) input LO, HI, LO
K	Drive (clock) input HI, LO, HI
L	Test output LO (0 volts) (active-HI output negated)
H	Test output HI (+5 volts) (active-HI output asserted)
Z	Test output for high impedance
X	Input HI or LO, output HI or LO. NOTE that not all device programmers treat X on inputs the same; some put it to 0, some allow input to be pulled to 1, and some leave it at the previous value.
N	Output not tested
P	Preload internal registers (value is applied to !Q output)
R	Random input generation
*	Outputs only -simulator determines test value and substitutes in vector
"	Enclose input values to be expanded to a specified BASE (octal, decimal, or hex). Valid values are 0-F and X.
""	Enclose output values to be expanded to a specified BASE (octal, decimal, or hex.) Valid values are 0-F, H, L, Z, and X.

Following is an example of a test vector table:

```
VECTORS:
0 0 1 1 1 'F' Z "H" /* test outputs HI */
0 1 1 0 0 '0' Z "L" /* test outputs LO */
```

Random Input Generation

The R vector can appear wherever a 0 or a 1 can appear. When encountered, a random value (either 0 or 1) is generated for the corresponding signal in that test vector.

An example of R in the SI file:

```
$repeat 10;
C 0 RRR 1RRRRRRR *****
```

Gives the following result in the SO file:

```
0035: C 0      000 10001011      HLLLHLHH
0036: C 0      000 11100111      HHHLLHHH
0037: C 0      110 10111101      HHHHLHHL
0038: C 0      111 11000100      HLLLHLLH
0039: C 0      101 10001011      LHLHHHLL
0040: C 0      101 10000110      LLHHLHLL
0041: C 0      010 10000001      LHHLLLLL
0042: C 0      000 10010000      HLLHLLLL
0043: C 0      001 11110100      LHHHHLHL
0044: C 0      001 10011110      LHLLHHHH
```

Don't Care

Unlike many other simulators, the PLD Simulator treats the DON'T-CARE (state X) as any other value. State X is not assumed to be 0 on input and N on the output. The X state allows specific determination of which inputs affect the output value, according to the rules listed in the truth tables below.

NOT : ones complement !		AND &		
A	!A	A	B	A & B
0	1	0	0	L
1	0	0	1	L
X	X	0	X	L
		1	0	L
		1	1	H
		1	X	X
		X	X	X

OR #			XOR : exclusive OR \$		
A	B	A # B	A	B	A \$ B
0	0	L	0	0	L
0	1	H	0	1	H
0	X	X	0	X	X
1	0	H	1	0	H
1	1	H	1	1	L
1	X	H	1	X	X
X	X	X	X	X	X

Vector Truth Tables

Preload

Use the P test value on the clock pin of a registered device to preload internal registers of a state machine or counter design to a known state, if the device does not have a dedicated TTL-level preload pin. The device programmer uses a supervoltage to actually load the registers. All input pins to the device are ignored and hence should be defined as X. The values that appear for registered variables are loaded into the !Q output of the register. These values (0 or 1) are absolute levels and are not affected by output polarity nor inverting buffers. The following is an example of a preload sequence for an active-LO output variable in a device with an inverting buffer between the register Q output and device pin:

```
ORDER: clock, input1, input2 , !output ;
VECTORS:
P X X 1 /* reset flip-flop */
      /* !Q goes to 1 */
      /* Q goes to 0 */
0 X X H /* output is HI due to */
      /* inverting buffer */
```

The Simulator can simulate and generate preload test vectors even for devices that do not have preload capability. However, not all PLDs are capable of preload using a supervoltage. Some devices have dedicated preload pins to use for this purpose. The

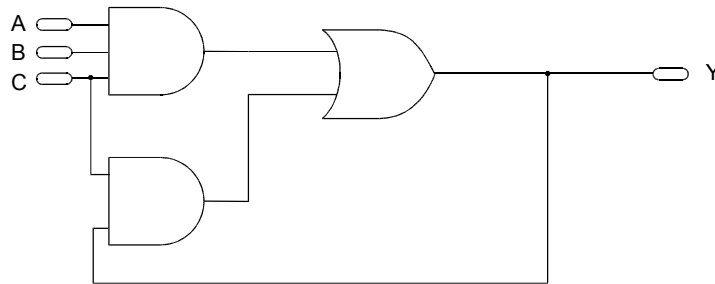
Simulator does not verify whether the device under simulation is actually capable of preload because parts from different manufacturers exhibit different characteristics. Before using the preload capability, determine whether the device being tested is physically capable of supervoltage preloading.

Clocks

Most synchronous devices (devices containing registers with a common clock tied to an output pin) use an active-HI (positive edge triggered) clock. To assure proper Simulator operation for these devices, always use a C test value (not a 1 or 0) on the clock pin. For synchronous devices with an active-LO (negative edge triggered) clock, use the K test value on the clock pin.

Asynchronous Vectors

When writing test vectors for a circuit with asynchronous feedback, changing two test values at once can create a spike condition that produces anomalous results. The circuit shown below is a circuit with three inputs (A, B, and C), and an output at Y that feeds back.



Circuit with feedback

The equation for the output at Y is as follows:

$$Y = A \ \& \ B \ \& \ C \ \# \ C \ \& \ Y$$

The vectors table shows an expected low output at Y based on the specified input values.

	A	B	C	Y
0001	0	0	0	L
0002	0	1	1	L
0003	1	0	1	L

Vectors table for circuit with feedback

Because one of the inputs is 0 in each of the vectors, the AND gate defined by A, B, and C produces a low output. The low value feeding back from the Y output keeps the

other AND gate low also. Therefore, the OR gate (driven by the output of the two AND gates) and consequently the output at Y remain low for the specified test vectors.

However, when the programmer operates on the test vectors, it applies values serially, beginning with the first pin. Because two test values change between vectors, the programmer creates intermediate results (labeled "a" in the table below).

	A	B	C	Y
0001	0	0	0	L
0001a	0	1	0	L
0002	0	1	1	L
0002a	1	1	1	H
0003	1	0	1	H

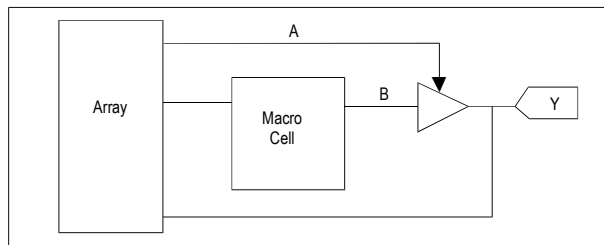
Vectors table with intermediate results

The intermediate result, [0002a], produces a high value for the output at Y. This high value feeds back and combines with the "1" value specified for input C in vector [0003] to produce a high output for the AND gate and consequently for the OR gate and for the output at Y. This high value conflicts with the expected low value specified in the third test vector, and the result is a spike condition.

By taking care to always change only one value between test vectors, the spike condition described above can be avoided. Also, in the source specification file, it is possible to specify a TRACE value of 1, 2, or 3 (rather than the default value of 0) that instructs the Simulator to display intermediate results in the output file (see "TRACE" in the following topic, *Simulator Directives*).

I/O Pin simulation

When writing test vectors for a design that has input/output capability and a controllable output enable (OE), the test vector value placed at the I/O pin will depend on the value of the output enable. If the output enable is active, the I/O pin needs an output test value (L, H, *,...). If the output enable becomes inactive, a Hi-Z (Z) will appear on the I/O pin. At this time, input test values (0, 1, ...) can be placed on the I/O pin allowing that pin to behave as an input pin. When the output enable is activated again, the test values for that pin will reflect the output of the macrocell.



I/O pin simulation

The following equations express the Boolean equation representation:

```

Y = B;
Y.OE = A;

```

When A is TRUE, the output of the macrocell (B) will appear at the pin (Y). When A is FALSE, the output enable will be deactivated and a Hi-Z will appear at the pin (Y). After the output enable is deactivated, input values can be placed on the pin. Here is an example of what the simulation file will look like:

```

Order: A, %1, B, %3, Y;

Vectors:
1 0  L      /* OE is ON */
1 1  H
0 0  Z      /* OE is OFF */
0 0  1      /* a valid input value can be
              placed on pin Y */
1 0  L      /* OE is ON again */

```

Variable Declaration (VAR)

Syntax: VAR <var_name> = <var_list>;

<var_name> - string of up to 20 characters that can be letters, digits or (underscore), but cannot end with a digit.

<var_list> - a list of symbols from the order statement (single, grouped or fields), previously defined variables, separated by commas.

<var_list> = [!]<field> | [!]<group> | [!]<var> [..[!]<var> | ,<var_list>]

Action:

It groups all the entities contained in <var_list> under one generic name for further references. It is similar to the FIELD statement, except this statement cannot appear before the ORDER statement. It is used between the ORDER statement and the VECTORS statement.

Example:

```
VAR Z = Q7..4;
```

Simulator Directives

The Simulator provides six directives that can be placed on any row of the file after the VECTOR statement. All directive names begin with a dollar sign and each directive statement must end with a semicolon. The Simulator directives are:

\$MSG	\$SIMOFF
\$REPEAT	\$SIMON
\$TRACE	\$EXIT

\$MSG

Use the \$MSG directive to place documentation messages or formatting information into the simulator output file. For example, a header for the simulator function table, listing the variable names, may be created. The format is as follows:

```
$MSG "any text string" ;
```

In the output table, the text string appears without the double quotes.

Blank lines can be inserted into the output, for example, between vectors, by using the following format:

```
$MSG " " ;
```

\$REPEAT

The \$REPEAT directive causes a vector to be repeated a specified number of times. Its format is:

```
$REPEAT n ;
```

where

n is a decimal value between 1 and 9999.

The vector following the \$REPEAT directive is repeated the specified number of times.

The \$REPEAT directive is particularly useful for testing counters and state transitions. Use the asterisk (*) to represent output test values supplied by the Simulator. The following example shows a 2-bit counter from a CUPL source file, and a VECTORS statement using the \$REPEAT directive to test it.

From the Compiler:

```
Q0.d = !Q0 ;
Q1.d = !Q1 & Q0 # Q1 & !Q0 ;
```

In the Simulator:

```
ORDER: clock, input, Q1, Q0 ;
VECTORS:
0 0 X X /* power-on condition */
P X 1 1 /* reset the flip-flops */
0 0 H H
$REPEAT 4 ; /* clock 4 times */
C 0 * *
```

The above file generates the following test vectors:

```
0 0 X X
P X 1 1
0 0 H H
C 0 L L
C 0 L H
C 0 H L
C 0 H H
```

The Simulator supplies four sets of vector values.

\$TRACE

Use the \$TRACE directive to set the amount of information that the Simulator prints for the vectors during simulation. The format is

```
$TRACE n ;
```

where n is a decimal value between 0 and 4.

Trace level 0 (the default) turns off any additional information and only the resulting test vectors are printed. When non-registered feedback is used in a design, the value for the output feeding back is unknown for the first evaluation pass of the vector. If the new feedback value changes any output value, the vector is evaluated again. All outputs must be identical for two passes before the vector is determined to be stable.

Trace level 1 prints the intermediate results for any vector that requires more than one evaluation pass to become stable. Any vector that requires more than twenty evaluation passes is considered unstable.

Trace level 2 identifies three phases of simulation for designs using registers. The first phase is "Before the Clock," where intermediate vectors using non-registered feedback are resolved. The second phase is "At the Clock," where the values of the registers are given immediately after the clock. The third phase is "After the Clock," where the outputs utilizing feedback are resolved as in trace level 1.

Trace level 3 provides the highest level of display information possible from the Simulator. Each simulation phase of "Before Clock," "At Clock," and "After Clock" is printed and the individual product term for each variable is listed. The output value for the AND gate is listed along with the value of the inputs to the AND array.

Trace level 4 provides the ability to watch the logical value before the output buffer. Using \$TRACE 4, the Simulator only reports the true output pin values, and assigns a "?" to inputs and buried nodes. For combinatorial output, trace level 4 displays the results of the OR term. For registered outputs, trace level 4 shows the Q output of the register. The following example uses a p22v10:

```
pin 1 = CLK;
pin 2 = IN2;
pin 3 = IN3;
....
pin 14 = OUT14;
pin 15 = OUT15;
....
OUT14.D = IN2;
OUT14.AR = IN3;
OUT14.OE = IN4;
```

The simulation result file:

```
order CLK, IN2, IN3, IN4, OUT14, OUT15;
***** before output buffer *****
???? ..LL...0001:
0011 ..HH.....
*****before output buffer*****
????  HH...0004
C100  ...ZZ.....
```

\$EXIT

Use the \$EXIT directive to abort the simulation at any point. Test vectors appearing after the \$EXIT directive are ignored. This directive is useful in debugging registered designs in which a false transition in one vector causes an error in every vector thereafter.

Placing a \$EXIT command after the vector in error directs attention to the true problem, instead of to the many false errors caused by the incorrect transition.

\$SIMOFF

Use the \$SIMOFF simulator directive to turn off test vector evaluation. Test vectors appearing after the \$SIMOFF directive are only evaluated for invalid test values and the correct number of test values. This directive is useful in testing asynchronously clocked designs in which the Simulator is unable to correctly evaluate registered outputs.

\$SIMON

Use the \$SIMON simulator directive to cancel the effects of the \$SIMOFF directive. Test vectors appearing after the \$SIMON directive are evaluated fully.

Advanced Syntax

All the following commands must be placed in the test vectors section of the SI file, after the VECTORS keyword.

Assignment Statement (\$SET)

Syntax: \$SET <variable> = <constant>;

<variable> = <single_sym> | <field> | <defined_variable>

<constant> = <quoted_val> | <tv_string>

<quoted_val> = numbers enclosed in single/double quotes representing inputs/outputs. They will be expanded according to the base in effect and should not contain “don't care” values.

<tv_string> = string of test vector values. The number of values must be equivalent to the number of bits in the variable that they are assigned to.

Action:

It assigns a constant value to a symbol, field or variable. It takes effect immediately, but affects only the user values of the variable; the last simulation results are unchanged. Can appear anywhere in the test vector section.

Example:

```

$set input = '3F'; /* single quotes for inputs */
$set output = "80"; /* double quotes for outputs */
$set Z = HHHH; /* test vector values for a 4-bit
                  output variable */

```

Arithmetic and Logic Operations (\$COMP)

Syntax: \$COMP <variable> = <expression>;

<variable> = <single_sym> | <field> | <defined_variable>

<expression> = any logic or arithmetic expression in which the operands can be variables (like above) or constants.

The allowed constants are decimal numbers (unquoted). Parentheses are permitted.

Operator	Function	Precedence
!	NOT	1
&	AND	2
#	OR	3
\$	XOR	4

Logic Operators

Operator	Function	Precedence
*	multiplication	1
/	division	1
+	addition	2
-	subtraction	2

Arithmetic Operators

The logical and arithmetic operators can be mixed freely in an expression. Normally the logical operators have a higher precedence, however, this rule can be overridden by using parentheses.

Action:

It evaluates the expression and assigns the result to the variable. The current values of the operands (user values) are used in evaluating the expression. Takes effect immediately, but affects only the user values of the variable; the last simulation results are unchanged. Can appear anywhere in the test vector section.

Examples:

```

$COMP A = (!B + C) * A + 1;
$COMP X = (Z / 2) # MASK;

```

Generate Test Vector (\$OUT)

Syntax: \$OUT

Action:

Triggers the simulation for the current values of the symbols and generates a test vector. It is useful when used after the \$set and \$comp command because it allows the previously assigned values to take effect in vector evaluation.

Example:

The following set of commands in the SI file:

```
ORDER: _CLOCK, %3, _OE, %3, shift, %1, input, %2, output;
VECTORS:
0 0 'X' XXXXXXXX LLLLLLLL /* power-on reset state */
$set _CLOCK = C;
$set shift = '0';
$set input = '80';
$set output = "80";
$out;
```

This will produce this result in the SO file:

```
0001: 0 0 XXX XXXXXXXX LLLLLLLL
0002: C 0 000 10000000 HLLLLLLL
```

Conditional Simulation (\$IF)

```
Syntax: $IF <condition> :
        <block_1>
        [ $ELSE :
        <block_2> ]
        $ENDIF;
```

<condition> = <var_list> <logic_operators> <constant>

```
logic operators :
= equal
# not equal
> greater than
< less than
>=greater than or equal to
<=less than or equal to
```

<constant> = <quoted_val> | <tv_string>

<block_1>,<block_2> = any sequence of statements, including test vectors

The \$ELSE branch is optional.

Action:

The condition is evaluated using the current simulation value of the variable. If the result is true, <block_1> is executed; otherwise, if \$ELSE is present, <block_2> is executed. \$ENDIF marks the end of the IF statement.

Looping Constructs

FOR statement

```
Syntax: $FOR <count> = <n1>..<n2> :
        <block>
        $ENDF;
```

<count> = the counter of the FOR loop; it takes values between <n1> and <n2>

<n1>,<n2> = limits for <count> values; should be positive decimal numbers

<block> = any sequence of statements, including test vectors

Action:

Step 1. <count> is initialized with the first value, <n1>

Step 2. execute <block>

Step 3. if <count> = <n2> STOP;

otherwise <count> is incremented by 1 (if <n1> less than <n2>) or decremented by 1 (if <n1> greater than <n2>) then repeat steps 2 and 3.

WHILE Statement

```
Syntax: $WHILE <condition> :
        <block>
        $ENDW;
```

<condition> = same as IF condition

<block> = any sequence of statements

Action:

Step 1: Evaluate condition; if false then STOP
else continue with step 2

Step 2: Execute <block>

Step 3: Continue with step 1

DO..UNTIL Statement

```
Syntax: $DO:
        <block>
        $UNTIL <condition> ;
```

<condition> = same as IF condition

<block> = any sequence of statements

Action:

Step 1: Execute <block>

Step 2: Evaluate condition; if true then STOP,
else continue with step 1

◆ IF and other repetitive statements can be nested; however, the maximum number of nested statements is 10.

MACRO and CALL Statements

Macro Definition

```
Syntax: $MACRO name(<arg_list>);
        <macro_body>
        $MEND;
```

name = the macro name

<arg_list> = symbolic names, separated by commas

<macro_body> = any sequence of statements, except \$MACRO (including macro calls)

Argument names can appear in the macro body wherever a variable name or a constant is allowed. They cannot substitute operators, special characters or reserved words.

Macro Call

```
Syntax: $CALL name(<act_arg_list>);
```

name = the name of a previously defined macro

<act_arg_list> = actual arguments list

The actual arguments can be variable names, constants or even macro arguments, if the CALL statement is placed within a macro body.

Action:

It executes the statements that form the invoked macro body by replacing any occurrence of a macro argument with the corresponding actual argument.

In order to successfully complete a macro call, check if the actual arguments fit the syntax of the macro body, that is they won't cause a syntax error by replacing the corresponding formal argument.

Example:

```
$MACRO m1(a,b,c);          /* Macro definition */
$set shift = a;
$set shift = b;
$set output = c;
$MEND;

$CALL m1('0','80',*****); /* Macro call */
```

The following statements will be executed:

```
$set shift = '0';
$set shift = '80';
$set output = *****;
```

The following is a full example of how these statements work and how they can help the user simulate his design without entering a lot of test vectors.

These two SI files produce the same output:

1. Old way:

```
Name      Barrel22;
Partno    CA0006;
Date      05/11/89;
Revision  02;
Designer  Kahl;
Company   ATI;
Assembly  None;
Location  None;
Device    g20v8a;
ORDER:    _CLOCK, %3, _OE, %3, shift, %1, input, %2, output;
VECTORS:
0  0 'X' XXXXXXXX HHHHHHHH /* power-on reset state */
C  0 '0' 10000000 HLLLLLLL /* shift 0 */
C  0 '1' 10000000 LHLLLLLL /* shift 1 */
C  0 '2' 10000000 LLHLLLLL /* shift 2 */
C  0 '3' 10000000 LLLHLLLL /* shift 3 */
C  0 '4' 10000000 LLLLHLLL /* shift 4 */
C  0 '5' 10000000 LLLLHLHL /* shift 5 */
C  0 '6' 10000000 LLLLHLHL /* shift 6 */
C  0 '7' 10000000 LLLLHLHL /* shift 7 */
C  0 '0' 01111111 LHHHHHHH /* shift 0 */
C  0 '1' 01111111 HLHHHHHH /* shift 1 */
C  0 '2' 01111111 HHLHHHHH /* shift 2 */
C  0 '3' 01111111 HHHLHHHH /* shift 3 */
C  0 '4' 01111111 HHHHLHHH /* shift 4 */
C  0 '5' 01111111 HHHHHLHH /* shift 5 */
C  0 '6' 01111111 HHHHHHLH /* shift 6 */
C  0 '7' 01111111 HHHHHHHL /* shift 7 */
```

2. New way:

```
ORDER:    _CLOCK, %3, _OE, %3, shift, %1, input, %2, output;
VECTORS:
0 0 'X' XXXXXXXX LLLLLLLL /* power-on reset state */
$set _CLOCK = C;
$set shift = '0';
$set input = '80';
$set output = "80";
$for i = 1..16 :
$out;
$if shift = '7':
$set shift = '0';
$set input = '7f';
```

```

$set input = '7f';
$set output = "7f";
$else:
$comp shift = shift + 1;
$comp output = output / 2;
$if input = '7f':
$comp output = output # 128;
$endif;
$endif;
$endif;

```

or, using macros:

```

ORDER:  _CLOCK, %3, _OE, %3, shift, %1, input, %2, output;
VECTORS:
$macro m1(x,y,z);
$set shift = x;
$set input = y;
$set output = z;
$mend;

$macro m2(a,b,c,d);
$call m1(a,b,c);
$for i = 1..8 :
$out; $comp shift = shift + 1;
$comp output = output / 2 + d;
$endif;
$mend;
0 0 'X' XXXXXXXX LLLLLLLL /* power-on reset state */
$set _CLOCK = C;
$call m2('0','80',"80", 0);
$call m2('0','7f',"7f", 128);

```

3. The Output:

```

CSIM: CUPL Simulation Program Version
4.2a Serial# ...

CREATED Wed Dec 04 03:00:11 1991
LISTING FOR SIMULATION FILE: barrel22.si
1: Name Barrel22;
2: Partno CA0006;
3: Date 05/11/89;
4: Revision02;
5: DesignerKahl;
6: Company ATI;
7: AssemblyNone;
8: LocationNone;
9: Device g20v8a;
10:
11: FIELD input = [D7,D6,D5,D4,D3,D2,D1,D0];
12: FIELD output = [Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0];
13: FIELD shift = [S2,S1,S0];
14:
15: ORDER: _CLOCK, %3, _OE, %3, shift, %1, input, %2, output;
16:
17: var X = Q7;
18: var Y = Q7..4;
19:
=====
      _
      C
      L
      O      _
      C      O      shi
      K      E      ft      input      output
=====
0001: 0      0      XXX XXXXXXXX      LLLLLLLL
0002: C      0      000 10000000      HLLLLLLL
0003: C      0      001 10000000      LHLLLLLL
0004: C      0      010 10000000      LLHLLLLL
0005: C      0      011 10000000      LLHLHLLL
0006: C      0      100 10000000      LLLLHLLL
0007: C      0      101 10000000      LLLLHLHL
0008: C      0      110 10000000      LLLLLLHL
0009: C      0      111 10000000      LLLLLLLH
0010: C      0      000 01111111      LHHHHHHH
0011: C      0      001 01111111      HLHHHHHH
0012: C      0      010 01111111      HHLHHHHH

```

0013:	C	0	011 01111111	HHHLHHHH
0014:	C	0	100 01111111	HHHHLHHH
0015:	C	0	101 01111111	HHHHHLHH
0016:	C	0	110 01111111	HHHHHHLH
0017:	C	0	111 01111111	HHHHHHHL

There is one thing the user must keep in mind when creating a simulation input file using the new syntax:

If one or more \$SET or \$COMP commands are placed right before some conditional statement (IF, WHILE, UNTIL) without any intermediate \$OUT statement, the values set by those commands (user values) will not affect the condition value, as the condition is evaluated using the last simulation values of the variables involved.

For example, let's assume that we want to generate the following simulation output:

```
ORDER: _CLOCK,clr,dir,!_OE,%2,count,%1,carry;
var mode = clr,dir;
VECTORS:
C 100 LLLL L /* synchronous clear to state 0 */
C 000 LLLH L /* count up to state 1 */
C 000 LLHL L /* count up to state 2 */
C 000 LLHH L /* count up to state 3 */
C 000 LHLL L /* count up to state 4 */
C 000 LHLH L /* count up to state 5 */
C 000 LHHL L /* count up to state 6 */
C 000 LHHH L /* count up to state 7 */
C 000 HLLL L /* count up to state 8 */
C 000 HLLH H /* count up to state 9 - carry */
```

The following sequence will generate a wrong output:

```
ORDER: _CLOCK,clr,dir,!_OE,%2,count,%1,carry;
var mode = clr,dir;
VECTORS:
C 100 LLLL L $set mode = '0';
$for i=1..9 :
$comp count = count + 1;
$if count="9":
$set carry = H;
$endif;
$out;
$endf;
```

that is:

```
0001: C 100  LLLL L
0002: C 000  LLLH L
0003: C 000  LLHL L
0004: C 000  LLHH L
0005: C 000  LHLL L
0006: C 000  LHLH L
0007: C 000  LHHL L
0008: C 000  LHHH L
0009: C 000  HLLL L
0010: C 000  HLLH H
      ^
[0019sa] user expected (L) for carry
```

This is because the value for count used in the evaluation of the IF condition for vector 10 was the current simulation value (that is the one displayed in vector 9) and not the one set by \$comp command.

The correct sequence is:

```
C 100 LLLL L
$set mode = '0';
$for i=1..9 :
$if count="8":
$set carry = H;
$endif;
$comp count = count + 1;
$out;
$endf;
```

Virtual Simulation

Virtual simulation allows you to create a design without a target device and simulate it. It is possible, therefore, to get a working design before deciding what architecture it will be targeted to. This will be especially useful for designs that will be eventually partitioned.

Usage of the virtual simulator is transparent. You do not need to learn any new commands or syntax. Just use the VIRTUAL device mnemonic when compiling and simulating to take advantage of the Virtual simulator.

Virtual simulation is also used to simulate FPGA designs. When a full architectural simulation is not possible due to the proprietary nature of the device internals or the level of complexity of the internal logic resources, Virtual simulation is the next best alternative for your design verification phase.

Fault Simulation

An internal fault can be simulated for any product term, to determine fault coverage for the test vectors. The format for this option is as follows:

```
STUCKL n ;
```

or

```
STUCKH n ;
```

where

n is the JEDEC fuse number for the first fuse in the product term.

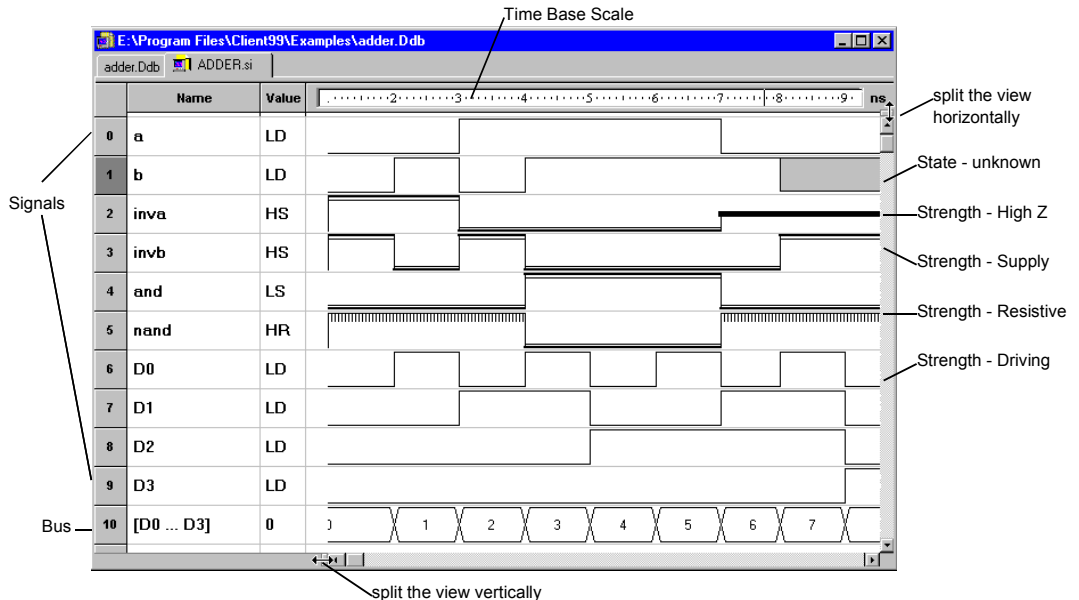
The documentation file (*filename.DOC*) fuse map lists the fuse numbers for the first fuse in each product term in the device.

Format 1 forces the product term to be stuck-at-0.

Format 2 forces the product term to be stuck-at-1. The STUCK command must be placed between the ORDER and VECTORS statements.

Viewing the Simulation Waveforms

Simulation results are normally displayed as waveforms, using the Waveform Editor. The ASCII simulation listing file (*filename.SO*) contains the results of simulation.



The Waveform Editor window

Time Base

The Time Base is displayed in the scale at the top of the waveform window. By default a change of event in the simulation listing file occurs at 1 ns intervals on the waveform.

A vertical marker appears on the Time Base Scale, indicating the current cursor position. The exact time value appears on the left hand end of the Status Line.

Signal Name

Signal names are extracted from the simulation listing file. The name can be edited and the font changed. Double-click on a name to edit.

Signal Value

The Value column reflects the state of each signal at the current time position. The current time position is the left hand edge of the Time Base Scale. As you scroll across the waveform you will notice that the contents of the Value column changes.

Each Value consists of two characters, which have the following meaning:

- L Signal State is Low
- H Signal State is High

- D Signal Strength is Driving
- S Signal Strength is Supply
- R Signal Strength is Resistive
- xx Bus signals display the current value of the bus (in binary, decimal or hex)

Why do the Signals Look Different?

Signals are rendered differently, depending on their Strength. Refer to the figure on the previous page for an example of how each Strength is rendered. Select the View-Legend menu item for an example of each signal strength.

The strength of the displayed signal is determined from the signal type in the simulation listing file.

Timing Marks

The Waveform Editor includes 10 timing marks which can be set at any point, select the Edit-Set Timing Marks sub menu to set the Timing Mark of your choice. Use the Edit-Jump sub menu to jump to the Timing Mark of your choice (shortcut: J, followed by the timing mark number).

Creating a Bus

You can create a Bus from any set of adjacent signals. To do this:

- Click and drag to select the signals that you wish to group as a Bus (they will highlight in black).
- Select the Insert-Bus menu item.

The Bus will be inserted just below the last signal in the selection. The Value column will display the value of the Bus at the current time position. Each signal that was included in the Bus represents one bit in the Bus, with the uppermost selected signal representing the Least Significant Bit.

Editing a Signal

- Double-click on a transition to edit it.
- Click and drag a signal transition to graphically change the Start Time.

Changing Your View

The Waveform Editor includes a number of features to help you examine the waveform.

Scrolling Horizontally and Vertically

Use the scroll bars to scroll back and forth and up and down the waveform window. Alternatively, use the arrow shortcut keys. To scroll up and down in larger steps, use the PAGEUP and PAGEDOWN shortcut keys.

Zooming In and Out

You can change the Time Base Scale (zoom in or out) by selecting the View-Zoom In (shortcut: +) or View-Zoom Out (shortcut: -) menu items. The Zoom In and Zoom Out

processes are relative to the current cursor position. Position the cursor prior to pressing the shortcut keys.

Panning

Press the HOME key to pan across the waveform display. This shortcut key is also relative to the current cursor position. Position the cursor prior to pressing the shortcut key.

Creating Multiple Views

The Waveform Editor supports multiple views, allowing you to view different regions of the waveform simultaneously. The view can be split vertically (to view different points in time simultaneously) and horizontally (to bring together different signals that cannot be viewed simultaneously). Refer to the previous figure for the cursor locations used to split the view. Position the cursor as shown, then click and drag to split the view.

The Waveform Editor Panel

Select the View-Panel menu item to display or hide the Waveform Editor Panel. Use the Panel to step through the transitions in the active signal and to hide and unhide signals.

The Active Signal

The box at the top of the Panel displays the name of the active signal. The Jump Transition buttons allow you to jump (on the active signal) to:

The First transition (shortcut: J, F)

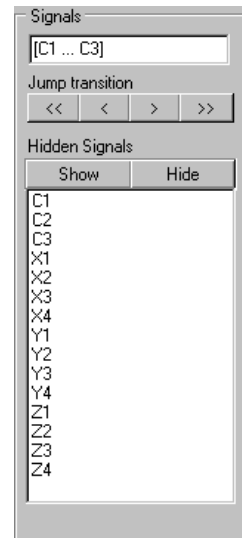
The Previous transition (shortcut: J, P)

The Next transition (shortcut: J, N)

The Last transition (shortcut: J, L)

Hiding and Unhiding Signals

The Waveform Editor allows you to view a sub-set of signals by hiding those signals you do not wish to view. Click and drag to select the signals you wish to hide, then press the Hide button on the Panel.



- ◆ Use the Select button on the main toolbar to select all the signals, press the Hide button to hide them all, then selectively Unhide those signals you wish to view. To build up a selection in the Panel to unhide, hold the SHIFT and CTRL keys as you click on the signal names.

Sample PLD Design Session

This chapter gives step-by-step instructions on how to design a PLD. It shows how to implement the same design, using both schematic capture and CUPL code. It also describes how to create the simulation test specification source file, and simulate the PLD.

The Design Steps

Examine the Design Task

Take a close look at the design that is needed. Remember that the design can be captured either as a schematic, or in CUPL. If you use CUPL then state-machine, Boolean equations, and truth tables are available for the design. Try to determine which type of syntax best suits the design.

Create the Compiler Source File

Use the template file provided and remove the sections that do not apply. Remember to edit the header to reflect the new file that is being created.

Formulate the Equations

Equations must be written in CUPL syntax to specify the logic desired. This can be in Boolean, state-machine, or truth table format.

Choose a Target Device

Make sure that there is a sufficient number of input pins. Check that the number of registered and non-registered output pins is sufficient for the design. Ensure that the device has three-state output control, if needed. Check that the device can adequately handle the number of product terms required. Select the target device in the Configure PLD dialog.

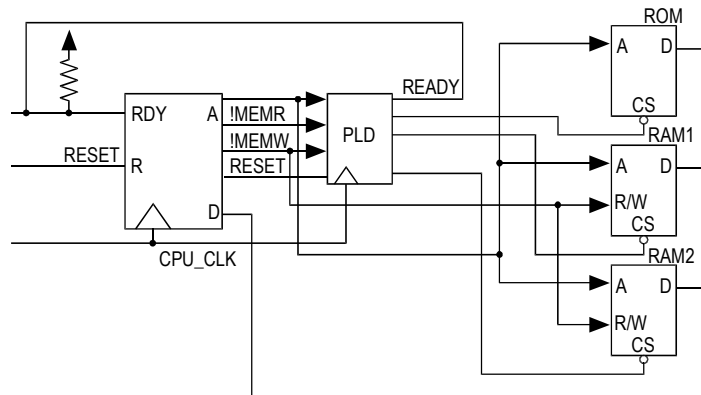
Make Pin Assignments

Assign the inputs and outputs of the design to the pins of the device. Make sure that the device is being used properly by consulting the reference material available from the manufacturer.

Ready to Compile

Decide which file formats will be needed for downloading and simulation. A choice of four minimizers are available. You are now ready to compile your design.

Step 1 - Examining the Design Task

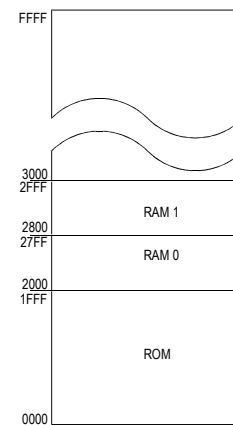


System diagram - Microprocessor-based system using a PLD

In this example a PLD provides a flexible interface between the CPU and peripherals, performing address decoding for chip selects, and timing control functions.

The memory map shows where the ROM and RAM chips reside in the CPU's address space. The PLD will decode these address ranges to generate the chip selects.

Addresses are marked and shown in hexadecimal in the memory map. From the memory map the following table has been created, which shows the state of the Address lines for each address range that is decoded. From the table it can be seen that the PLD must include address lines A11 to A15 as inputs.

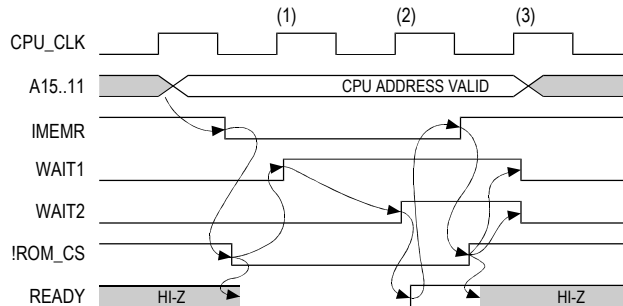


Memory map

Chip Select	A15	A14	A13	A12	A11	A10	A9	A8	A7-A4	A3-A0
ROM (0000-1FFF)	0	0	0	X	X	X	X	X	X	X
RAM0 (2000-27FF)	0	0	1	0	0	X	X	X	X	X
RAM1 (2800-2FFF)	0	0	1	0	1	X	X	X	X	X

Because the ROM chip is slow, the PLD must also be designed to perform a wait state generation that adds at least one CPU clock period to the ROM access.

The worm arrows on the timing diagram show signals affected or created by other signals.



Wait State Generator timing diagram.

A description of the operation of the timing diagram follows. The numbers in parentheses indicate the rising edge of the CLOCK signal.

A wait state sequence starts with the CPU address becoming valid prior to the memory read strobe. Only the !MEMR signal needs to be considered, because the wait state is generated only for the ROM.

When the !MEMR strobe is active for an address corresponding to the ROM, the !ROM_CS signal is asserted and turns on the three-state buffer, driving the CPU READY signal LO, (indicating not ready, or wait). The next rising edge of the CPU clock (1) after !ROM_CS becomes active and sets the WAIT1 signal. After one CPU clock period passes, the WAIT2 signal is asserted (2); the wait state period (one CPU clock) is completed, causing the CPU READY signal to be driven HI, which causes the CPU to continue its read cycle and remove the !MEMR strobe at the appropriate time. The !ROM_CS signal is negated, disabling the three-state buffer driving the ready signal and, at the next rising edge of the CPU clock (3), causing WAIT1 and WAIT2 to be reset. The wait state generator is now prepared for the next CPU access time.

Step 2 - Creating the CUPL Source Code

In this step, a logic description file will be created to describe the design for the PLD. This logic description file is written in the CUPL logic description language. The logic description file serves as input to the Compiler, which compiles the design for downloading to a device programmer.

Create the header section of the PLD file, as shown below. Following the header section is a title block, with comment symbols (*/** and **/*). In the title block, type in information describing the design.

```

                                WAITGEN.PLD
Name                           WaitGen;
Partno                         P9000183;
Date                           07/16/87;
Revision                        02;
Designer                       Osann;
Company                        ATI;
Assembly                       PC Memory;
Location                       U106;
Device                         ; /* the device is selected later */
/*****
/* This device generates chip select signals for one
/* 8Kx8 ROM and two 2Kx8 static RAMs. It also drives
/* the system READY line to insert a wait-state of at
/* least one cpu clock for ROM accesses
*****/

```

Step 3 - Formulating the Equations

To make it easier to enter the specific equations for address decoding and wait state generation, first enter equations for intermediate variables. Intermediate variables are arbitrary names; that is, they do not represent specific pins. Enter the intermediate equations in the space provided in the WAITGEN.PLD file for “Declarations and Intermediate Variable Definitions.”

The first intermediate equation to enter is a bit field declaration to define the address bus. Use the name MEMADR (memory address) to represent the address, and type the equation as follows:

```
FIELD MEMADR = [A15..11] ;
```

In the system diagram shown previously you will notice that the chip select signals for the static RAMs are not dependent solely upon address but must be asserted for either the MEMW or MEMR data strobes.

To simplify the equations for the static RAM chip select signals, create a signal called MEMREQ (memory request). Type the following:

```
MEMREQ = MEMW # MEMR ;
```

Whenever MEMREQ is used in other equations, the Compiler substitutes MEMW # MEMR.

Notice in the timing diagram that the decoding of the address corresponding to the ROM combines with the !MEMR strobe to produce the ROM chip select (ROM_CS), and to initiate the wait state sequence.

Create an intermediate variable, called `SELECT_ROM`, representing the combination of the `!MEMR` strobe and the specific address decoding for the ROM's address space, by typing the following:

```
SELECT_ROM = MEMR & MEMADR : [0000..1FFF] ;
```

After entering the above intermediate equations, the specific equations for address decoding and wait state generation may be entered.

If the signal `ROM_CS`, which feeds back into the array, is being used to initiate the wait state timing, an additional pass delay is incurred through the PLD. Because the clock rate is relatively slow (4-8 MHz), in this example the additional delay is not a problem. However, at higher clock rates it is better to recreate the same logic (using the `SELECT_ROM` intermediate) in the registered logic equations.

Create the ROM chip select (`ROM_CS`) using the intermediate variable `SELECT_ROM`, by typing:

```
ROM_CS = SELECT_ROM ;
```

The chip-selects for the two RAMs, `RAM_CS0` and `RAM_CS1`, are dependent on `MEMREQ` and the address bus being within the hexadecimal boundaries taken from the memory map. Use the CUPL range operation with the lower and upper boundaries of the desired address range to decode these signals. Type the following:

```
RAM_CS0 = MEMREQ & MEMADR : [2000..27FF] ;
RAM_CS1 = MEMREQ & MEMADR : [2800..2FFF] ;
```

Next, create the equations that relate to the wait state timing and generation. First, as shown in the timing diagram, a signal called `WAIT1` is required that responds to the selection of the ROM chip by being set at the next rising edge of the CPU clock. According to the rules for a D-type flip-flop, the logic level at the D input is transferred to the Q output after the clock. Enter the equation for this signal, where `WAIT1.D` represents the signal at the D input of the flip-flop within the PLD, by typing the following:

```
WAIT1.D = SELECT_ROM & !RESET ;
```

Notice that in the equation for `WAIT1.D`, the `!RESET` signal has been ANDed with the rest of the equation to perform a synchronous reset when the `RESET` signal is asserted.

Next, create the signal `WAIT2` at the next clock edge following the one that causes `WAIT1` to set, by making the equation for `WAIT2.D` dependent on the signal `WAIT1`. Since `WAIT2.D` must reset at the next clock edge following the removal of the CPU's access of the ROM, AND the variable `SELECT_ROM` with `WAIT1`. Type the following equation:

```
WAIT2.D = SELECT_ROM & WAIT1 ;
```

This creates the signal `SELECT_ROM` in accordance with the timing diagram, to indicate that the three-state buffer should be turned on while the ROM is being

decoded, and the MEMR data strobe is active. Therefore, enter the equation for the three-state output by typing the following:

```
READY.OE = SELECT_ROM ;
```

While this equation determines when the three-state buffer actually drives its output and leaves the high impedance state, it does not determine which logic level the signal is driven to. The equation for READY determines the logic level to which the signal is driven; the signal should remain inactive at READY until the completion of a wait state period equal to one full CPU clock cycle. As this condition does not occur until WAIT2 becomes set, type the equation for READY as follows:

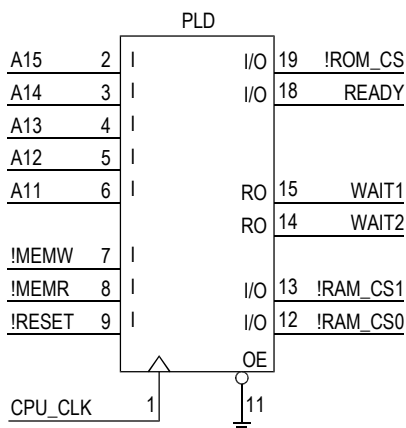
```
READY = WAIT2;
```

Step 4 - Choosing a Target Device

After the equations are completed, the next step is to identify a compatible, commercially available PLD. Points to consider when choosing a target device are:

- The number of input pins required.
- The relative number of registered and non-registered output pins.
- Three-state output control (if required).
- The number of product terms required to implement the logic function for each equation.

The smallest device with sufficient inputs and product terms is a PAL16R4, or alternatively, a GAL16v8 could be used. The PLD package diagram below shows pin assignments configured for this device.



Pin assignments for the PLD

Notice that in the pin assignments the three chip select signals are assigned to I/O type pins, that should always be in the output drive mode. The READY pin, attached to the READY signal on the CPU bus, is used in a controllable three-state mode. The two flip-flops that are needed to implement the wait state generator have been assigned to output pins that are internally connected to registers.

One of the registered outputs could be used to drive the READY signal directly, since the logical function of READY is the same as that of the signal WAIT2. However, use of the dedicated three-state output enable signal connected to pin 11 of the target device would be required. Since pin 11 controls the three state outputs of all four pins connected to internal registers, this defeats the ability to use the other two registered output pins for any purpose other than wait state generation.

It is better to keep options open by not using the dedicated three-state control, since it is difficult to predict all the changes that might be made during the evolution of a design. Therefore, pin 11 is tied to ground, which always enables the three-state outputs coming from the registers.

The PAL16R4 has at least seven product terms available on all outputs, which is an adequate number for this application. Alternatively, a GAL16V8 could be used.

With WAITGEN.PLD as the active document select **PLD » Configure** from the menus, then press the Target Device Change button to pop up the Target Device dialog. Scroll down and select GAL in the Device Type list, then click on g16v8 in the Device Name list. Click OK twice to close both dialogs. The DEVICE line in the Header section of the PLD source will now read:

```
Device      g16v8;
```

Step 5 - Making the Pin Assignments

Match the pin assignments to those shown in the Pin Assignment diagram shown previously.

To ensure consistent documentation when making the pin assignments, be certain that the signal polarities (signal levels) assigned are the same as those in the logic schematic. The pin assignments are shown below:

```

                                WAITGEN PIN ASSIGNMENTS
/**  Inputs  **/
Pin 1      = cpu_clk      ;      /* CPU clock                */
Pin [2..6] = [a15..11]    ;      /* CPU Address Bus         */
Pin [7,8]   = ![memw,memr] ;      /* Memory Data Strokes     */
Pin 9       = reset       ;      /* System Reset            */
Pin 11      = !oe         ;      /* Output Enable            */

/**  Outputs  **/
Pin 19      = !rom_cs     ;      /* ROM Chip select         */
Pin 18      = ready       ;      /* CPU Ready signal        */

```

```

Pin 15      = wait1      ;          /* Start wait state          */
Pin 14      = wait2      ;          /* End wait state          */
Pin [13,12] = ![ram_cs1..0] ;      /* RAM Chip selects        */

```

The completed logic description file:

```

WAITGEN.PLD
Name          WaitGen;
Partno        P9000183;
Date          07/16/87;
Revision      02;
Designer      Osann;
Company       ATI;
Assembly      PC Memory;
Device        gl6v8;

/*****
/* This device generates chip select signals for one
/* 8Kx8 ROM and two 2Kx8 static RAMs. It also drives
/* the system READY line to insert a wait-state of at
/* least one cpu clock for ROM accesses
*****/
/*****
/** Allowable Target Device Types : GAL16V8, PAL16R4 **/
/*****
/** Inputs **/
Pin 1          = cpu_clk;          /* CPU clock          */
Pin [2..6]     = [a15..11];        /* CPU Address Bus    */
Pin [7,8]      = ![memw,memr] ;
                          /* Memory Data Strokes (active low)*/
Pin 9          = reset;            /* System Reset       */
Pin 11         = !oe; /* Output Enable (active low) */

/** Outputs **/
Pin 19         = !rom_cs; /* ROM chip select (active low)*/
Pin 18         = ready      ;      /* CPU ready          */
Pin 15         = wait1      ;      /* Wait state 1       */
Pin 14         = wait2      ;      /* Wait state 2       */
Pin [13,12]    = ![ram_cs1..0] ;
                          /* RAM chip select (active low) */

/* Declarations and Intermediate Variable Definitions */
Field memadr = [a15..11] ; /* Give the address bus */
                          /* the Name "memadr" */

memreq = memw # memr ;      /* Create the intermediate */
                          /* variable "memreq" */

```

```

select_rom = memr & memadr:[0000..1FFF] ; /* = rom_cs */

/** Logic Equations */
rom_cs = select_rom;
ram_cs0 = memreq & memadr:[2000..27FF] ;
ram_cs1 = memreq & memadr:[2800..2FFF] ;

/* read as: when select_rom is true and reset is false */
wait1.d = select_rom & !reset ;

/* read as: when when select_rom is true and wait1 is true */
/* Synchronous Reset */
wait2.d = select_rom & wait1 ; /* wait1 delayed */

ready.oe = select_rom ; /* Turn Buffer off */
ready = wait2 ; /* end wait */

```

Step 6 - Compiling the PLD Source File

In this step, the logic description file WAITGEN.PLD is compiled for the target device GAL20V8.

Specify the target PLD and other compiler options in the Configure PLD dialog. Enable the following options:

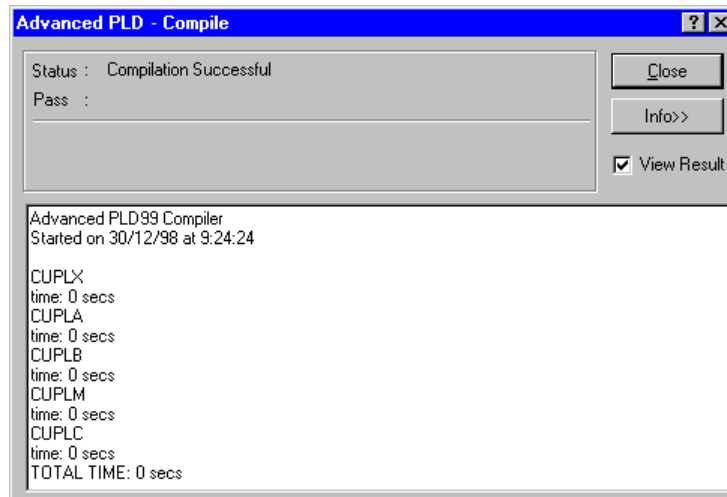
- Absolute ABS - Creates WAITGEN.ABS. This is the absolute file for later use by the Simulator (This file is needed for step 8). It contains a condensed representation of the logical function to be programmed into the device. The Simulator compares this representation to test vectors in a user-created input file to determine whether the response vectors in the input file are a correct response to the stimulus vectors.
- Fuse Plot in Doc File and Equations in Doc File - Creates WAITGEN.DOC. This is the documentation file. It provides fully expanded product terms for both intermediate and output pin variables, and a fuse plot and chip diagram.
- Error List LST - Creates WAITGEN.LST. This is the list file. It is a recreation of the description file, except line numbers have been added and any error messages generated during compilation are appended at the end of the file.
- JEDEC - Creates WAITGEN.JED. This is a JEDEC file for downloading to a device programmer. It contains a fuse pattern. Test vectors will be added to the JEDEC file during simulation. The name of the JEDEC file is determined by the NAME field in the Header section.

Once you have configured the Compiler, press OK in the Configure PLD dialog.

To compile the design make WAITGEN.PLD the active document, and press the Compile button on the PldTools toolbar.

◆ Save the PLD source code before you run the Compiler.

The PLD - Compiling dialog will pop up. Press the Info button. The following messages appear in the dialog, indicating how much time each Compiler module takes for completion. The actual time will vary depending on the system being used.



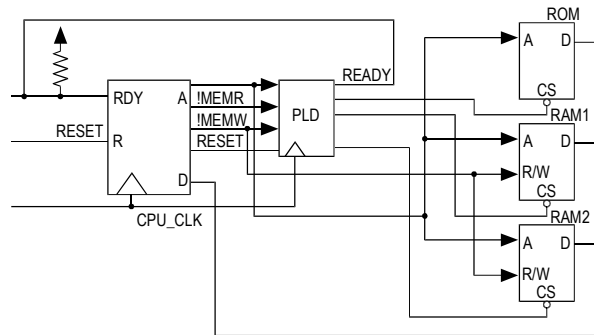
The compiler will produce each of the output formats that have been enabled. If the View Result option is enabled the LST, DOC and JEDEC files will be opened automatically.

The list file, WAITGEN.LST, is essentially a recreation of the source file with line numbers inserted and any error messages added to the end. The line numbers facilitate the quick locating of error sources, if any are detected by the Compiler.

The documentation file, WAITGEN.DOC, includes the expanded product terms, the symbol table, the fuse plot and a chip diagram detailing the pinouts.

Step 7 – Creating a Schematic-based version of the Design

Programmable logic designs implemented as a schematic must be based on the symbols in the PLD Symbols.lib library, which is in the \Program Files\Design Explorer 99 SE\Library\Pld.ddb Library Database. This library includes a comprehensive range of symbols, suitable for both PLD, CPLD, and FPGA design. It is important that you are familiar with the functionality of the target device before implementing your design, and ensure that the design only includes symbols that can be compiled into the chosen device.



System diagram

Chip Select Logic

There are numerous ways of implementing the chip select decoding logic. This example uses a number of smaller logic symbols, so that it is easier to understand.

The table below shows the state of each of the address lines, for each of the chip selects that the PLD decodes.

Chip Select	A15	A14	A13	A12	A11	A10	A9	A8	A7-A4	A3-A0
ROM (0000-1FFF)	0	0	0	X	X	X	X	X	X	X
RAM0 (2000-27FF)	0	0	1	0	0	X	X	X	X	X
RAM1 (2800-2FFF)	0	0	1	0	1	X	X	X	X	X

Using the information in the table and the system diagram, we could write the following general equations:

$$\text{ROM\_CS} = \neg A_{15} \ \& \ \neg A_{14} \ \& \ A_{13} \ \& \ \neg \text{MEMR}$$

$$\text{RAM\_CS0} = \neg A_{15} \ \& \ \neg A_{14} \ \& \ A_{13} \ \& \ \neg A_{12} \ \& \ \neg A_{11} \ \& \ (\neg \text{MEMR} \ \# \ \neg \text{MEMW})$$

$$\text{RAM\_CS1} = \neg A_{15} \ \& \ \neg A_{14} \ \& \ A_{13} \ \& \ \neg A_{12} \ \& \ A_{11} \ \& \ (\neg \text{MEMR} \ \# \ \neg \text{MEMW})$$

It is not possible to implement the RAM chip select equations directly as they contain a mixture of ANDs and ORs, so we must first create intermediate signals.

From the table you will notice that all the chip selects require A15 and A14 to be low. These can be combined using an AND2B2 from the Symbol library, giving the following expression:

$$\text{ADR\_15\_14} = !A15 \ \& \ !A14$$

(when A15 and A14 are both low then assert ADR_15_14)

Both RAM selects require A13 to be high and A12 to be low. These can be combined using an AND2B1, giving the following expression:

$$\text{ADR\_13\_12} = !A13 \ \& \ A12$$

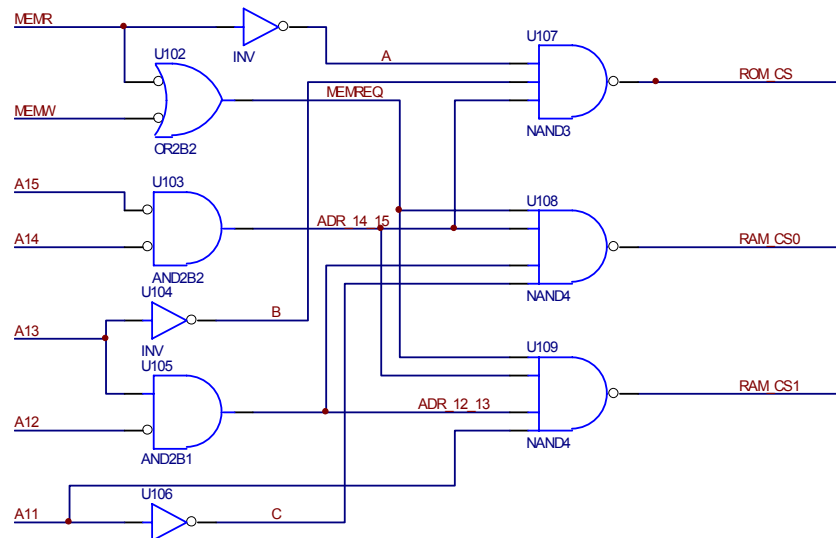
(when A13 is high and A12 is low then assert ADR_13_12)

The memory read and write can also be used to create a memory request, which will be used to qualify the 2 RAM chip selects. This is done with an OR2B2:

$$\text{MEMREQ} = !\text{MEMR} \ \& \ !\text{MEMW}$$

(when either MEMR or MEMW is low then assert MEMREQ)

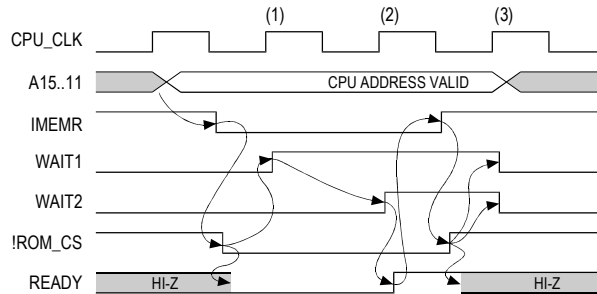
These intermediate signals can now be combined with the other address lines to create the 3 chip selects. The circuit is shown below.



Chip select circuit

Wait State Logic

Refer to the timing diagram as the wait state logic is described.



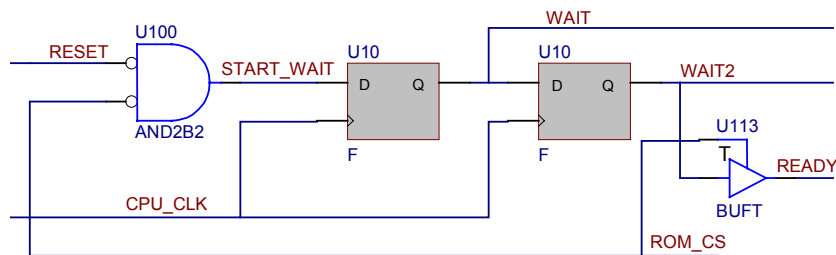
Wait State Generator timing diagram

WAIT1 is required to respond to the assertion of the ROM_CS, by being set at the next rising edge of the CPU clock. Before this is implemented using flip-flops, note that the system diagram shows that the PLD must also be able to be reset by the CPU Reset signal. The active-high Reset can be incorporated into the wait state generator, by ANDING it with ROM_CS. The equation that expresses this is:

$$\text{START\_WAIT} = \text{!ROM\_CS} \ \& \ \text{!RESET} \ ;$$

The wait state generator must ensure that the READY line is not asserted for at least one full clock cycle. WAIT1 will become active on the first rising edge of the CPU_CLK after ROM_CS asserts, we now want a signal that asserts on the next rising edge of the CPU_CLK. This is achieved by feeding the output of the first flip-flop, into the D input of a second flip-flop. This 2nd flip-flop's output (WAIT2) will go high on the next rising edge of the CPU_CLK.

Note that the timing diagram shows that the READY signal back to the CPU must be Hi-Z after completion of the ROM access. To achieve this a three-state buffer is used, whose output is enabled by the ROM_CS. The following circuit diagram shows how the wait state generator is implemented:



Wait State Generator circuit

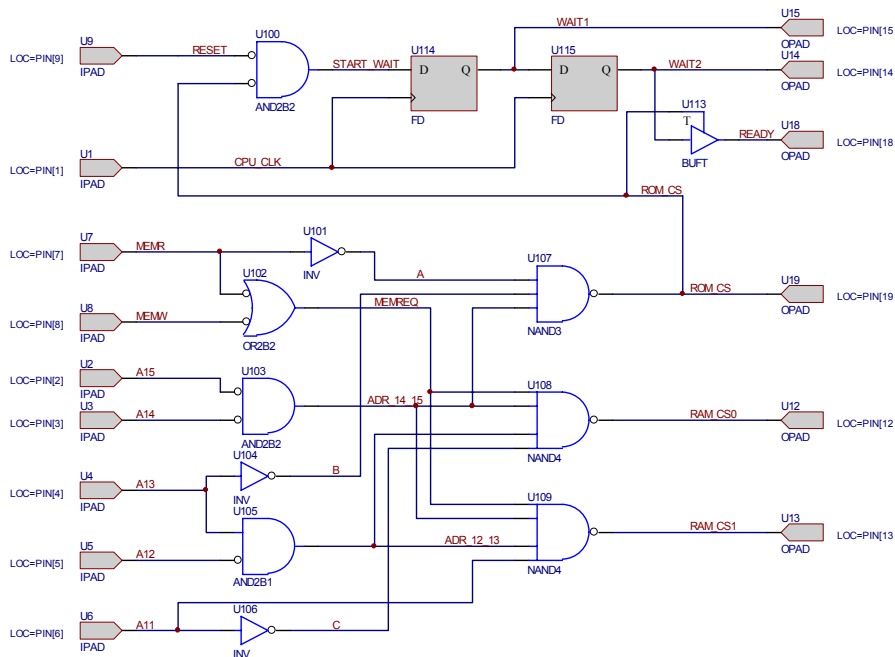
Interfacing the Internal Logic to the Component Pins

The last step in implementing the design in a PLD is to interface from the internal logic to the actual device pins. There are special “PAD” symbols in the Symbols library for this, IPADs, OPADs, and IOPADs. Place one of the appropriate pads at each point that connects to a component pin. The number of the component pin that the pad connects to is specified in Part Field 1 of the pad symbol, using the following syntax:

LOC=PIN[*pin_number*]

After placing the appropriate pad symbols and numbering them according to what the target device supports, the completed circuit looks like:

◆ Refer to the topic, *Interfacing from Internal Logic to Component Pins*, in the chapter, *Schematic-based PLD Design Entry*, for details on using the PAD components.



Notes:

- The completed design is compiled in the same way as the CUPL-based version, refer to *Step 6 - Compiling the PLD Source File*, for details.
- All internal logic must have unique designators, there are no specific requirements for assigning them.
- Assigning net names to all internal nets can assist in debugging.
- Internal net names must be unique, and cannot be the negated value of any other net name.
- You can find this circuit in the PLD examples.

Step 8 - Creating a Simulation Test Vector File

In this step, a simulation will be performed to verify the compiled design for the GAL16V8 device. Performing this step before downloading to a logic programmer decreases the probability of programming a device with incorrect logic.

In this section a source specification file, WAITGEN.SI, is created. It contains test vectors for input to the Simulator. The Simulator compares the test vector inputs and expected outputs to the actual values contained in the WAITGEN.ABS file that was created during Compiler operation, and flags any discrepancies.

The contents of a WaitGen source specification file.

```
Name                WaitGen;
Partno              P9000183;
Date                07/16/87;
Revision            02;
Designer            Osann;
Company             ATI;
Assembly            PC Memory;
Location            g20v8;

/*****
/* This device generates chip select signals for one */
/* 8Kx8 ROM and two 2Kx8 static RAMs. It also drives */
/* the system READY line to insert a wait-state of at */
/* least one cpu clock for ROM accesses                */
*****/

ORDER:
    cpu_clk, %2, a15, %2, a14, %2,
    a13, %2, a12, %2, a11, %2,
    !memw, %2, !memr, %2, reset, %2, !oe,
    %4, !ram_cs1, %2, !ram_cs0, %2, !rom_cs, %2,
    wait1, %2, wait2, %2, ready;

VECTORS:
/* 123456-leave six blanks to allow for numbers in .SO file */
$msg "      Power On Reset                                ";
      O X X X X X 1 1 1 0      H H H * * Z
$msg "      Reset Flip Flops                                ";
      C X X X X X 1 1 0 0      H H H L L Z
$msg "      Write RAM0                                        ";
      0 0 0 1 0 0 0 1 0 0      H L H L L Z
$msg "      Read RAM0                                         ";
```

```

      0 0 0 1 0 0 1 0 0 0   H L H L L Z
$msg "      Write RAM1                                     ";
      0 0 0 1 0 1 0 1 0 0   L H H L L Z
$msg "      Read RAM1                                      ";
      0 0 0 1 0 1 1 0 0 0   L H H L L Z
$msg "      Begin ROM read                                ";
      0 0 0 0 0 0 1 0 0 0   H H L L L L
$msg "      Two clocks for wait state, Then drive READY High ";
$repeat2;
      C 0 0 0 0 0 1 0 0 0   H H L * * *
$msg "      End ROM Read                                   ";
      0 0 0 0 0 1 1 0 0   H H H H H Z
$msg "      End ROM Read                                   ";
      C 0 0 0 0 1 1 0 0   H H H L L Z

```

The source specification file contains three major parts: header information and title block, an ORDER statement, and a VECTORS statement.

WAITGEN.SI must have the same header information as WAITGEN.PLD to ensure that the proper files, including current revision level, are being compared against each other. Save WAITGEN.PLD as WAITGEN.SI and delete everything in WAITGEN.SI, except the header and title block. Below is the result.

```

Name          WaitGen;
Partno        P9000183;
Date          07/16/87;
Revision      02;
Designer      Osann;
Company       ATI;
Assembly      PC Memory;
Location      U106;

/*****
/* This device generates chip select signals for one */
/* 8Kx8 ROM and two 2Kx8 static RAMs. It also drives */
/* the system READY line to insert a wait-state of a */
/* least one cpu clock for ROM accesses             */
*****/

```

In the ORDER statement, list the input and output variables from WAITGEN.PLD to be included in test vectors. List the variables in the order in which they will be used in test variables; that is, put the clock variable, CPU_CLK, first, followed by the other input variables. Put the output variables to the right. Separate variables with commas. Use the % symbol to insert spaces between the variables; put two spaces between each variable, and four spaces between the last input variable in the list, !OE, and the first output variable, !RAM_CS1. Type the ORDER statement as follows:

ORDER:

```
CPU_CLK, %2, A15, %2, A14, %2,
A13, %2, A12, %2, ALL, %2,
!MEMW, %2, !MEMR, %2, RESET, %2, !OE,
%4, !RAM_CS1, %2, !RAM_CS0, %2, !ROM_CS, %2
WAIT1, %2, READY;
```

Following the ORDER statement, enter a VECTORS statement that creates a function table containing eleven test vectors. To make the vectors easier to understand, a header for the test vectors is generated and placed in the .SO file right after the ORDER statement. It consists of the symbol names that appear in the ORDER statement, aligned in a manner that increases the readability of the vector section.

Now enter the test vectors. Create the vectors by assigning a value to each of the input variables and an expected value to each of the output variables.

◆ Refer to the topic *Creating a Test Specification Source File* in the chapter *Simulating the Design* for more information on defining text vectors.

Use the \$MSG directive to describe the device function tested by the function. The ORDER statement above specifies the spacing when creating the test vectors. For example, create the first vector, Power On Reset, by typing:

```
$msg " Power On Reset          ";
0 X X X X X 1 1 1 0 H H H * * Z
```

Note that the output value (*) has been used for WAIT1 and WAIT2 to instruct the Simulator to calculate the power-on state of the registers, since some devices power-on to X and some to H or L. Using the asterisk gives a more universal simulation file.

Type in the rest of the test vectors, as shown below.

```
/* 123456-leave six blanks to allow for numbers in .SO file */
$msg "      Power On Reset          ";
      0 X X X X X 1 1 1 0      H H H * * Z
$msg "      Reset Flip Flops          ";
      C X X X X X 1 1 0 0      H H H L L Z
$msg "      Write RAM0          ";
      0 0 0 1 0 0 0 1 0 0      H L H L L Z
$msg "      Read RAM0          ";
      0 0 0 1 0 0 1 0 0 0      H L H L L Z
$msg "      Write RAM1          ";
      0 0 0 1 0 1 0 1 0 0      L H H L L Z
$msg "      Read RAM1          ";
      0 0 0 1 0 1 1 0 0 0      L H H L L Z
$msg "      Begin ROM read          ";
      0 0 0 0 0 0 1 0 0 0      H H L L L L
$msg " Two clocks for wait state, Then drive READY High          ";
$repeat2;
```

	C	0	0	0	0	0	1	0	0	0	H	H	L	*	*	*
\$msg "	End ROM Read															
	0	0	0	0	0	0	1	1	0	0	H	H	H	H	H	Z
\$msg "	End ROM Read															
	C	0	0	0	0	0	1	1	0	0	H	H	H	L	L	Z

The \$REPEAT directive in the test vectors causes the eighth vector to be repeated twice. The asterisks in the eighth vector for WAIT1, WAIT2, and READY tell the Simulator to compute the output based on the inputs and place the results in the output file.

The value of the clock variable, CPU_CLK, is 0 in some vectors and C in others. A value of 0 causes no clocking to occur. A value of C causes the Simulator to examine the input values in the vector and also look back to the previous vector for any registered outputs that would be fed back internally prior to the clock. Then, after a clock is applied, the Simulator computes the appropriate expected outputs for registered and non-registered variables.

After putting in the VECTORS statement, save the file. The next step is to run the Simulator to perform the simulation.

Step 9 - Simulating the Device

When the Simulator is run, it creates WAITGEN.SO, which contains the result of the simulation. Error messages are listed following each vector, with the signal name in error displayed.

To run the Simulator, make WAITGEN.PLD the current document and press the Simulate button on the PldTools toolbar.

Press the Info button in the PLD - Simulate dialog to display simulation information. Enable the View Results check box to automatically load the simulation results into the Waveform Editor. WAITGEN.SO is an ASCII file, so it also possible to open it with a text editor.

The contents of WAITGEN.SO.

```

                                WAITGEN.SO
CSIM:      CUPL Simulation Program
Version 4.XX Serial # XX-XXX-XXXX

CREATED Thur Aug 20 09:34:16 1990
 1: Name                      WaitGen;
 2: Partno                    P9000183;
 3: Date                      07/16/87;
 4: Revision                  02;
 5: Designer                  Osann;
 6: Company                  ATI;
 7: Assembly                  PC Memory;
 8: Location                  U106;
 9:
10: /*****
11: /* This device generates chip select signals for one */
12: /* 8Kx8 ROM and two 2Kx8 static RAMs. It also drives */
13: /* the system READY line to insert a wait-state of at */
14: /* least one cpu clock for ROM accesses                */
15: /*****/
16:
17: ORDER:
18:     cpu_clk, %2, a15, %2, a14, %2,
19:     a13, %2, a12, %2, a11, %2,
20:     !memw, %2, !memr, %2, reset, %2, !oe,
21:     %4, !ram_cs1, %2, !ram_cs0, %2, !rom_cs, %2,
22:     wait1, %2, wait2, %2, ready;
23:
=====
                                Simulation Results
=====

```

										!	!									
c										r	r	!								
p										a	a	r								
u										m	m	o	w	w	r					
—										m	m	e								
c	a	a	a	a	a	e	e	s	!	c	c	—	i	i	a					
l	1	1	1	1	1	m	m	e	o	s	s	c	t	t	d					
k	5	4	3	2	1	w	r	t	e	1	0	s	1	2	y					
Power On Reset																				
0001:	O	X	X	X	X	X	1	1	1	0	H	H	H	X	X	Z				
Reset Flip Flops																				
0002:	C	X	X	X	X	X	1	1	0	0	H	H	H	L	L	Z				
Write RAM0																				
0003:	0	0	0	1	0	0	0	1	0	0	H	L	H	L	L	Z				
Read RAM0																				
0004:	0	0	0	1	0	0	1	0	0	0	H	L	H	L	L	Z				
Write RAM1																				
0005:	0	0	0	1	0	1	0	1	0	0	L	H	H	L	L	Z				
Read RAM1																				
0006:	0	0	0	1	0	1	1	0	0	0	L	H	H	L	L	Z				
Begin ROM read																				
0007:	0	0	0	0	0	0	1	0	0	0	H	H	L	L	L	L				
Two clocks for wait state, Then										drive READY High										
0008:	C	0	0	0	0	0	1	0	0	0	H	H	L	H	L	L				
0009:	C	0	0	0	0	0	1	0	0	0	H	H	L	H	H	H				
End ROM Read																				
0010:	0	0	0	0	0	0	1	1	0	0	H	H	H	H	H	Z				
End ROM Read																				
0011:	C	0	0	0	0	0	1	1	0	0	H	H	H	L	L	Z				

Compare WAITGEN.SO to the WAITGEN.SI file. Note that vectors 8 and 9 were created as a result of the \$REPEAT directive, and that the Simulator has replaced the asterisks from WAITGEN.SI with the appropriate logic levels (H and L) for the WAIT1, WAIT2 and READY signals.

Now that a successful simulation has been completed, test vectors can be added to the JEDEC file created while running the Compiler (in step 6). Run the simulation again with the JEDEC option enabled in the Configure PLD dialog.

The contents of WAITGEN.JED, which now contains both programming and testing information.

WAITGEN.JED	
CUPL	4.XX Serial# XX-XXX-XXXX
Device	pl6r4 Library DLIB-d-26-11
Created	Thur Aug 20 09:52:02 1990
Name	WaitGen

```

Partno                P9000183
Revision              02
Date                  12/16/89
Designer              Osann
Company               ATI
Assembly              PC Memory;
Location              U106;
*QP20
*QF2048
*G0
*F0
*L00000 11111111111111111111111111111111
*L00032 101110111011111111111111111110111111
*L00256 101110111011111111111111111110111111
*L00288 111111111111111111111111111101111111
*L01024 111111111111111111111111111101111111
*L01056 011111111111111111111111111111111111
*L01088 111101111111111111111111111111111111
*L01120 111111110111111111111111111111111111
*L01152 111111111111111111111111111111110111
*L01280 111111111111111111111111111101111111
*L01312 011111111111111111111111111111111111
*L01344 111101111111111111111111111111111111
*L01376 111111110111111111111111111111111111
*L01408 111111111111111111111101111111111111
*L01536 111111111111111111111111111111111111
*L01568 101110110111101101111011111111111111
*L01600 101110110111101101111111111110111111
*L01792 111111111111111111111111111111111111
*L01824 101110110111101110111011101111111111
*L01856 101110110111101110111111111110111111
*C4D50
*V0001 0XXXXX111N0HHXXXXZHN
*V0002 CXXXXX110N0HLLXXZHN
*V0003 000100010N0LHLLXXZHN
*V0004 000100100N0LHLLXXZHN
*V0005 000101010N0HLLXXZHN
*V0006 000101100N0HLLXXZHN
*V0007 000000100N0HLLXXLLN
*V0008 C00000100N0HLLHXXLLN
*V0009 C00000100N0HHHXXHLN
*V0010 000000110N0HHHXXZHN
*V0011 C00000110N1HLLXXZHN
*3152

```

Step 10 - Viewing the Simulation Waveforms

You can view the simulation results at any time by double-clicking on the WAITGEN.SO icon.. The simulation results will be displayed as a series of waveforms, in a spreadsheet style window. For tips on using the Waveform Editor refer to the topic *Viewing the Simulation Waveforms* in the chapter *Simulating the Design*.

Summary

This chapter provided an example of how to move from a conceptual design, through to a PLD source file (or schematic), compile the design, create a simulator test specification file, and simulate the design. The important points were how to:

- Examine the design task and extract the necessary information.
- Choose a PLD and make pin assignments.
- Write intermediate and logic equations to describe the design.
- Run the Compiler.
- Create a schematic-based version of the design.
- Create and compile a test specification file to verify the design.
- Run the Simulator to simulate a logic design.
- View the Simulation Waveforms.

PLD Design Examples

This chapter contains design examples that show how the CUPL language is used to describe different types of designs. The examples are:

Example 1	Simple gates (GATES.PLD)
Example 2	Two-bit counter (FLOPS.PLD)
Example 3	Simple State Machine Design
Example 4	Decade up/down counter using state-machine syntax (COUNT10.PLD)
Example 5	Seven-segment display decoder (HEXDISP.PLD)
Example 6	4-Bit Counter with Load and Reset

The logic description file for each design is shown in parentheses. Any of these files can be compiled to generate documentation or download files.

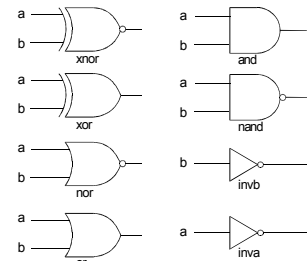
A corresponding test specification file (*filename.SI*) is provided for each logic description file, so that the Simulator can be run to verify the designs.

Example 1 - Simple Logic Gates

This section will explain in detail how to write a simple gates program for a PLD. The diagram shows what gates will be implemented.

This design gives a simple set of inputs and generates output simulating some simple gates. The outputs are labeled to reflect the function of their gate; for example, the AND gate has an output labeled AND.

Below is the CUPL source code that describes the design.



Simple Gates example

```

                GATES.PLD
Name            Gates;
Partno          CA0001;
Date            07/16/87;
Designer        G Woolheiser;
Company         ATI;
Location        San Jose, CA.;
Assembly        Example

/*****
/*
/* This is an example to demonstrate how the PLD
/* compiler compiles simple gates
/*
/*
/* Target Devices: P16L8, P16P8, EP300, and 82S153
/*
*****/

/* Inputs:   define inputs to build simple gates
Pin 1 = a;
Pin 2 = b;

/* Outputs:   define outputs as active HI levels

                For PAL16L8 and PAL16LD8, De Morgan's Theorem is
                applied to invert all outputs due to fixed
                inverting buffer in the device.

Pin 12 = inva;
Pin 13 = invb;
Pin 14 = and;

```

```

Pin 15 = nand;
Pin 16 = or;
Pin 17 = nor;
Pin 18 = xor;
Pin 19 = xnor;

/* Logic:  examples of simple gates expressed in CUPL */

inva = !a;  /* inverters  */
invb = !b;
and  = a & b;      /* and gate  */
nand = !(a & b);   /* nand gate */
or   = a # b;      /* or gate   */
nor  = !(a # b);   /*nor gate   */
xor  = a $ b;      /*exclusive or gate */
xnor = !(a $ b);   /*exclusive nor gate */

```

Simple Gates Source File (GATES.PLD)

The first part of the file provides archival information and a description of the intended function of the design, including compatible PLDs. First, there is the 'Name' line, which the Compiler uses to name the output files by adding extensions. 'Partno' specifies the Company's proprietary part number, issued by manufacturing, for a particular PLD design. The part number is not the type of the target PLD. 'Date' is used to specify the date of compilation. The date should be changed to the current date as good documentation practice. 'Designer' should be the designer's name or the name of the design team. 'Assembly' is used to specify the assembly name or number of the PC board on which the PLD will be used. Use the abbreviation ASSY if desired. 'Location' is supposed to be used to indicate the PC board reference or coordinate where the PLD is located. The abbreviation LOC may also be used. This may be used for other purposes.

Pin declarations are made corresponding to the inputs and outputs in the design diagram. The gates in this example require two inputs, which are passed through the gates as necessary. 'a' and 'b' are names for the input pins. Next, names are assigned to the output pins. The names chosen are descriptive of the function being performed. The use of descriptive names is encouraged, as it makes files easier to debug and update at a later time.

In the "Logic" section of the file, equations describe each of the gates in the design. Boolean syntax is used to specify each output pin as a function of the input pins 'a' and 'b'.

For the PAL16L8 and PAL16LD8 devices, which contain fixed inverting buffers, the Compiler applies DeMorgan's Theorem to invert all outputs because they were all declared active-HI in the pin list. For example, the Compiler converts the following equation for an OR gate, on an output pin that has been declared as active-HI:

```
or = a # b ;
```

to the following single expanded product term (as shown in the documentation file):

```
or => !a & !b
```

The devices chosen for this design were selected because they fit the criteria, as specified earlier, for choosing a device. They have an adequate number of pins, both input and output. They have tri-state control. The number of registered and non-registered pins fits our design, and the device can handle the number of product terms.

Compiling the Source File

In this section, the logic description file (GATES.PLD) is compiled. Files are created for documentation, downloading to a device programmer and for later use by the simulator.

To compile the file GATES.PLD:

- From the Configure PLD dialog enable the following options:

Equations in Doc File	This will generate a documentation file (GATES.DOC) that contains expanded logic equations, a variable symbol table and product term utilization so that the expanded listing generated by the Compiler can be viewed.
Fuse Plot in Doc File	This will add fusemap information to the .DOC file.
Absolute ABS.	This will create the file GATES.ABS for later use by the Simulator.
Jedec.	This will generate the file GATES.JED for input to a device programmer and later use by the Simulator.

- Press the Change button to specify the target device. In the Target Device dialog highlight Device Type 19 and select device P16L8.
- Press OK to close the dialogs.
- Make the source .PLD file (GATES.PLD) the current document
- Press the Compile button on the PldTools toolbar to start the compile process.

The target device is a PAL16L8 and the source file is GATES.PLD. The output files will be GATES.DOC, GATES.ABS and GATES.JEC.

After running the Compiler open the file GATES.DOC. GATES.DOC contains the expanded listing generated. This shows how the Compiler expands the logic equations when it compiles the design for the device chosen.

In order to see how the Compiler reports errors

- Edit the source file GATES.PLD and remove the semicolon at the end of one of the assignment statements.

- Save the file.
- To create an error listing file, enable the Error list LST format in the Configure PLD dialog, then run the Compiler.

After the Compiler has finished, examine the file GATES.LST. Notice that an error line appears next to the error, and there are line numbers at the beginning of each line.

Simulating the Design

In this section the GATES.PLD logic design is simulated with the PLD Simulator. Test vectors are appended to the GATES.JED file created during compilation.

A test specification source file (*filename.SI*) is the input to the Simulator. For this example, the file GATES.SI is included in the example designs. This file has a description of the function of the device in the circuit. For more information about creating a test specification file refer to the *Creating a Simulation Test Vector File* topic in the *PLD Simulation* chapter.

Test vectors specify the expected functional operation of a PLD by defining the outputs as a function of the inputs. Test vectors are also used to do functional testing of a device once it has been programmed, to see if the device functions as expected.

The Simulator compares the input pin stimuli and output pin test values entered in the GATES.SI file (as shown below), with the actual values calculated from the logic equations in the CUPL source file. These calculated values are contained in the absolute file GATES.ABS, which was created during Compiler operation.

To simulate do the following:

- Make Gates.PLD the active document in the Design Explorer. The file Gates.SI must be present in the current directory. Do not attempt to run the Simulator with the .SI file as the current document.
- Press the Simulate button on the PldTools toolbar.

Structured test vectors generated by the simulation will automatically be appended onto the JEDEC download file created during Compiler operation.

Simulator input file (GATES.SI).

```
Name      Gates;
Partno     000000;
Revision   03;
Date       9/12/83;
Designer   CUPL Engineering;
Company    ATI;
Location   None;
Assembly   None;
/*****
/*
```

[illegible]

Gates Simulator Input File (GATES.SI)

Below is the Simulator output file (GATES.SO). The inputs are listed with the corresponding outputs.

```
1:Name          Gates;
2:Partno        000000;
3:Revision      03;
4:Date          9/12/83;
5:Designer      CUPL Engineering;
6:Company       ATI;
7:Location      None;
8:Assembly      None;
```

```

9:
10:/*****/
11:/*
12:/*      This is a example to demonstrate how the Compiler */
13:/*      compiles simple gates.                               */
14:/*
15:/*****/
16:/* Target Devices:P16L8, P16LD8, P16P8, EP300, and 82S153 */
17:/*****/
18:
19:
20:/*
21: * Order:  define order, polarity, and output
22: * spacing of stimulus and response values
23: */
24:
25:Order:  a, %2, b, %4, inva, %3, invb, %5, and, %8,
26:        nand, %7, or, %8, nor, %7, xor, %8, xnor;
27:
28:/*
29: * Vectors:  define stimulus and response values, with header
30: *           and intermediate messages for the simulator listing.
31: *
32: * Note: Don't Care state (X) on inputs is reflected in
33: *       outputs where appropriate.
34: */
35:
=====
                        Simulation Results
=====
                        Simple Gates Simple Simulation

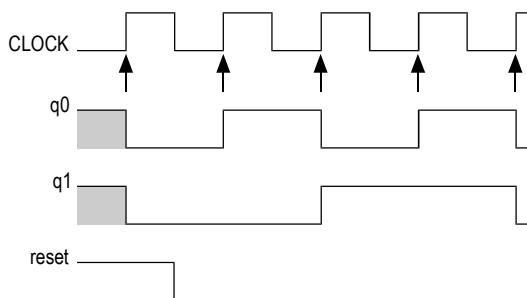
```

			inverters		and	nand	or	nor	xor	xnor
	a	a	!a	!b	a&b	!(a&b)	a#b	!(a#b)	a\$b	!(a\$b)
	-	-	-	-	-	-	-	-	-	-
0001:	0	0	H	H	L	H	L	H	L	H
0002:	0	1	H	L	L	H	H	L	H	L
0003:	1	0	L	H	L	H	H	L	H	L
0004:	1	1	L	L	H	L	H	L	L	H
0005:	1	X	L	X	X	X	H	L	X	X
0006:	X	1	X	L	X	X	H	L	X	X
0007:	0	X	H	X	L	H	X	X	X	X
0008:	X	0	X	H	L	H	X	X	X	X
0009:	X	X	X	X	X	X	X	X	X	X

Gates Simulator Output File (GATES.SO)

Example 2 - Two-Bit Counter

This example demonstrates the implementation of a two-bit counter for a D-type flip-flop. The timing diagram for the counter is shown below:



Two-Bit Counter Timing diagram

As indicated by the arrows, the registers are clocked on the rising edge of the clock signal.

Below is the CUPL source code that describes the two-bit counter design (refer to FLOPS.PLD in the examples).

```

FLOPS.PLD
Name           Flops;
Partno         CA0002;
Revision       02;
Date           07/16/87;
Designer       G. Woolheiser;
Company        ATI;
Location       None;
Assembly       None;

/*****
/*
/* This example demonstrates the use of D-type flip-flop
/* to implement a two bit counter using the following
/* timing diagram.
/*
/*
/* clock  |__|__|__|__|__|__|__|__|
/*
/*
/* q0      ___|_____|_____|_____|_____|__
/*
/*
/* q1      ___|_____|_____|_____|_____|__
/*
/*
/*
*/

```



```

/* reset          |_____ */
/*                                                         */
/*****/
/*      Target Devices: PAL16R8, PAL16RP8, GAL16V8      */
/*****/
Pin 1 = clock;
Pin 2 = reset;

/* Outputs: define outputs and output active levels */
Pin 17 = q0;
Pin 16 = q1;

/* Logic: two bit counter using expanded exclusive ors */
/*      with d-type flip-flop                               */
q0.d = !reset & (!q0 & !q1
              # !q0 & q1);
q1.d = !reset & (!q0 & q1
              # q0 & !q1);
/* ANDed !reset defines a synchronous register reset */

```

The first part of the file provides archival information and a description of the intended function of the design, including compatible PLDs.

Pin declarations are made to correspond to each input and output in the design diagram.

In the “Logic” section of the file, equations are written to implement the counter. The equation for q0 is written to define when q0 asserts; that is, it defines the situation immediately before the rising clock edge.

The !reset term is used in the equations for both q0 and q1 to initialize the circuit, providing a synchronous reset. At power-on, the registers can be either high or low, as indicated by the DON’T CARE slashes in the timing diagram in the source file; the reset signal is initially asserted. By ANDing !reset into the equation for each variable, the conditions are not met at power-on, so the registers do not set. Because the reset signal returns LO (false) after the power-on process is complete, !reset is then true and does not affect the value of the registers at any other point in the circuit.

The .d extension in the equations specifies a D-type flip-flop. However, when an output is used as feedback, the .d extension is dropped. For example, if q0 is fed back to q1, an equation could be written as:

```
q1.d = q0 & !reset ;
```

not as:

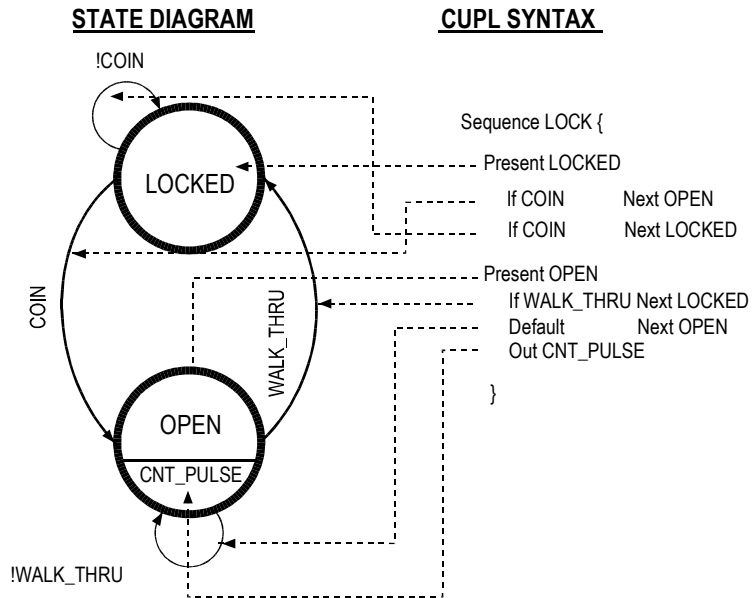
```
q1.d = q0.d & !reset ;
```

or:

```
q1.d = q0.dq & !reset ;
```

Example 3 - Simple State Machine Design

The Subway Turnstile controller is the simplest state machine design. This controller waits for a signal that a coin has been deposited. It then changes its state from locked to open. In the open phase, it waits for someone to walk through the turnstile, then it changes from open to locked. This two-state design cycles between open and locked using a coin detector and a walk-through detector as inputs. The diagram below shows the states, and the pulses that change the device from one state to the next.



Subway Turnstile Example

The corresponding CUPL state machine code is included on the right, so that the relationship between the code and the design diagram is easily understood.

```

/*****
/*   Target Devices: P16R4
*****/

/* Inputs:   define inputs to Turnstile controller   */
Pin 1 = CLK;
Pin 2 = WALK_THRU;
Pin 3 = COIN;

/* Outputs:   define outputs as active HI levels
..... */
Pin 14 = CNT_PULSE;
Pin 15 = LOCK;

/* Logic:   Subway Turnstile example expressed in CUPL */
$DEFINE LOCKED 'b'0
$DEFINE OPEN  'b'1
Sequence LOCK{

    Present LOCKED
        if COIN      Next OPEN;
        if !COIN     Next LOCKED;

    Present OPEN

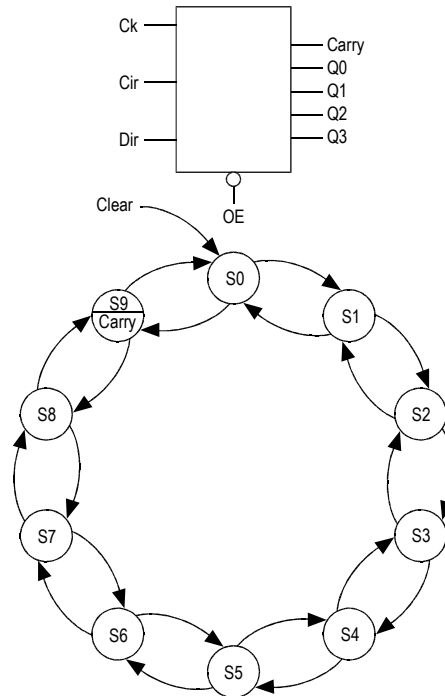
        if WALK_THRU Next LOCKED;
        Default      Next OPEN;
        Out CNT_PULSE;
}

```

TURNSTIL.PLD

Example 4 - Decade Up/Down Counter

This example describes a four-bit up/down decade counter with a synchronous clear capacity. The counter also provides an asynchronous ripple carry output for cascading multiple devices. The source file to implement the counter uses CUPL state machine syntax.



Up/Down Counter Diagram

The input signal dir determines the direction of the count. When dir is high, the count goes down one on each clock; when dir is low, the count goes up one on each clock. The clr signal performs a synchronous reset.

The CUPL source code that implements the design is shown below (refer to COUNT10.PLD in the examples).

```

COUNT10.PLD
Name          Count10;
Partno        CA0018;
Revision      02;
Date          07/16/87;
Designer      Kahl;
Company       ATI;
Location      None;
Assembly      None;
Device        p16rp4;

/*****
/*
/*          Decade Counter
/* This is a 4-bit up/down decade counter with
/* synchronous clear capability. An asynchronous
/* ripple carry output is provided for cascading
/* multiple devices. CUPL state machine syntax
/* is used
*****/
/* Allowable Target Device Types: PAL16RP4, GAL16V8, EP300 */
/*****
** Inputs **/
Pin 1 = clk;          /* counter clock          */
Pin 2 = clr;          /* counter clear input   */
Pin 3 = dir;          /* counter direction input */
Pin 11 = !oe;         /* Register output enable */

/* Outputs

Pin [14..17] = [Q3..0]; /* counter outputs
Pin 18 = carry;         /* ripple carry out

/* Declarations and Intermediate Variable Definitions
field count = [Q3..0]; /* declare counter bit field */
$define S0 'b'0000
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110

```

```

$define S7 `b'0111
$define S8 `b'1000
$define S9 `b'1001
field node = [clr,dir];      /* declare filed node control */
up = mode:0;                 /* define count up mode      */
down = mode:1;               /* define count down mode    */
clear = mode:[2..3];         /* define count clear mode   */

/* Logic Equations */
sequence count {            /* free running counter      */

present S0                  if up          next S1;
                             if down        next S9;
                             if clear       next S0;
present S1                  if up          next S2;
                             if down        next S0;
                             if clear       next S0;
present S2                  if up          next S3;
                             if down        next S1;
                             if clear       next S0;
present S3                  if up          next S4;
                             if down        next S2;
                             if clear       next S0;
present S4                  if up          next S5;
                             if down        next S3;
                             if clear       next S0;
present S5                  if up          next S6;
                             if down        next S4;
                             if clear       next S0;
present S6                  if up          next S7;
                             if down        next S5;
                             if clear       next S0;
present S7                  if up          next S8;
                             if down        next S6;
                             if clear       next S0;
present S8                  if up          next S9;
                             if down        next S7;
                             if clear       next S0;
present S9                  if up          next S0;
                             if down        next S8;
                             if clear       next S0;

out                          carry;      /* assert carry output */

```

The first part of the file provides archival information and a description of the intended function of the design, including compatible PLDs.

Pin declarations are made corresponding to the inputs and outputs in the design diagram.

The “Declarations and Intermediate Variable Definitions” section contains declarations that simplify the notation.

The name “count” is assigned to the output variables Q3, Q2, Q1, and Q0.

The \$DEFINE command is used to assign names to ten binary states representing the state machine output. The state name can then be used in the logic equations to represent the corresponding binary number.

The FIELD keyword is used to combine the clr and dir inputs into a set called mode. Mode is defined by the following equations:

```
up = mode:0;
down = mode:1;
clear = mode [2..3];
```

Mode represents the inputs clr and dir, so the three equations above are equivalent to the following equations:

```
up = !clr & !dir ;
down = !clr & dir ;
clear = (clr & !dir) # (clr & dir) ;
```

The three modes are defined as follows:

up - Both the dir and clr inputs are not asserted.

down - The dir input is asserted and clr is not asserted.

clear - The clr input is asserted and dir is either asserted or not asserted.

The “Logic Equations” section contains the state machine syntax that specifies the states in the counter. In the first line, the SEQUENCE keyword identifies count (that is, Q3, Q2, Q1, and Q0) as the outputs to which the state values apply.

Conditional statements have been written to specify the transition from each possible present state to a next state, for each of the three modes. For example, when the present state is S4, if the mode is up, the counter goes to S5; if the mode is down the counter goes to S3; or if the mode is clear, the counter goes to S0. As example 4 shows, one advantage of the state machine syntax is that it clearly documents the operation of the design.

In Example 4, state 0 (binary value 0000) is defined, because it is the result of the clr signal. It is recommended that all designs have a valid 0000 defined to avoid being stuck at state 0. For example, in this design, if a state that hasn’t been defined occurs at power-on, such as hexadecimal A-F, none of the conditions described in the equations is met, so the state goes to state 0 (hex value 0000). If 0000 has not been defined as a valid state, the counter stays at state 0.

Below you can see how this example could have been written as a virtual design. It is the same file, but it has been modified where necessary to show the difference between a virtual design and a device specific design.

```

COUNT10.PLD
Name          Count10;
Partno        CA0018;
Revision      02;
Date          07/16/87;
Designer      Kahl;
Company       ATI;
Location      None;
Assembly      None;
Device        VIRTUAL;

/*****
/*
/*          Decade Counter          */
/* This is a 4-bit up/down decade counter with */
/* synchronous clear capability. An asynchronous */
/* ripple carry output is provided for cascading */
/* multiple devices. CUPL state machine syntax */
/* is used */
*****/
/* Allowable Target Device Types: PAL16RP4, GAL16V8, EP300 */
*****/
/** Inputs **/
Pin = clk;          /* counter clock */
Pin = clr;          /* counter clear input */
Pin = dir;          /* counter direction input */
Pin = !oe;          /* Register output enable */

/** Outputs **/

Pin = [Q3..0];      /* counter outputs */
Pin = carry;        /* ripple carry out */

/** Declarations and Intermediate Variable Definitions **/
field count = [Q3..0]; /* declare counter bit field */
$define S0 'b'0000
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110

```



```

$define S7 'b'0111
$define S8 'b'1000
$define S9 'b'1001
field node = [clr,dir];      /* declare filed node control */
up = mode:0;                 /* define count up mode      */
down = mode:1;               /* define count down mode    */
clear = mode:2..3];         /* define count clear mode   */
/* Logic Equations */
sequence count {            /* free running counter      */

present S0                  if up          next S1;
                           if down         next S9;
                           if clear        next S0;
present S1                  if up          next S2;
                           if down         next S0;
                           if clear        next S0;
present S2                  if up          next S3;
                           if down         next S1;
                           if clear        next S0;
present S3                  if up          next S4;
                           if down         next S2;
                           if clear        next S0;
present S4                  if up          next S5;
                           if down         next S3;
                           if clear        next S0;
present S5                  if up          next S6;
                           if down         next S4;
                           if clear        next S0;
present S6                  if up          next S7;
                           if down         next S5;
                           if clear        next S0;
present S7                  if up          next S8;
                           if down         next S6;
                           if clear        next S0;
present S8                  if up          next S9;
                           if down         next S7;
                           if clear        next S0;
present S9                  if up          next S0;
                           if down         next S8;
                           if clear        next S0;
out                          carry;        /* assert carry output */

```

It is possible to use some of the features of the CUPL preprocessor to considerably shorten this PLD file. The sample file below shows how this same file could be written with a \$REPEAT structure, which reduces the file size considerably.

```

Name          Count10;
Partno        CA0018;
Revision      02;
Date          07/16/87;
Designer      Kahl;
Company       ATI;
Location      None;
Assembly      None;
Device        VIRTUAL;

/*****
/*
/*          Decade Counter          */
/* This is a 4-bit up/down decade counter with */
/* synchronous clear capability. An asynchronous */
/* ripple carry output is provided for cascading */
/* multiple devices. CUPL state machine syntax */
/* is used */
/*
/* Allowable Target Device Types: PAL16RP4, GAL16V8, EP300 */
*****/

/** Inputs **/
Pin = clk;          /* counter clock          */
Pin = clr;          /* counter clear input    */
Pin = dir;          /* counter direction input */
Pin = !oe;          /* Register output enable */

/** Outputs **/
Pin = [Q3..0];      /* counter outputs        */
Pin = carry;        /* ripple carry out        */

/* Declarations and Intermediate Variable Definitions */
field count = [Q3..0]; /* declare counter bit field */
field node = [clr,dir]; /* declare filed node control */
up = mode:0;          /* define count up mode     */
down = mode:1;        /* define count down mode   */
clear = mode:2..3];   /* define count clear mode  */

/* state machine description */
sequence count {

```

```

present 0
    if up & !clear    next 1;
    if down & !clear next 9;
    if clear          next 0;

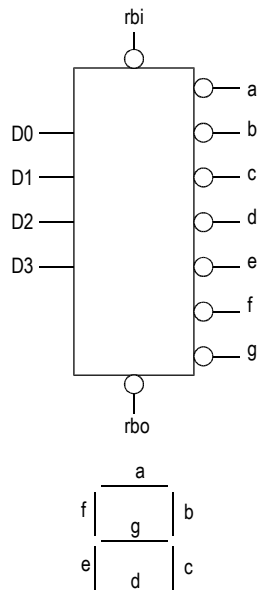
$REPEAT i=[1..9]
present i
    if up & !clear    next {(i+1)%10};
    if down & !clear next {(i-1)%10}
    if clear          next 0;
$REPEND

```

In this variation, we removed the \$DEFINE statements because we use the raw numbers instead. The most significant change is that we used a \$REPEAT loop to define most of the states instead of defining each state individually. It is possible to do this because all the states are identical except for the next state that each goes to. Notice that we define state 0 by itself and then all the other states are defined in one \$REPEAT loop. The \$REPEAT loop expands upon compilation to give a definition for each state. Notice that the statement indicating the next state is given as a calculation from the repeat variable 'i'. In the loop, 'i' represents the number of the current state. The next state is therefore 'i+1'. This will work for all states except the last state. In the last state, the state machine must go back to state 0. To accomplish this, the formula to calculate the next state is given as '(i+1)%10'. This means 'i+1' modulo 10. The number 10 represents the number of states. Therefore, when in state 9 the next state is calculated as $9+1 = 10$ then modulo 10 which gives 0. A similar condition occurs in calculating the previous state except that we subtract 1 instead of adding it. You might have noticed that we defined state 0 separately. This was done because the \$REPEAT variables can only handle positive numbers. If we had defined state 0 in the \$REPEAT loop this would result in evaluating to next state -1 and the compiler would produce an unexpected result.

Example 5 - Seven-Segment Display Decoder

This example shows a hexadecimal-to-seven-segment decoder for driving common-anode LEDs. The design incorporates both a ripple-blanking input to inhibit the display of leading zeroes, and a ripple-blanking output for easy cascading of digits.



Seven-Segment Display Decoder

The segments in the display, labeled a-g, correspond to the outputs in the diagram.

Below is the source code (HEXDISP.PLD in the examples):

```

      HEXDISP.PLD
Name      Hexdisp;
Partno    CA0007;
Revision  02;
Date      07/16/87;
Designer  T. Kahl;
Company   ATI;
Location  None;
Assembly  None;
/*****
/*
/*                                     a
/*

```

```

/* This is a hexadecimal-to-seven-segment      ----- */
/* decoder capable of driving common-anode      |      | */
/* LEDs. It incorporates both a ripple-        f|      |b */
/* blanking input (to inhibit displaying        |  g  | */
/* leading zeroes) and a ripple blanking output ----- */
/* to allow for easy cascading of digits      |      | */
/*                                             e|      |c */
/*                                             |      | */
/*                                             ----- */
/*                                             d      */
/*                                             */
/*****
/* Allowable Target Device Types: 32 x 8 PROM (82S123 or */
/* equivalent */
/*****
/** Input group (Note this is only a comment) */

pin [10..13] = [D0..3];      /* data input lines to display */
pin 14 = !rbi;               /* ripple blanking input */

/** Output Group ( Note this is only a comment) */

pin [7..1] = ![a,b,c,d,e,f,g]; /* Segment output lines */
pin 9 = !rbo;                 /* Ripple Blanking output */
/** Declarations and Intermediate Variable Definitions */
field data = [D3..0];         /* hexadecimal input field */
field segment=[abcdefg];      /* Display segment field */
$define ON 'b'1               /* segment lit when logically "ON" */
$define OFF 'b'0              /* segment dark when logically "OFF" */

```

Display Decoder Source File (HEXDISP.PLD) Sheet 1 of 2

The first part of the file provides archival information and a description of the intended function of the design, including compatible PLDs.

Pin declarations are made corresponding to the inputs and outputs in the design diagram.

In the “Declarations and Intermediate Variables” section, field assignments are made to group the input pins into a set named ‘data’ and the output pins into a set named ‘segment’. ON and OFF are defined respectively as binary 1 and binary 0.

```

/** Logic Equations */
/*      a      b      c      d      e      f      g */
segment =
/* 0 */      [ ON,   ON,   ON,   ON,   ON,   ON,   OFF] & data:0 & !rbi
/* 1 */      # [OFF,  ON,   ON,   OFF, OFF, OFF, OFF] & data:1
/* 2 */      # [ ON,  ON,   OFF, ON,   ON, OFF, ON] & data:2
/* 3 */      # [ ON,  ON,   ON,  ON,   OFF, OFF, ON] & data:3
/* 4 */      # [OFF,  ON,   ON,  OFF, OFF, ON,  ON] & data:4
/* 5 */      # [ ON,  OFF, ON,   ON,  OFF, ON,  ON] & data:5
/* 6 */      # [ ON,  OFF, ON,   ON,  ON,  ON,  ON] & data:6
/* 7 */      # [ ON,  ON,   ON,  OFF, OFF, OFF, ON] & data:7
/* 8 */      # [ ON,  ON,   ON,  ON,  ON,  ON,  OFF] & data:8
/* 9 */      # [ ON,  ON,   ON,  ON,  OFF, ON,  ON] & data:9
/* A */      # [ ON,  ON,   ON,  OFF, ON,  ON,  ON] & data:A
/* B */      # [OFF,  OFF, ON,   ON,  ON,  ON,  ON] & data:B
/* C */      # [ ON,  OFF, OFF, ON,  ON,  ON,  OFF] & data:C
/* D */      # [OFF,  ON,   ON,  ON,  ON,  OFF, ON] & data:D
/* E */      # [ ON,  OFF, OFF, ON,  ON,  ON,  ON] & data:E
/* F */      # [ ON,  OFF, OFF, OFF, ON,  ON,  ON] & data:F;

rbo = rbi & data:0;

```

Display Decoder Source File (HEXDISP.PLD) Sheet 2 of 2

The logic equations are set up as a function table to describe the segments that are lit up by each input pattern. Comments create a header for the function table, listing the output segments across the top and the input numbers vertically down the side.

Each line of the table describes a decoded hex value and the segments of the display that the hex value turns on or off. For example, the line for an input value of 4 is written as follows:

[OFF, ON, ON, OFF, OFF, ON, ON] & data:4

The function table format expresses the intent of the design more clearly than equations; that is, the example above shows that an input value of 4 turns segment a off, segment b on, segment c on, and so on.

Example 6 - 4-Bit Counter with Load and Reset

This example will present a counter that can be loaded and reset. The design is done using the VIRTUAL device but it could easily be implemented in almost any PLD that has four or more registers.

```
Name                Counter;
Partno              FL1201;
Revision            01;
Date                08/26/91;
Designer            RGT;
Company             LDI;
Location            None;
Assembly            None;
Device              VIRTUAL;
/*****
/* 4-bit counter
*****/

/** inputs */
PIN = clk; /* clock signal for registers */
PIN = load; /* load signal */
PIN = !ClrFlag;
PIN = [LoadPin0..3]; /* pins from which to load data */

/** outputs */
PIN = [CountPin0..3];

/* intermediate variables and fields */
field STATE_BITS = [Count0..3];
field LOAD_BUS = [LoadPin0..3];

/** state machine definition */
Sequenced STATE_BITS {
/* build a repeated loop for the states */
$REPEAT i = [0..15]
    Present `h'{i}
        /* go to state 0 if clear signal is true */
        /* if the load signal is false go to the */
        /* next state. Note that the next state */
        /* is (current state + 1) modulo 16. */
        /* This causes the counter to wrap back */
        /* to 0 when in the last state */
}
```

```

If !load Next 'h' {(i+1)%16};
If !load & ClrFlag Next 'b'0;
$REPEND
/* Add the load capability by using the */
/* APPEND statement. This has the effect of */
/* adding more inputs to the OR gate for */
/* this output. This equation states that if */
/* the load signal is true then the counter */
/* registers are loaded with data from the */
/* load pins. This is why 'load' was used in */
/* the equations for the 'IF' statements in */
/* the state definitions. */
APPEND STATE_BITS.d = load & LOAD_BUS;

```

Since this is a virtual design, pin numbering can be ignored. The signals to be used are declared without any pin numbers.

When the load signal is asserted, the values at the load pins are fed into the state bit registers. When the 'Clear' signal is asserted, the state machine is forced to state 0.

The \$REPEAT loop is defined as [0 ..15]. Inside the loop, the present state is defined as 'h' {i}. This causes the numbers to be evaluated as hex numbers. The expansion that results from this is a state machine with 16 states ranging from 0 to F in hexadecimal. If the 'h' was left off, the states would have been expanded as 0 to 15 in decimal. The compiler would have interpreted these as hex anyway and states A-F would not have been defined. Additionally, State 10 in hex cannot exist with this number of statebits since that would require 5 statebits but all that we have is 4.

Notice that all the IF statements are anded with "!load". This was done because we want to give loading the highest possible priority. Also this removes any possible conflict that could occur if load and some other signal were asserted at the same time.

The next state is calculated as (present_state + 1) modulo 16. This causes the counter to wrap back to the zero state when it is in the last state. We use modulo 16 because this is the number of states in the state machine.

The last part of this design involves adding the load capability to our design. For this we use the APPEND statement. The append statement causes the expression given to be Ored to the variable specified. This entire state machine eventually becomes a set of equations for each of the state bits. Our intent is to add another condition where the registers are loaded with the value from a set of input pins. Remember that we used '!load' in all the IF statements. That was to make sure that the equations generated by those IF statements, would not conflict with the APPEND statement.

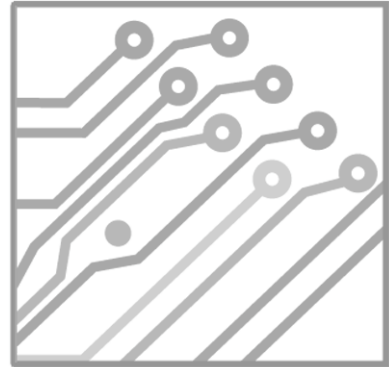

```
CountPin0.d = !load & ???.....  
             # !load & ???.....;
```

Equations Before APPEND

```
CountPin0.d = !load & ???.....  
             # !load & ???.....  
             # load & LoadPin0;
```

Equations After APPEND

As an exercise, try to add up/down capability to this example. Also, try implementing it into an actual device.

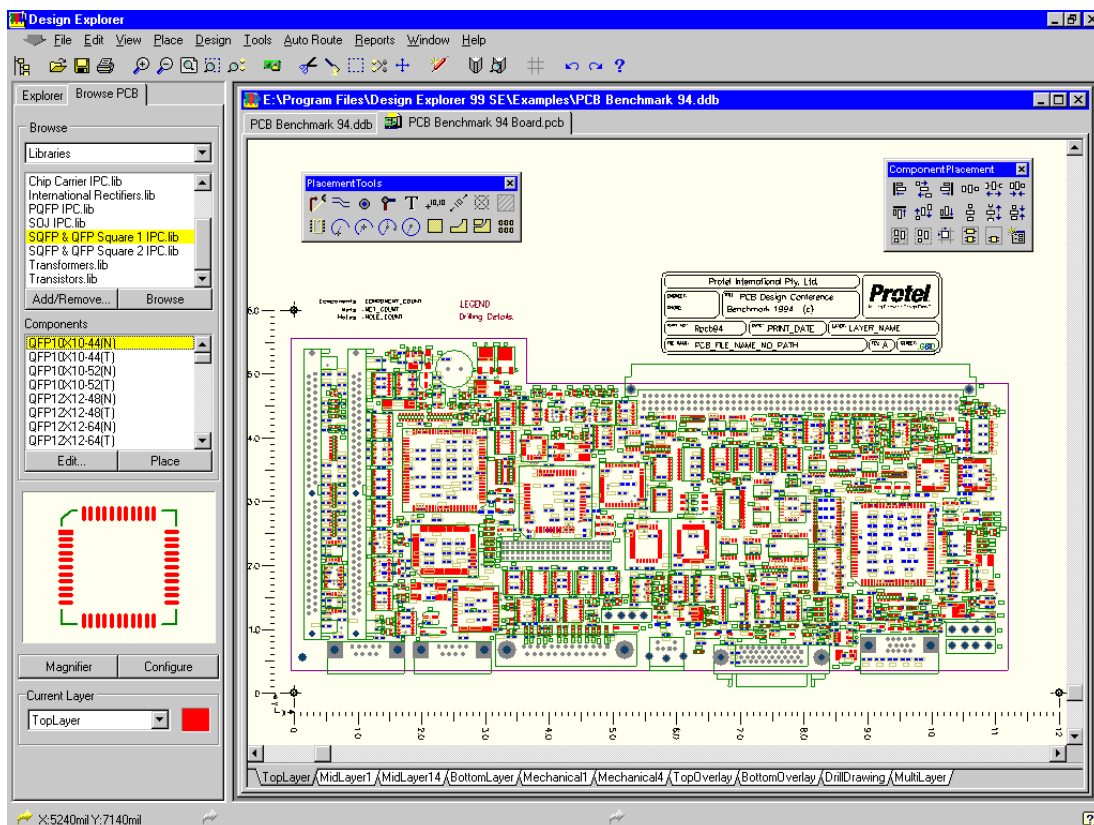


Section 6 ———

PCB Design

<i>PCB Design – Feature Highlights</i>	439
<i>Setting up the PCB Workspace</i>	446
<i>Creating, Opening and Saving PCB Documents</i>	460
<i>Working in the PCB Design Window</i>	461
<i>PCB Editing Shortcuts</i>	487
<i>PCB Design Objects</i>	492
<i>PCB Component Footprints and Libraries</i>	509
<i>Defining the Board</i>	521
<i>Specifying the PCB Design Requirements</i>	526
<i>Component Placement Tools and Techniques</i>	563
<i>Understanding Connectivity and Topology</i>	576
<i>Manually Routing the PCB</i>	583
<i>Autorouting the PCB</i>	599
<i>Including Testpoints and Teardrops on the PCB</i>	605
<i>Verifying the PCB Design</i>	609
<i>3 Dimensional PCB Visualization</i>	618
<i>Printing to a Windows Printing Device</i>	620
<i>Generating the Manufacturing Files</i>	626
<i>Passing Design Changes Back to the Schematic</i>	642
<i>Interfacing to Third-Party Tools</i>	644

PCB Design – Feature Highlights



This section of the Protel Designer's Handbook explains how to use Protel 99 SE's PCB Editor and Autorouter to carry out the printed circuit board layout and routing phase of the design process. As you read this section you will find all the information you need to get up and running with the PCB Editor, and learn how to use the basic features required to layout and route a printed circuit board, perform design rule checks, generate PCB fabrication files and print out design documentation.

This section of the Handbook also includes information on how to perform a signal integrity analysis, directly from the PCB layout. You can set up design rules to test for out of spec performance, including net impedance, overshoot, undershoot, slope and flight time. You can also analyze the exact behavior of the routing by performing reflection and crosstalk analyses, producing accurate waveforms of the results. Refer to the chapter, *Verifying the PCB Design*, for more information.

The PCB Layout Editor

The PCB Layout Editor is the primary PCB document editor in Protel 99 SE. This editor allows you to create, edit and verify the PCB design. From the PCB Layout Editor you can also generate the output files required to manufacture the printed circuit board. Throughout this Handbook the PCB Layout Editor is simply referred to as the PCB Editor.

The PCB Library Editor

The PCB Library Editor is the second PCB document editor in Protel 99 SE. It is used to create, edit and manage libraries of component footprints. The PCB Library Editor shares many common features with the PCB Editor, plus specialized tools and features for library management tasks.

Shape-Based Autorouting

Protel 99 SE includes an easy-to-use, powerful, high quality, shape-based autorouting server, which is tightly integrated with the PCB Editor. Although it plugs in as a separate server in the Design Explorer, the Autorouter is set up and run in the PCB Editor, and routes directly in the PCB window.

Protel 99 SE's Autorouter uses a combination of new routing technologies, as well as time-proven developments. The new technology consists of a suite of Neural utilities including a Neural Net, Neural Costs and Neural Shapes – utilities that have grown from the development of Artificial Intelligence. The well-proven technologies include a variety of time-tested routing algorithms, that have been greatly enhanced to support the objectives of Protel 99 SE's Autorouter.

The Autorouter has three objectives: 100% completion, routing speed and quality of routing - like that of a professional board designer. Once you start routing with Protel 99 SE's Autorouter you will wonder how you ever survived without it.

Design Capabilities

The PCB Editor is a complete PCB layout environment with many attractive features for productive design work. When you use the PCB Editor in combination with Protel 99 SE's Schematic Editor, Circuit Simulator, and PLD Compiler, the PCB Editor becomes the backbone of a fully-automated, integrated, end-to-end PCB layout system.

PCB layout differs from other drawing-oriented tasks in its requirement for extreme precision. As a result, Protel's PCB Editor is more of a "placing" environment than a freehand "drawing" environment.

Another key difference is connectivity – the system’s ability to recognize connections between track segments, tracks and component pads, and so on. For example, you can delete a track segment, and the system will automatically add a connection line, ensuring that the connective integrity of the design is always maintained.

PCB layouts are generated and displayed as a set of layers, which correspond to the individual “phototools” used to fabricate the board, such as the top and bottom signal layers or the silkscreen overlay layer. Some operations, such as manual track placement, are layer-dependent – you must first select the layer, then place the track.

Whether your design is a simple single-sided PCB, or a multi-layer board with multiple internal planes, you will be able to layout every item exactly as it will be fabricated in Protel 99 SE’s PCB Editor.

Schematic-to-PCB Design Synchronization

Protel 99 SE includes a powerful design synchronization feature, which can automatically match the schematic design data with the PCB design data. Previously, schematic design changes were carried forward using a technique called *forward annotation*, and PCB re-annotation information was taken back to the schematic using *backward annotation*. Both of these functions are now carried out by the synchronizer.

When you run the synchronizer it separately examines the component and connectivity information in both the schematic sheets, and the PCB, and then updates one to match the other. When you run it you select either **Update PCB from Schematic**, or **Update Schematic from PCB** from the menus, nominating which is the source, and which is the target.

32-bit PCB Design Database

The PCB Editor uses a 32-bit design database and can generate through-hole and SMD designs of up to sixteen signal layers, plus four mid-layer power planes. Four mechanical drawing layers allow you to generate fabrication and assembly drawings for your design. Boards can be as big as 100 inches (254 cm) by 100 inches. Placement accuracy on the 0.001 mils grid system is ± 0.0005 mils.

The switchable metric/imperial grid system allows you to work accurately in both measurement systems and can be toggled on-the-fly as you design.

Comprehensive Design Rules

Today’s electronic designs impose more than simple mechanical and current/voltage requirements on the PCB layout. They can also require that you apply specific conditions to individual nets, components, or regions of the board, as well as considering such issues as crosstalk, reflections and net lengths.

To specify these requirements the PCB Editor incorporates a large set of design rules. These include clearances, object geometry, parallelism, impedance control, routing priority and routing topology. Each rule can be applied to the board, to objects, nets,

net classes, from-tos, from-to classes, components, component classes, layers, or user definable regions.

On-Line and Batch Design Rule Checking

The on-line DRC flags design rule violations as you route. The batch DRC allows comprehensive verification of the board layout to user-specified physical and logical design rules.

Automatic Component Placement

The Protel 99 SE design system includes two high-performance auto component placement features.

The first autoplacer uses a component clustering algorithm, that attempts to place components first by their connectivity (creating clusters of components), then by the component geometry. This placer is suited to boards with a lower component count (less than 100).

The second auto component placement feature uses an AI-based methodology called simulated annealing. It analyzes the entire design as it places, considering the connection length, the connection density on the board and the alignment of the components, all in accordance with the design rules. As this placer uses a statistical algorithm it is more suited to designs with a higher component count (100 or more).

Unbreakable Connectivity

A key feature of the PCB Editor is the way logical and physical (or electrical) connections between the elements in your design are recognized and managed. At all times the PCB Editor monitors the state of the connectivity, adding and removing connection lines as you place and delete tracks.

Sophisticated Gridless Manual Routing

With the ever-increasing variety of component packaging technologies it is difficult for today's designer to manually route in a grid-based design environment. To keep pace with the changing demands of manual routing, tracks can be routed gridless in the PCB Editor.

Combine the electrical grid (which snaps objects together) with the avoid obstacle mode and the six track placement modes (with look-ahead), and you can predictively route tracks to any object without creating a violation.

Flexible Selection

Groups of items can be selected by layer, by physical connectivity or by designating an area of the board. Individual items can be added to or removed from the selection. The PCB Editor also includes a Query Wizard, allowing you to create complex selections

of different primitives, using standard query operators such as not equal to, less than, and so on.

Selections can be manipulated using standard Windows Edit menu items like Cut, Copy, Paste or Clear; moved; flipped on either axis or rotated in .001 degree increments.

Powerful Global Editing Options

Attributes can be edited by double-clicking directly on the item to open a dialog. In the PCB Editor changes made to one object can be globally applied across an entire design using specific conditions to define the targets. For example, when editing tracks you can change the track width, track layer or both the width and layer. These changes can be globally applied to all tracks of the same width and/or layer; tracks which are not the same width and/or layer; all selected tracks; all non-selected tracks. Similar global options are provided for other design objects.

Linear and Circular Array Placement Options

Array placement allows selections to be placed in circular arrays, as well as straight lines. Circular repeats are defined by radius and angular increment. Each repeated item can be rotated around its own axis.

Undo and Redo

Multi-level Undo and Redo processes work for all physical changes to the board layout. The designer can make multiple changes, backtrack using Undo, then reinstate each “Undo” change with the Redo process.

Complete Component and Library Management

Multiple libraries can be opened simultaneously. Open libraries in the PCB Library Editor while working on the board design in the PCB Editor. Over 300 component patterns, including through-hole and SMD footprints are included in the standard PCB design system library. Simultaneous multi-user library access is supported for network installations.

The PCB Library Editor also includes a powerful component building Wizard. This Wizard will ask a few questions and then build the component footprint for you, from a simple two pin resistor, through to a Pin Grid Array with hundreds of pins.

Intelligent Polygon Planes

Solid or lattice polygon planes can be placed on any layer with optional automatic connection to a specified net. The copper “pours” automatically, wrapping around all placed objects, obeying any relevant design rules. Polygon shapes can be defined using line or arc perimeters and vertices can be moved, added or deleted after the polygon is

generated. Polygons can be re-poured around new obstacles and you can redefine the polygon parameters each time the polygon is re-poured.

Split Internal Power Planes

Internal power planes can be “split” to be shared between multiple power rails. Split power planes are fully supported by the Design Rule Checker.

Thermal Relief Control

Where pins connect to polygons or power plane layers they can have thermal relief or direct connections. Both the conductor path width and air-gap can be user-defined, with a choice of 2 or 4 entry points.

Pad Stacks and Pad Removal

Multi-layer pads can be assigned independent size and shape attributes for the Top (component side) layer, Mid layers (1–14) and Bottom (solder) side layer. Unconnected multi-layer pads on Mid layers can be automatically removed when printing or plotting artwork.

Blind and Buried Vias

Vias can either pass through the whole board or connect any two layers. Blind and buried vias can be automatically placed during manual routing. Vias use layer colors to indicate which layers are connecting. Blind and buried vias can also be used between any two layers, supporting build-up fabrication technology.

Fractional Arcs

The PCB Editor has an arc placement resolution of .001 degree. Arcs can be placed on any layer with full connectivity checking on signal layer arcs.

Component Rotation

Full rotation of components and their pads, down to .001 degrees. The same angular resolution is available for the rotation of any selection.

Multiple Fonts

Three display fonts (default, Serif and Sans Serif) support vector plotting and photoplotting.

Automatic Photoplot Generation

Fully-automatic Gerber® plot file generation. Fully-automatic aperture file generation. On-line aperture editing. Embedded aperture support for Gerber 274X format. Composite photoplots of multiple layers. Automatically panelized plot files to specified film size and border requirements.

The PCB Editor can import and display Gerber files, as well as batch load a set of Gerber files, with each plot assigned to the appropriate PCB layer.

Windows Support for Printing and Pen Plotting

Dot matrix and laser printing, pen plotting and PostScript® output are all controlled from common Print options. Any device supported by Windows is available for output. Plots or prints can either be panelized or generated as a composite of multiple layers, with auto-centering on the sheet.

Automatic NC Drill File Generation

NC drill output is generated automatically without the need for user-defined tool files. A report file is generated that lists each tool required, in both metric and imperial units, and the travel distance for each tool. A fast sorting algorithm processes the NC drill output file for efficient drilling.

Editable Drill Drawings

Drill drawings are fully user-editable with optional markers for each hole location, including: coded symbol, alphabetical codes (A, B, C etc) or the assigned hole size.

Windows Display Options

Protel's PCB design system allows full use of all 24-bit color graphics cards and monitors supported under Windows. Zoom levels support the full 32-bit system resolution (accurate to ± 0.0005 mils).

Import and Export DXF Format Files

Import DXF files (AutoCAD®) and export PCB designs to DXF format files. Multi-layer DXF files are supported.

Reports

The PCB Editor can generate the following reports; Bill of Materials (BOM); Back Annotation files; NC Drill and Pick and Place reports for board fabrication and assembly; a Netlist Status report; Engineering Change Order (ECO) reports and other design reports.

Setting up the PCB Workspace

Coordinate System

Coordinates displayed on the left end of the Status Bar indicate the position of the cursor relative to the *current* workspace origin. The coordinates indicate the cursor distance from the *current origin* in mils (thousandths of an inch) or millimeters, depending on the units selected. The PCB Editor allows you to set a new current origin anywhere in the workspace.

The *absolute origin* (the default position of the current origin) is the extreme lower left corner of the workspace.

Setting the Current Origin

Select **Edit » Origin » Set** to set the current origin at the current cursor position. Once it has been set the Status Bar will display X:0 mils Y:0 mils (or X :0 mm Y:0. mm if a metric snap grid is used) at the current cursor position.

To set the current origin back to the absolute origin (extreme lower left corner of the workspace) select **Edit » Origin » Reset**.

Workspace Size and Accuracy

Printed circuit boards are manufactured to very close tolerances. The PCB Editor provides an absolute design resolution of ± 0.001 mils (.000001 inch or .00025 mm) – which will provide sufficient precision for any PCB design task. The workspace is 100 inches by 100 inches.

Switching from Imperial to Metric

The PCB Editor supports both imperial (mils) and metric (mm) measurement units and dimensions. The unit of measure changes when you select the **View » Toggle Units** menu item, or press the Q shortcut key. When a metric snap grid is selected, workspace coordinates and other dimensional information are displayed in millimeters (indicated as “mm” on the Status Bar). This allows accurate dimensioning of boards in millimeters or creation of new library components with metric pin spacing. Measurement units can be switched at any time by pressing the Q shortcut key.

Grids

The PCB Editor includes four user-definable grid systems. The first two are the *snap grid* and the *component grid*, which control the placement of design objects in the workspace. The third is the *electrical grid*, which defines a “range of attraction” within which electrical objects attract to each other. The forth are the *visible grids*, which provide a visual reference as you move around the workspace.

Snap Grid and Component Grid

The snap grid defines an array of points in the workspace which restrict cursor movement and the placement of primitives, while the component grid restricts the placement of components. When using the mouse to control the cursor, you will notice that the cursor moves freely between snap grid points. If an edit function is being performed, such as placing a component or selecting an object, a cross hair appears. This cross hair will “snap” to the current snap grid for a primitive, or to the current component grid for a component. When the cursor keys are used, the cursor always “snaps” to the grid.

These grids can be changed at any time in the Options Tab of the Document Options dialog (**Design » Options**) or via the Set Snap Grid button on the main toolbar (shortcut: CTRL+G). Setting the snap grid to 100 mils will mean the cursor can only be on points, 0.0 inch 0.1 inch 0.2 inch, etc. The snap grid can have a value between 0.001–1000 mils (or 0.0025–25.0mm).

◆ Hold SHIFT while pressing a cursor key to make the cursor jump 10 times the current setting of the snap or component grid.

The snap grid and component grid setting define where objects can be placed in the workspace. It is vital that these grids are set appropriately for good board design. They are typically set to either a multiple of the component pin pitch, or a fraction of it. For example, while placing components with a pin pitch of 100 mil, a component grid of 50 or 100 mil could be used. To route one track between the pins of these components a snap grid of 25 mil could be used. Working with appropriate grids will assist in orderly component placement and provide the maximum amount of routing channels.

Electrical Grid

To ease the placement of electrical objects, such as tracks and vias, the PCB Editor includes an *electrical grid*. This grid defines a range within which a moving electrical object (such as a track, pad or via) will attract, or snap to, another electrical object.

As you move an electrical object around the workspace and it falls within the electrical grid range of another electrical object, the object you are moving will snap to a hot spot on the fixed object. The electrical grid is configured in the Options Tab of the Document Options dialog (**Design » Options**).

◆ The electrical grid overrides the snap grid. This allows you to easily connect to an off grid object. Toggle the electrical grid on and off as you work with the SHIFT+E shortcut.

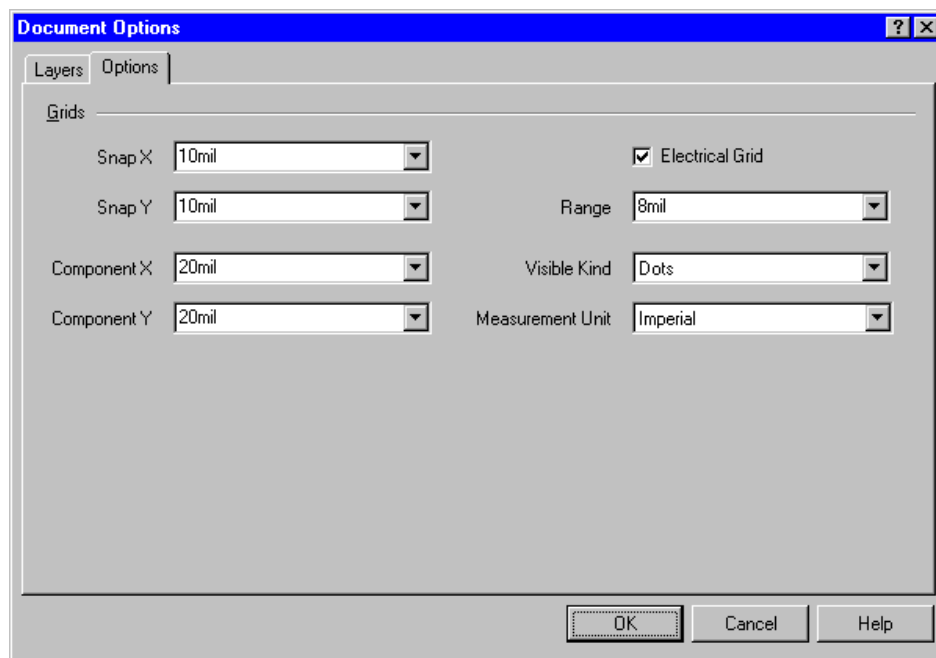
Visible Grids

Two visible grids are provided as a visual reference for placing and moving items. You can set the sizes of these grids independently. For example, you could select one visible grid to be fine and the other coarse, or even separate metric and imperial visible grids.

The visible grid displays a system of coordinate lines (or dots) in the workspace background. The display of the visible grids is constrained by the current zoom level, if you cannot see a visible grid you are either zoomed too far out or zoomed too far in.

Setting the Grids and Units

All grids are set in the Options Tab of the Document Options dialog. This Tab also allows you to toggle the units and the visible grid kind (dots or lines).



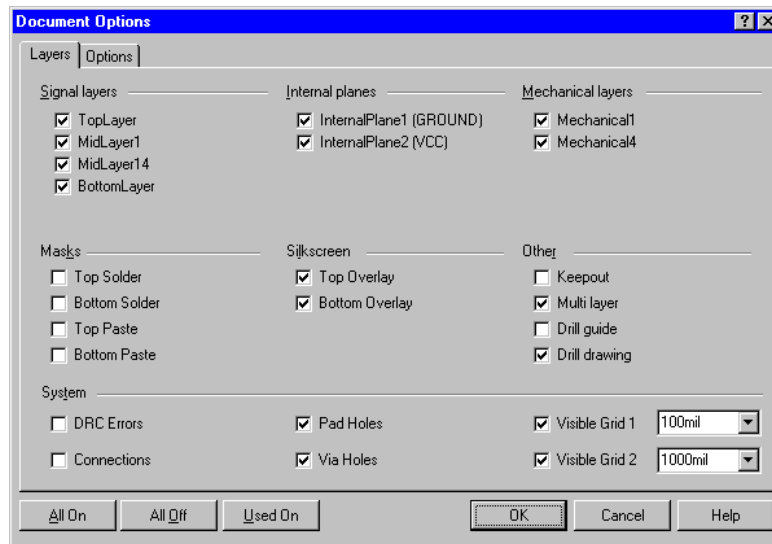
Set all the grids, toggle the units and enable the ECO feature in the Options Tab

Layers

The PCB Editor is a “layered” environment. You create your board design by placing objects on these layers. These layers are either “physical” layers, from which the fabrication information is created, or system layers, such as the Connect layer which displays the unrouted connections. Physical layers include the signal layers, internal plane layers, silkscreen, solder mask and paste mask layers. Each of the layers can be assigned a unique identifying color (select **Tools » Preferences** to set the color).

◆ Signal and plane layers are added in the Layer Stack Manager dialog, mechanical layers are added in the Setup mechanical Layers dialog.

This concept of multi-layered design distinguishes the PCB Editor from many other drawing or design applications. Although all the layers in your design can be viewed simultaneously, you will need to select individual layers for some tasks, such as placing an object which belongs to a single layer; typically tracks, polygons, fills or text strings.



Enable the required layers in the Layers Tab of the Document Options dialog.

Defining the Active Layers

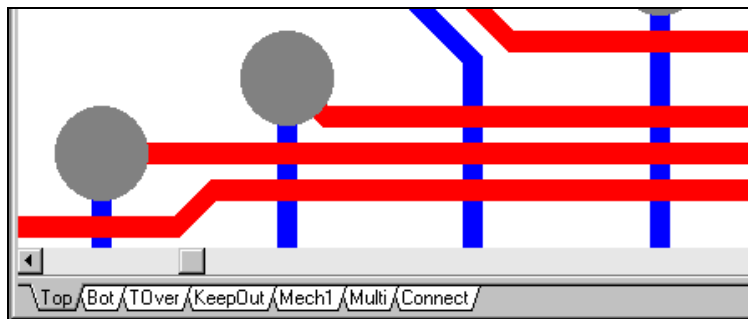
Before you can access any of these layers, the layer must be available in the workspace. Refer to the chapter, *Defining the Board* for information about making layers available in the workspace. Once the layers are made available in the workspace their visibility is controlled in the Document Options dialog.

To display layers:

1. Select the Layers Tab in the Document Options dialog (**Design » Options**).

Note how the layers are grouped by layer type. For each of the layers there is a check box next to the layer name, which you can click (LEFT MOUSE) to turn the layer on or off. A tick in the check box indicates that this layer is active. Any layers that you have activated will be active the next time you open this design.

2. Click in the layer check boxes to activate the required layers.
3. Click OK to close the Preferences dialog.



A Tab for each active layer will appear at the bottom of the document window.

The Current Layer

One workspace layer is “current” at any given time. At the bottom of the workspace there is a Tab for each active layer. The Tab for the current layer will be displayed on the top. Some items, such as tracks, text, fills or single-layer pads are placed on the current layer. Other items, such as components, multi-layer pads and vias can be placed without regard to the current layer. Selection (for moving, deleting, etc) is layer-independent – you can perform these operations on any primitives without changing the current layer.

◆ Click on a layer tab to make that layer the current layer, or use the + and - keys on the numeric keypad to toggle through all the active layers. Press the * key on the numeric keypad to toggle through active signal layers.

Signal Layers

There are 32 signal layers for routing. Anything placed on these layers will be plotted as solid (copper) areas in PCB artwork. As well as tracks, other primitives (area fills, text, polygon planes, etc) can also be placed on these layers. Signal layers are made available in the workspace by adding them in the Layer Stack Manager dialog. These layers include:

Top – component side signal layer

Mid Layers – inner signal layers (numbered Mid Layer 1–30)

Bottom – solder side signal layer

Internal Planes

Sixteen solid copper plane layers are available. Plane layers are made available in the workspace by adding them in the Layer Stack Manager dialog. Nets can be assigned and automatically connected to these planes and individual component pins can be assigned to internal planes at any time. Special *thermal relief* pad shapes are an option when plotting internal plane artwork. These planes are displayed (and printed/plotted) in the negative for efficiency. In other words, placing any primitive on these layers will create a void in the copper. These plane layers can also be split into two or more regions for different nets, refer to the topic *Creating a Split Power Plane* in the chapter, *Manually Routing the PCB*, for information on how to do this.

Silkscreen Overlay layers

Top Overlay and Bottom Overlay (or “silkscreen”) layers are typically used to display component footprint outlines, and component text (designator and comment fields that are automatically added to the footprint as it is placed from the Library). Component footprints are normally built as Top Layer footprints in the PCB Library Editor. If the component is placed or moved to the bottom layer, these items are automatically mirrored and moved to the Bottom Overlay. You can also include other primitives, such as free text strings, on the Overlay layers.

Mechanical Layers

Sixteen mechanical drawing layers are provided for fabrication and assembly details such as dimensions, alignment targets, annotation or other details. Mechanical layer items can be automatically added to other layers when printing or plotting artwork. Mechanical layers are made available in the workspace by adding them in the Setup Mechanical Layers dialog.

Mask Layers

Solder masks

Top and bottom masks are provided for photo or silkscreen solder masks. These automatically generated layers are used to create masks for wave soldering, usually covering everything except component pins and vias. You can control the expansions for these masks when printing/plotting by including a Solder Mask Expansion rule. Refer to the chapter, *Specifying the PCB Design Requirements*, for more information on the Solder Mask Expansion rule. This chapter also includes tips on masking all the vias. These layers are plotted in the negative, for efficiency.

Paste masks

Top and bottom masks are provided for photo or silkscreen masks of solder paste locations for boards with surface mount devices (SMDs). You can control the expansions (or contractions) for these masks by defining a Paste Mask Expansion design rule. Refer to the chapter, *Specifying the PCB Design Requirements*, for further information. These layers are automatically generated and are plotted in the negative, for efficiency.

Drill Layers

Drill Drawing

Coded plots of board hole locations, typically used to create a manufacturing drawing. Individual layer pair plots are provided when blind/buried vias are specified. Symbols are plotted at each hole location. Three symbol styles are available; coded symbol, alphabetical codes (A, B, C etc) or the assigned size. A table of symbols, metric and imperial hole sizes and hole counts can be included in the plot. Refer to the *Generating Output* chapter for more information.

Drill Guide

Plots of all holes in the layout – sometimes called *pad masters*. Individual layer pair plots are provided when blind/buried vias are specified. These plots include all pads and vias with holes greater than zero (0) size. Refer to the *Generating Output* chapter for more information.

Other Layers

Keep Out

This layer is used to define the regions where components and routes can validly be placed. For example, the board boundary can be defined by placing a rectangular perimeter of tracks and arcs, defining the region within which all components and routes must be placed. “No-go” areas for mechanical objects can be created inside this boundary by blocking off regions with tracks, arcs and fills. Keep outs apply to all copper layers. The basic rule is; components can not be placed over an object on the Keep Out layer and routes can not cross an object on the Keep Out layer.

Multi Layer

Objects placed on the multi layer will appear on all copper layers when output is generated. The multi layer is typically used for through-hole pads and vias.

Connect

This option controls the display of the *connection lines*. The PCB Editor creates connection lines on the connection layer when it locates part of a net that is unrouted.

DRC Errors

This option controls the display of the DRC errors.

Visible Grids

Controls the display of the two visible grids. Grids can be displayed as either dots or lines (set in the Options Tab).

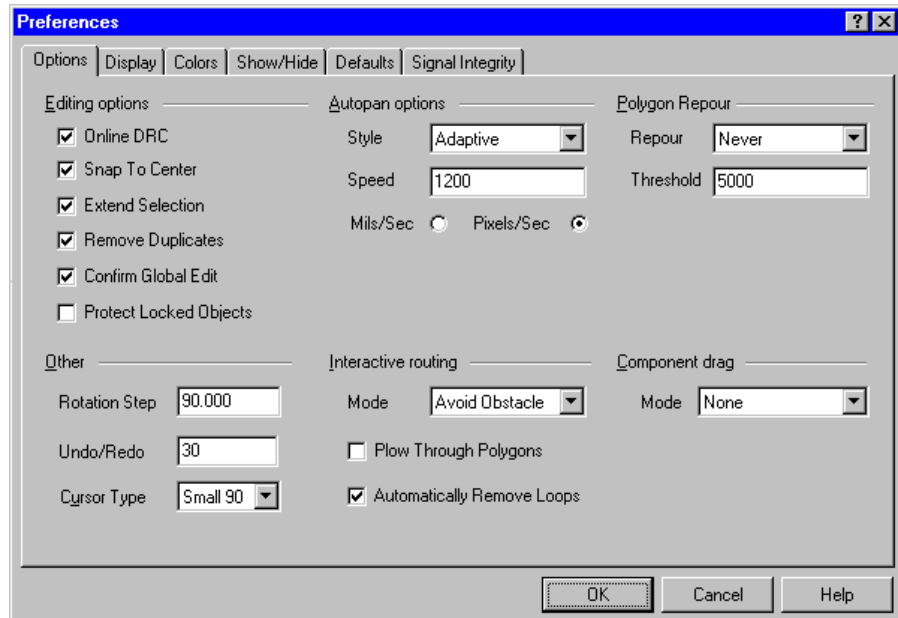
Pad and Via Holes

Controls the display of pad and via holes. To be able to distinguish pads from vias in draft mode, pad holes are outlined in the current Pad Holes color (set in the Colors Tab of the Preferences dialog).

PCB Workspace Preferences

User-definable workspace preferences are set in the Preferences dialog (select the **Tools » Preferences** menu item). The Preferences dialog is divided into four Tabs.

PCB Preferences - Options Tab



Editing Options

Online DRC

Enables online checking to ensure that the object currently being placed in the workspace does not violate the design rules. The design rules are defined in the Design Rules dialog (select the **Design » Rules** menu item). The rules that will be tested for are selected in the On-line Tab of the Design Rule Check dialog (select **Tools » Design Rule Check**).

Snap to Center

Snaps to the center when moving a free pad or via, snaps to the reference point of a component, snaps to the vertex when moving a track segment. The object will be “held” at the cursor location if this option is disabled.

Extend Selection

Selection is cumulative with this option enabled. With it disabled all currently selected objects are de-selected each time a new selection is made.

Remove Duplicates

With this option enabled a special pass is included when data is being prepared for output. This pass checks for and removes duplicate primitives from the output data.

Confirm Global Edit

Pops a dialog reporting the number of objects which will be altered by the global edit and allows you to cancel.

Protect Locked Objects

If this option is enabled locked objects can not be moved, and are ignored if they are part of a selection that you are moving.

Other

Rotation Step

When an object that can be rotated is floating on the cursor, press the SPACEBAR to rotate it by this amount in an anti-clockwise direction. Hold the SHIFT key whilst pressing the SPACEBAR to rotate it in a clockwise direction.

Undo/Redo

Set the stack size to specify how many of the previous operations can be undone. Set the stack size to zero to empty the Undo stack.

Cursor Type

Set the cursor to small or large 90 degree cross, or small 45 degree cross.

Autopan Options

Style

If this option is enabled it is possible to autopan when the cursor includes a cross hair. There are six Autopan modes:

Re-Center – Re-centers the display around the location where the cursor touched the Window edge. It also holds the cursor position relative to its location on the board, bringing it back to the center of the display.

Fixed Size Jump – Pans across in steps defined by the Step Size. Hold the SHIFT key to pan in steps defined by the Shift Step Size.

Shift Accelerate – Pans across in steps defined by the Step Size. Hold the SHIFT key to accelerate the panning up to the maximum step size, defined by the Shift Step Size.

Shift Decelerate – Pans across in steps defined by the Shift Step Size. Hold the SHIFT key to decelerate the panning down to the minimum step size, defined by the Step Size.

Ballistic – Panning speed is determined by the distance the cursor is moved outside the workspace.

Adaptive – Panning speed is set either as the number of mils per second, or pixels per second.

Step Size

Specifies the amount the display should shift each time the cursor touches the Window edge. The numerical value is in the current measurement units.

Shift Step Size

Specifies the amount the display should shift each time the cursor touches the Window edge when you are using one of the Shift autopan options. The numerical value is in the current measurement units.

Interactive Routing

Interactive Routing Mode

Ignore Obstacle – If you select this option you can place primitives anywhere in the workspace. If the Online DRC feature is enabled clearance violations are flagged immediately.

◆ Use the SHIFT+R shortcut keys to cycle through the modes as you are routing.

Avoid Obstacle – If you select this option you can only place primitives where they do not violate any clearance design rules. This feature is particularly useful when routing as it allows you to route hard up against existing objects, without fear of violating any clearance rules. Refer to the *Routing Your Design* chapter for more information about using this feature.

Push Obstacle – If you select this option placed primitives will be “pushed” as you route a net (in accordance with the design rules). If you tracks you are pushing can no longer move the routing mode reverts to Ignore Obstacle.

Automatically Remove Loops

With this option enabled loops that are created during manual routing will be automatically removed. Enable this whenever you need to reroute.

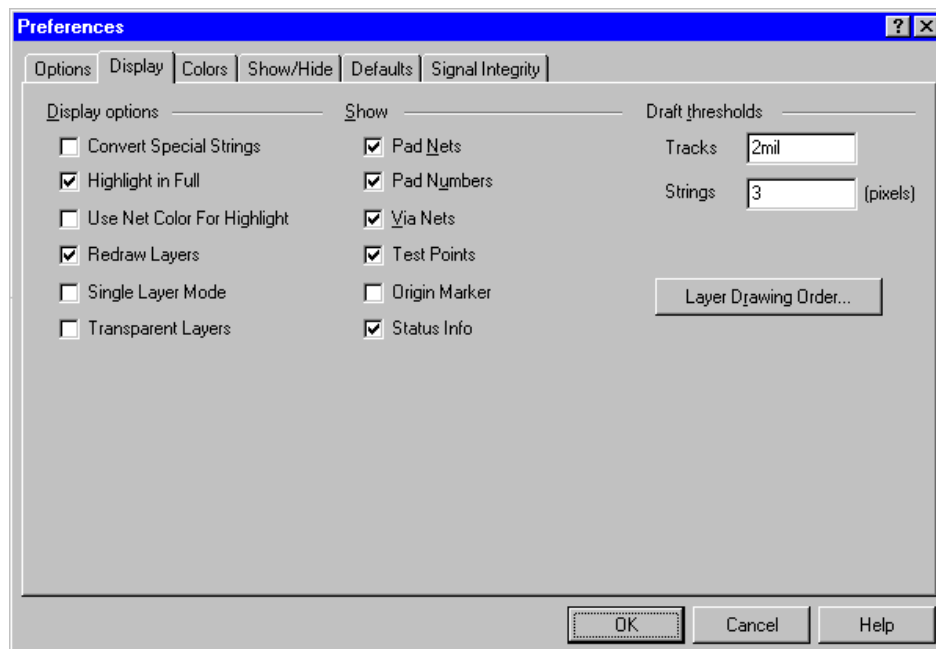
Plow Through Polygons

When this option is enabled you can route over the top of a polygon – the polygon is automatically repoured (depending on the Repour Polygon settings).

Component Drag

This option determines how tracks are dealt with when moving a component. The Enclosed Tracks option will move tracks that pass under the component as well as connected tracks. The Connected Tracks option will only drag tracks which connect to the component.

PCB Preferences – Display Tab



Display Options

Convert Special Strings

Special strings, such as .LAYER_NAME or .PRINT_DATE, are interpreted on screen as well as when output is generated. Refer to the *Strings* topic in the *PCB Design Objects* chapter for information about using special strings.

Highlight in Full

Completely highlights the selected object in the current selection color. With this disabled the selected object is outlined in the current Selection color.

Use Net Color For Highlight

Highlights the selected net in the net color (assigned in the Change Net dialog). Use with the Highlight in Full option for better results.

Redraw Layers

Forces a screen redraw as you toggle through layers, with the current layer being redrawn last.

◆ To redraw the current layer only use the ALT+END shortcut keys.

Single Layer Mode

Displays the current layer only. Provides a method of examining what will be output on each layer. If the current layer is a signal layer, multi layer objects are also displayed. Use the “+” and “-” keys to toggle through the layers, press END to redraw the screen.

Transparent Layers

Gives layer colors a “transparent” nature by changing the color of an object which overlaps an object on another layer. Allows objects which would otherwise be hidden by an object on the current layer to be readily identified.

Show

Control the display of the pad numbers, net names, testpoint labels, the origin marker and the Status bar info that appears when the cursor is held over a workspace object.

Draft Thresholds

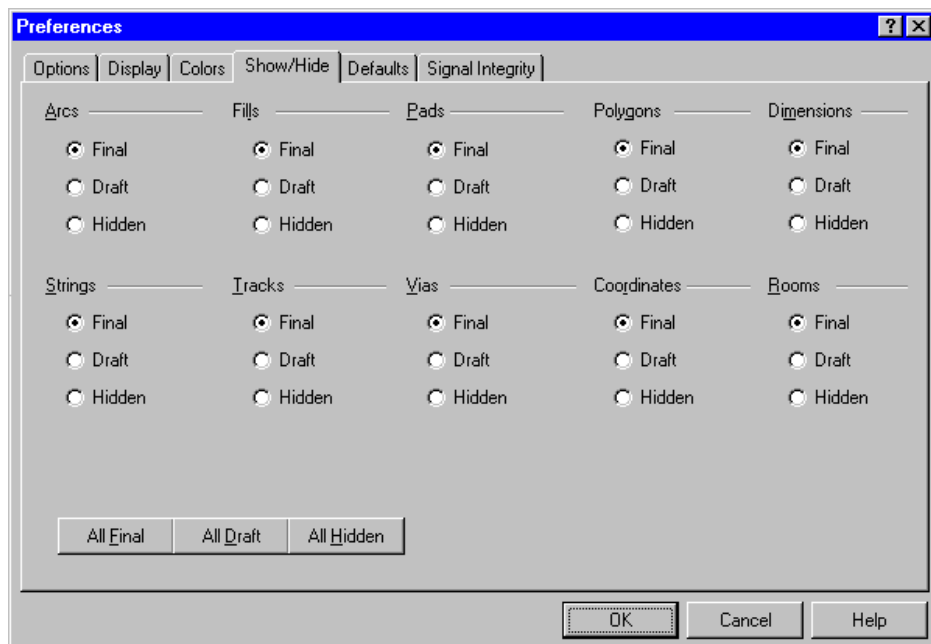
Tracks of this width or narrower will be displayed as a single line, tracks of greater width will be displayed as an outline (when tracks are displayed in Draft Mode).

Strings which are this many pixels high or more (at the current zoom level) will be displayed as text, text which are fewer pixels high will be replaced by an outline box. Set these thresholds as required.

Layer Drawing Order

The PCB Editor allows you to control the order in which layers are redrawn. Press the Draw Order button to pop up the Layer Drawing Order dialog. The order that the layers appear in the list is the order they will be redrawn in. The layer at the top of the list is the layer which will appear on top of all other layers on the screen.

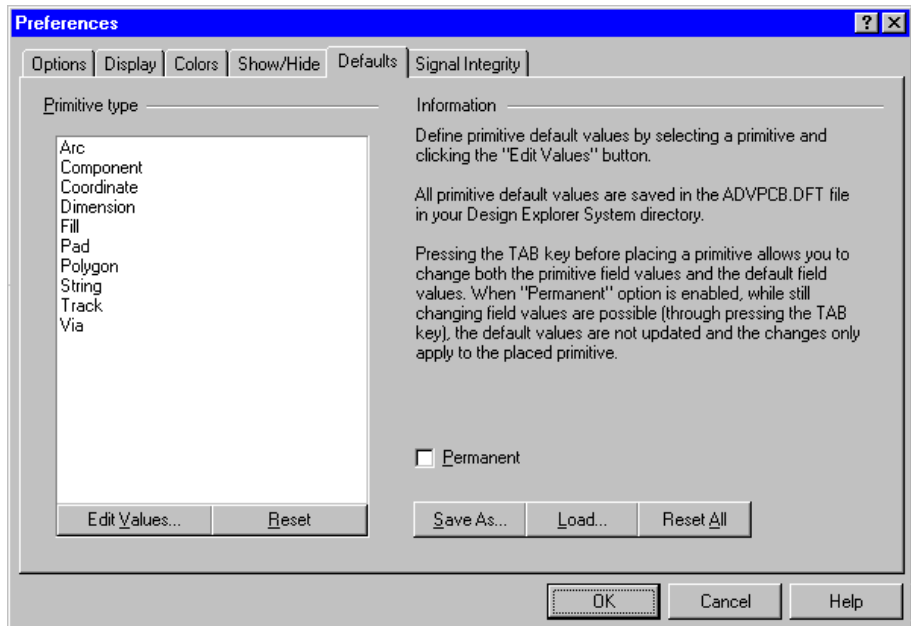
PCB Preferences – Show/Hide Tab

**Display Mode**

The display mode sets how design objects will be displayed on the screen. There are three possible modes; Final, where each object is displayed as solid; Draft, where each object is displayed as an outline (influenced by the draft threshold); and Hidden. Click to select the preferred display mode for each object type, or use the “All” buttons to switch all simultaneously.

PCB Preferences - Default Primitives

The default settings for each of the design objects available in the PCB Editor are configured in the Default Primitives Tab of the Preferences dialog.



Refer to the *PCB Design Objects* chapter for details about setting the attributes of each design object.

- ◆ These defaults can be changed “on-the-fly” during object placement, by pressing the TAB key while the object is floating on the cursor. The changes made on-the-fly will not affect the defaults if the Permanent option is enabled in the Default Primitives Tab.

Creating, Opening and Saving PCB Documents

For general information about creating and saving documents in the Design Explorer refer to *The Design Explorer* section. This chapter only discusses PCB specific information.

Creating a New PCB

The PCB Editor includes a PCB Creation wizard. This wizard allows you to select from one of the many industry standard board templates, or one of your own custom templates.

To launch the PCB Wizard select **File » New**, and then click on the Wizards Tab. Double-click on the PCB Wizard icon to start the Wizard.

Save Copy As Option

To save the active document with a new name, or in the Protel Text format, select **File » Save Copy As** from the menus.

Importing and Exporting older Protel PCB File Formats

Protel 99 SE can read all versions of older Protel PCB files. There are two approaches to importing an older format Protel PCB design into Protel 99 SE.

You can import the PCB as a new document into the Design Database. If you are unfamiliar with this process refer to the section, *The Design Explorer* for more information on importing documents into a Design Database. Once the document has been imported, double-click to open it. As the document opens all the design information is translated into the Protel 99 SE PCB format.

The second approach is to import the PCB data into an open PCB document. To import into an open PCB Document select **File » Import** from the PCB Editor menus. In the Import File dialog set the Files of Type field to the required Protel file format, then locate and select the required PCB document. When you click the Open button the PCB data is imported into the active PCB workspace.

The PCB file format has changed in Protel 99 SE, the file format is now referred to as PCB 4.0. To save a PCB back to the previous format (PCB 3.0) which was used by Advanced PCB V3, Protel 98, Protel 99 and Protel99 SP1, select **File » Save Copy As** from the PCB Editor menus.

Working in the PCB Design Window

Protel 99 SE's PCB server includes two document editors, the PCB Layout Editor and the PCB Library Editor. Working in either of these document editors is quite similar, you build your design from the set of objects provided, placing these objects in the workspace. The strategies used for placing objects, editing their attributes, positioning and deleting them in the workspace, and so on, are common to both document editors. In essence, the way you work in either editor is the same, what you do in each editor is different.

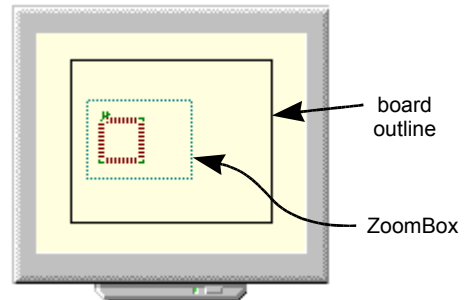
In the PCB Library Editor you create, edit and validate components and component libraries. In the PCB Editor you create, edit and validate Printed Circuit Boards.

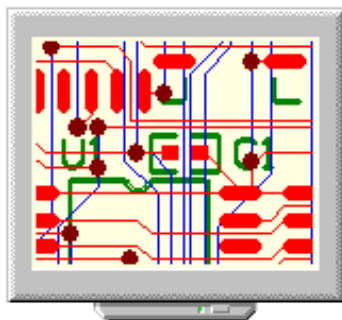
Changing Your View of the Workspace

Each sheet you open will appear in its own window. You look “through” this window to view your design. You can bring the design closer to you (zoom in), or move the design away (zoom out). The PCB Editor Panel and the View menu provide a number of ways of changing your view of the workspace.

The PCB MiniViewer

Use the MiniViewer in the PCB Editor Panel to change your view of the workspace. The dotted ZoomBox marks the current zoom level in the main document window (if the board has a boundary on the Keep Out layer or one of the mechanical layers). Click and drag on one of the ZoomBox corners to change the zoom level in the main document window. Click anywhere within the ZoomBox and drag it to change the region being viewed in the main document window.





Use the Magnifier to present a magnified view of the board in the MiniViewer. Click on the Magnifier button, then simply move the magnifying glass cursor over the board in the main window to examine a particular area in detail. The Configure button allows you to change the magnification level. Alternatively, press the SPACEBAR while using the magnifying glass cursor to toggle through the three magnification levels.

View » Fit Document

Selecting this menu item will change the view to display all objects in the workspace. (shortcut: V, F)

View » Fit Board

Selecting this menu item will change the view to display the entire board, based on the board keep out boundary. (shortcut: V, D)

View » Area

Select this menu item to re-define the display area. Click to define the first corner, then drag the dotted zoom window to define the new display area. (shortcut: V, A)

View » Around Point

Select this menu item to re-define the display area. Click to define the center point, then drag the dotted zoom window to define the new display area. (shortcut: V, P)

View » Zoom Options

The **View » Zoom In** menu item will bring the design closer to you, relative to the cursor position on the board (shortcuts; V, I or PAGEUP).

View » Zoom Out will move the design away from you (shortcuts; V, O or PAGEDOWN). This is also relative to the cursor, so position the cursor first.

The **View » Zoom Last** menu item will return you to your last view of the screen (shortcut: V, L). Repeatedly pressing V, L allows you to toggle back and forth between views.

♦ Hold the SHIFT key while pressing PAGEUP or PAGEDOWN to zoom in and out in smaller increments.

Moving Around the PCB Workspace

Scrolling

Scroll bars allow you to scroll around the workspace. These scroll bars have a sliding button, which you can click and drag to scroll up and down or left and right across the workspace. The position of the sliding buttons gives an indication of where in the workspace you are currently viewing. Click above or below the sliding button to scroll across the workspace in large steps, or click on the arrows at each end of the scroll bars to scroll in small steps.

Manual Panning

To pan across the workspace without using the scroll bars, select the **View » Pan** menu item (shortcut: V, N or HOME). This will re-center the screen around the current cursor position. The cursor will remain where it was, allowing you to continue panning.

You can also pan across the workspace by pressing one of the arrow keys. The cursor will move one snap grid increment with each press. Holding the SHIFT key as you press an arrow key moves the cursor in steps of 10 times the current snap grid.

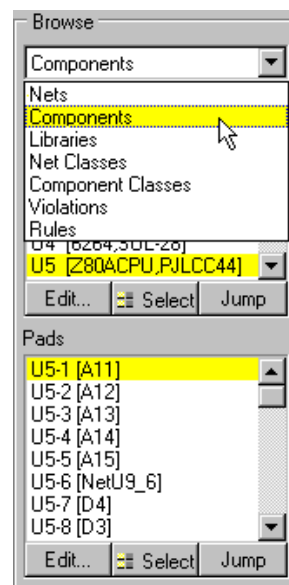
Autopanning

Autopanning is enabled whenever you have a cross hair cursor. You have a cross hair cursor whenever you perform an “edit” type operation such as placing, selecting, moving or deleting an object. This cursor can be moved either by moving the mouse or pressing the arrow keys on the keyboard. If the cursor is moved such that it hits the window frame, you will pan across the workspace. The PCB Editor has four Autopan modes. The mode is set in the Options Tab of the Preferences dialog.

Browsing the PCB Workspace

The PCB Editor Panel can be used for browsing through the Current PCB. There are six Browse modes specifically for this task; Nets, Components, Net Classes, Component Classes, Violations and Rules.

- If you browse by Net or Component the small MiniViewer in the panel will display the selected net or component.
- Use the Zoom button to quickly locate and identify the selected net or component in the main document window.
- When you select an item in a list information about that item will appear on the Status Bar.



Jumping Around the PCB Workspace

The **Edit » Jump** menu (shortcut: J) allows you to conveniently locate a specific component, net, pad on a component, text string or board location without having to zoom, pan or scroll through multiple screens.

With all of these options, the PCB Editor will position the cursor on the target, only redrawing the screen if the search target is outside the current display area. When a redraw is needed, the target will be centered in the active window.

Jump options include:

Absolute Origin

Jump to the absolute origin. This is the lower-left corner of the workspace. (shortcuts; J, A or CTRL HOME)

Current Origin

Jump to the current (or relative) 0,0 origin (shortcuts; J, O or CTRL END). This origin is specified by selecting the **Edit » Origin » Set** menu item.

New Location

Jump to the specified location. **Edit » Jump » New Location** pops up the Jump To Location dialog. The X and Y location text fields will contain the current cursor position. The cursor will jump to the specified location. (shortcut: J, L)

Component

Jump to the specified component. **Edit » Jump » Component** pops up the Component Designator dialog. Type in the designator and click OK. If you do not know the designator, type ? and press ENTER or click LEFT MOUSE to scan the board for all placed components. Choose from the Components Placed dialog and click OK. The cursor will jump to the reference point of the selected component. (shortcut: J, C)

Net

Jump to a pin on the specified net. Type the net name in the Net Name dialog and click OK. If you do not know the net, type ? and press ENTER or click LEFT MOUSE to scan the board for all nets. Choose from the Nets Loaded dialog and click OK. The cursor will jump to the nearest pin that belongs to the selected net (shortcut: J, N).

Pad

Jump to the specified pin on the specified component. Type the component designator and pin number in the Jump to Pin Number dialog (eg U1-6) and press ENTER. The cursor will jump to the center of the pin. (shortcut: J, P)

String

The cursor will jump to the named string. The system will perform three searches:

First – for a string that matches the specified string in both case, characters and length.

Then – for a string with same characters in it but perhaps having more characters.

Finally – for a string with same characters but ignoring case.

For example, typing “component” would find the string “component” first. If no match is found it would next find the string “components” and finally “CompONENT”. When the string is found, the cursor will be relocated to the specified string. (shortcut: J, S)

Error Marker

Select this menu item to jump to the first DRC error marker. Repeating will jump to a second error marker, and so on. Removing the violation will clear the error from the “Jump to” list. Otherwise, repeating will continue to cycle through all errors in the current document window.

Selection

Select this menu item to jump to the first selected object. Repeating will jump to a second selected object, and so on. Repeating will continue to cycle through all selected objects in the current document window.

◆ Jumping around the workspace can be a very efficient way of working in the PCB Editor, as it allows you to reposition your view of the workspace without zooming. To speed the process even more, all the Jump process launchers can be executed by using the shortcut keys to pop the dialog. For example, to jump to the location 1000, 1000, press the J, L shortcut keys. When the Jump To Location dialog pops up the X-Location text box will be highlighted. Highlighted text is replaced by whatever you type, simply type the new X-location in. To move to the Y-Location text box, press TAB. Type the new Y-Location in. Press ENTER on the keyboard. The dialog will close and the cursor will jump to the location 1000, 1000.

Editing PCB Objects

There are two approaches to editing an object in the PCB Editor, either by changing its attributes in its Change dialog or graphically modifying the object. Certain operations (such as changing the color of a primitive) can only be performed by editing the attributes, others (such as re-sizing a fill) can be done through the dialog or by graphically modifying the object.

To edit the look of an object in the workspace it is generally easier to do it graphically. To do this you must first bring the object into *focus*. To edit the object through its Change dialog select the **Edit » Change** menu item and then click on the object.

Editing While Placing

You can edit the attributes of an object while it is being placed. While the object is floating on the cursor, press the TAB key. This pops up the object's dialog. The advantages of editing during placement are:

- Changes made to attributes can become the defaults for that type of object. These changes are stored in the defaults file ADVPCB.DFT. Note – this method of setting defaults depends on the setting of the “Permanent” check box in the Preferences dialog (**Tools » Preferences**). If this is on, these changes will not become the defaults and will only apply to the object floating on the cursor.
- Objects that have a numeric identifier, such as pad designators, will auto-increment.
- There is no need to edit the object after it has been placed, speeding the design process.

Changing Placed Objects

The **Edit » Change** menu item is used to modify a placed object. Each object has its own range of editable attributes. You can change one object or extend changes across your entire design using powerful global editing options.

To change any placed object select the **Edit » Change** menu item, move the cursor over the item and click LEFT MOUSE (shortcut: double-click LEFT MOUSE). If there is more than one object under the cursor a selection list will pop up.

◆ Refer to the *PCB Design Objects* chapter for details of the editable attributes of each object and the *Global Editing* topic in this chapter for tips on Global editing.

Editing Graphically – Focus and Selection

One of the advantages of a graphical-based editing environment is the ability to make changes directly to objects displayed on the screen. To be able to graphically edit an

object or a group of objects you must first identify the objects. This is done through *focus* or *selection*.

In other Windows applications selection is a single concept, the process of choosing or designating objects as a prerequisite for modification. A typical example would be selecting one or more objects to be *copied* to the clipboard and then *pasted* to another location. Often selected objects can be modified directly. For example, selected objects can be moved or re-shaped in most graphical applications.

Unlike other Windows applications, the PCB Editor uses two independent methods for accomplishing selection oriented tasks. These methods, *selection* and *focus* are used repeatedly when creating or editing your PCB. Breaking selection into these two independent processes allows you to perform complex modifications, which would be either difficult or impossible using the simple selection method described above.

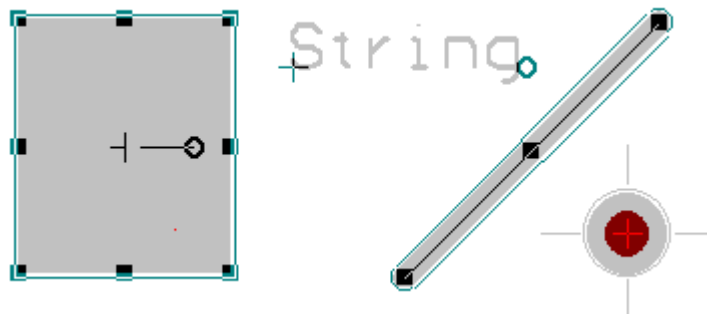
◆ Focus and selection provide two distinct and independent methods for changing objects in the workspace. These two methods distinguish Protel 99 SE's PCB Editor from other Windows applications, where focus and selection are normally merged into a single operation.

Focus

When you position the cursor over a design object and click the LEFT MOUSE button, the object then “has the focus”, and the way it is displayed changes. This is similar to the way you can change the focus in Windows by clicking on an open window to make it active.

Only one object can be in focus at a time. You can tell which object is currently in focus because its graphical editing handles and/or a focus cross-hair are displayed. For example, if you click LEFT MOUSE over a fill a re-sizing handle will appear at each corner and along each side, and a rotation handle will appear near the center. If you click LEFT MOUSE over a via a focus cross-hair will appear. To move the focus to another object click on that object. Click in a clear area of the workspace to release the focus.

Graphical editing



Objects in focus, displaying their focus handles

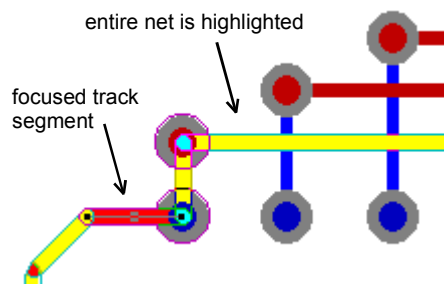
When an object is in focus, you can move the object or edit its graphical characteristics. For example, you can change the size or shape of a fill by dragging the square re-sizing handles. Click LEFT MOUSE on the circular rotation handle to rotate a fill or a string.

When you focus a track segment three editing handles appear, one at each end and one in the center. Click on one of the end handles to move that end or click on the center handle to “break” the original track segment into two segments.

Click anywhere on a focused object to move it. For strategies about moving or dragging objects refer to the *Moving and Dragging* topic later in this chapter.

Focusing a Track Segment that Belongs to a Net

When you click on a track segment it comes into focus. When you click on a track segment that belongs to a net, as well as the segment of track that you clicked on coming into focus, all other objects that belong to the same net will highlight. This makes it very easy to trace a net through your design. Remember, this feature only operates when the objects have a net name.



Summary

As illustrated in these examples, *focus* is a prerequisite to a number of graphical editing functions that are performed on individual objects. Note that you cannot use the clipboard menu items: **Edit » Copy**, **Cut**, **Paste** or **Clear** with the focused object. These Clipboard features work only on a *selection*.

Selection

Selection provides a second, distinct method of manipulating objects. Unlike focus, selection can be used with both individual objects and with groups of objects.

◆ Selection works with the **Edit » Copy**, **Edit » Cut**, **Edit » Paste** and **Edit » Clear** menu items.

Unlike *focus*, described above, selection does not display an object's graphical editing handles or a cross-hair. Instead, the object is outlined in the *selection color* (**Tools » Preferences**).

Once selected, objects can be moved, grouped, un-grouped, exported to another file, cut, copied, pasted into another window or location in the current window, or cleared. Special **Edit » Move** process launchers allow selections to be moved or rotated in a single operation. Selection also works with the PCB Editor's global editing feature, which can limit global changes to selected or un-selected objects.

◆ A handy feature of the PCB Editor's selection options is the ability to click LEFT MOUSE without *de-selecting* objects that were previously added to the current selection. This allows you to perform a wide variety of operations, without effecting the current selection.

Selection in the PCB Editor can also operate in two modes. Selection can be cumulative (extendible), where objects remain selected until specifically de-selected. The other mode is non-cumulative. In the non-cumulative mode all currently selected objects are de-selected when one of the **Edit » Select** menu items is chosen. The extend selection mode is toggled in the Options Tab of the Preferences dialog.

Selections are made in the following ways:

- Direct selection, using SHIFT+LEFT MOUSE to add (or remove) individual items to the current selection.
- The click-and-drag-a-window-around shortcut.
- Use the **Edit » Select** and **Edit » DeSelect** sub-menus to define a selection.
- By using the Selection field in Change dialogs. This option allows you to use the PCB Editor's global editing feature to apply selection status changes to other primitives in the current board window. Refer to the *Global Editing* topic in the *Working in the PCB Editor* chapter for tips on global editing.

◆ Care must be taken when manipulating selections to ensure that the current selection includes *only* the desired objects. It is good practice to select **Edit » DeSelect All** from the menus (shortcut: X, A) to clear the current selection, prior to making a new selection.

◆ If something unexpected happens, select **Edit » Undo** to restore the design to the previous state. Multiple Undos can be performed.

Displaying Selections

Selections are outlined in the selection color specified in the Display Tab of the Preferences dialog. There are a number of ways of affecting the visibility of a selection.

Draft Mode

If the primitive has its display mode set to Draft, it will be outlined in the selection color and displayed in Final mode when it is selected. This makes it very easy to identify. Set the display mode in the Show/Hide Tab of the Preferences dialog.

Highlight In Full

To display the entire primitive in the selection color, enable the Highlight In Full option in the Display Tab of the preferences dialog.

Use Net Color For Highlight

Nets can also be highlighted in their net color. Set the net color in the Change Net dialog (set the Browse mode in the panel to Nets, select the net and press the Edit button). This option works well in combination with the Highlight In Full option. Enable the Use Net Color For Highlight option in the Display Tab of the preferences dialog.

Making Selections

Direct Selection of an Individual Object

Direct selection is the most flexible way to select an individual object. To select one object at a time:

1. Hold down SHIFT and click LEFT MOUSE with the cursor positioned over an object.

The item will be redrawn, outlined in the Selection color (**Tools » Preferences**). You can do this repeatedly, each time adding another item to the current selection.

If you hear a “beep” or nothing appears to be selected, try zooming in closer (press PGUP) and make sure that the cursor is directly over the item you wish to select. To select a component, position the cursor within the component outline. Components, especially complex components can take a moment to select.

To add another item to the current selection:

2. Hold down SHIFT and click LEFT MOUSE over another item.

To release individual items from the selection:

3. Hold down SHIFT and click on the selected item.

When released, the item will be redrawn in its original colors. Other selected items remain selected until they are either individually released (SHIFT+LEFT MOUSE) or until an **Edit » DeSelect** is executed.

Direct Selection of an Area

Direct selection can also be performed on an area. To select all objects within an area:

1. Position the mouse where there are no objects under the cursor.
2. Click and hold the LEFT MOUSE button.

The Status Bar will prompt “Select Second Corner”.

◆ The Extend Selection option (Options Tab of the Preferences dialog) determines whether selection is cumulative, or non-cumulative.

3. Drag the mouse diagonally away. Define the area with the selection rectangle and release the LEFT MOUSE button.

Only objects that fall entirely within the rectangle will be selected. The selected objects will highlight in the current selection color.

4. Repeat the process to extend this selection or to make a new selection.

The Select and DeSelect Sub Menus

The **Edit » Select** menu allows you to select all items inside or outside of an area, all items on one layer, or all free primitives (all items other than components). You can also select by physical net or off grid component pads.

Use the Select and DeSelect options to define complex grouping which can then be moved, copied or deleted.

The Select and De-select menus include:

Inside Area

Allows you to define a rectangular selection area.

Only those objects that lie completely inside the area are included. Free pads or vias are included in the selection if their center is inside the rectangle.

◆ Use the S shortcut key to pop up the Select sub menu, and the X shortcut key to pop up the DeSelect sub menu.

Outside Area

This option selects everything outside the selection rectangle. The rules for inclusion in the selection are the same as for **Select » Inside Area**. The procedure for defining the selection rectangle is the same as for Inside Area.

All

Selects everything placed in the document window. This includes all objects which have their display state set to hidden or are not visible because their layer is off.

Physical Net

This selection will include all primitives (tracks, vias, fills) that are in *physical contact* with the point where you click. It will not include any part of a net that is not physically connected, that is, connected by a connection line.

To use this feature:

1. Choose **Edit » Select Physical Net** (shortcuts; CTRL+H or S, N).

You will be prompted “Select Physical Net Starting Point”.

2. Position the cursor over any primitive within the desired net and press ENTER or LEFT MOUSE.

The continuous physical net, extending from the selection point, will highlight in the selection color.

All On Layer

This selection will include all primitives on the current layer. Multi layer items (typically multi-layer pads and vias) are excluded from the selection.

Free Objects

This option selects all objects which are not part of a group (component, polygon, dimension or coordinate). This feature is useful for stripping a routed, or partially routed board back to its “placed” condition. Limit the selection by turning off layers which contains objects you do not want selected.

All Locked

This will select all primitives and components that have their Locked attribute set.

Off Grid Pads

Choose **Edit » Select » Off Grid Pads** to select all component pads that do not fall on the current snap grid. Use this prior to autorouting to check how many component pads are off grid. Use the partner process, **Tools » Interactive Placement » Move To Grid** to bring the components onto the current snap grid (shortcut: A, G).

Selecting PCB Components from the Schematic

To help in the process of working between the 2 views of your design – the schematics and the PCB – you can directly select PCB components from the schematic. To do this select the components on the schematic sheet, then choose **Tools » Select PCB Components** from the Schematic Editor menus. These components will be selected on the PCB, and the PCB view zoomed to show the selection. With this set of selected components you can now create a component class, which is described in more detail on the following page.

Selecting from the Panel

As you manipulate your design one of the most commonly used tools is selection. Selection can be used to move, copy or delete a group of objects, and is also a convenient way of highlighting an object or a group of objects. As well as direct selection (SHIFT+click) and selection from the menus, you can also select directly from the Browse PCB Panel. The following selection operations are available:

- Select the current net, and any node in that net
- Select the current component, and any pad on that component

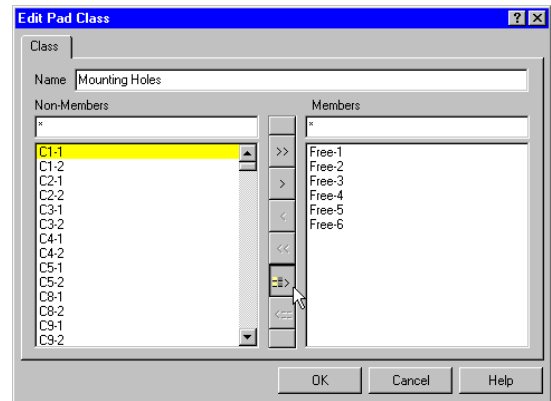
- Select the current net class, and any net in that net class
- Select the current component class, and any component in that class
- Select objects targeted by a design rule

Set the Extend Selection option in the **Preferences** dialog to create multiple selections, disable it to always replace the current selection with the new selection.

Creating Classes from Selections

Like selection, classes are an excellent tool for working on a set of objects. The advantage of classes over selections is that they are stored and can be used at any time. Classes can be used as the scope of a design rule, and also as a method of selection. Classes can be created for components, nets, pads, and from-tos.

A convenient way of creating a class is to create it based on the current selection. Select pads, components, or nets using the standard selection strategies, or the selection buttons in the Browse PCB Panel.



Creating a new pad class from the selected pads

To create a class from a group of selected pads select **Design » Classes** to pop up the Object Classes dialog, then click the Add button on the Pad Tab of the dialog to add a new class. In the Edit Pad Class dialog click the take-over-selected objects button to transfer the pads currently selected on the board from the Non-Members list to the Members list. Follow the same process for nets and components.

Stepping through Selected Objects

After selecting the objects you need to work on, use the Find Selection toolbar to quickly move from one selected object to the next. To display the toolbar press the B shortcut to pop up the **Toolbars** sub-menu and choose **Find Selections** from the menu.



Use the Find Selections toolbar to step through the selected objects

The top row of buttons are used to step through the currently selected primitive objects (tracks, pads, vias, arcs, fills and strings), the bottom row of buttons are used to step through the selected group objects (components, dimensions, coordinates and polygons).

Creating Selection Queries

Complex PCB selection queries can be created and stored with the new Query Manager. Select **Edit » Query Manager** to pop up the Query Manager dialog. To create a new query, first type in a Name then click the Add button to pop up the Statement dialog. Each statement specifies if an Object's Property is equal to (or not equal to) a Value. Define the statement and click OK to return to the Query Manager. Multiple statements can be added to a query, these are logically ANDed together.

When the query has been defined click the Apply button to select all objects that satisfy the query conditions.

Using the Selection Wizard to Create Complex Selections

The PCB Editor also includes a Selection Wizard, which you can use to create complex sets of selections. This Wizard allows you to simultaneously select different types of primitives, based on a set of user-definable selection criteria. For example, you could select all pads and vias with a hole size ≤ 0.5 mm, or all tracks whose width > 8 mils.

◆ Click on the Wizard button in the Query Manager dialog to launch the Selection Wizard.

Working with a Selection

The PCB Editor uses a special proprietary clipboard format that supports PCB data such as connectivity and layer attributes of primitives. This internal Protel clipboard is not the same as the standard Windows clipboard that allows you to move selections, such as text, between various Windows applications. The Windows MetaFile (.WMF) graphics format is not supported in the PCB clipboard.

Use the clipboard in the PCB Editor the same as you would in any Windows application. The sequence is; select the objects to perform the operation on, cut or copy the selection to the clipboard, then paste the clipboard contents to the desired location.

Notes on Using the Clipboard

- The PCB Editor includes a Clipboard Reference Location option. A reference location is a coordinate you nominate when you copy or cut the items. When you paste the selection, you will be “holding” it by this reference location, allowing you to accurately position the pasted objects. You will be prompted to “Choose Clipboard Reference Location” when performing a Copy or Cut.
- **Edit » Copy** copies the current selection to the clipboard.
- **Edit » Cut** clears the current selection from the workspace and copies it to the clipboard.
- Select **Edit » Paste** to paste the selection back into any open PCB document.

◆ You can copy PCB data to the Windows clipboard in the Power Print tool – refer to the chapter *Printing to a Windows Printing Device* later in this section of the Handbook for more information.

- Make sure that the selection includes only those items you wish to copy or cut. To ensure that nothing is selected before making a new selection use the deselect all shortcut: X, A.
- Use the shortcut SHIFT+LEFT MOUSE to add or remove items from the current selection.
- The clipboard holds the last selection only, each time you select Cut or Copy you overwrite the clipboard contents.
- Select **Edit » Clear** to delete the current selection from the workspace without copying it to the clipboard (shortcut: CTRL+DELETE).

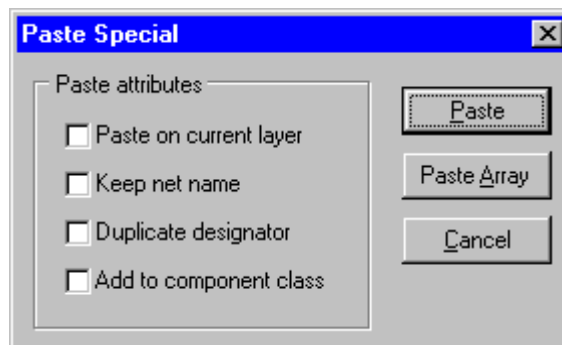
Special Paste Options

Paste Special allows you to control what happens to the attributes of the objects that are in the clipboard when they are pasted back into the workspace.

To control the attributes as you paste the current clipboard contents:

1. Choose **Edit » Paste Special**.

The Paste Special dialog will appear. Paste Special includes the following options:



Paste on current layer

If this option is disabled all single layer objects such as tracks, fills, arcs and single layer pads keep their existing layer assignments. If this option is enabled then all single layer objects are pasted onto the *current layer*.

Keep net name

If this option is enabled then all objects which have a net name will keep the assigned net name. If this option is disabled the net attribute is set to "No Net".

Duplicate designator

This option supports the creation of a PCB panel, where you wish to copy and paste the entire design. Typically the Keep net name option would be disabled if this option is enabled.

Add to component class

Pasted component(s) are added to the same class as the component(s) they were copied from.

2. Click OK after setting the options in the dialog.
3. Position the selection in the workspace and click LEFT MOUSE or press ENTER.

Creating a Panel

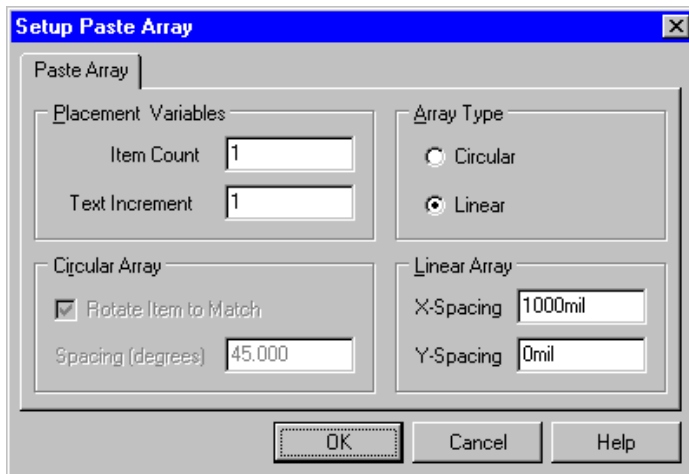
The PCB Editor can be used to create a panel of PCBs. This panel could be multiple copies of the same PCB, or be made up of different PCBs. To create a panel, first copy the PCB to the clipboard. Then use the Paste Special feature to copy the PCB the required number of times. If you have difficulty positioning accurately as you paste, use the Jump Location process (shortcut: J, L) while the paste is floating on the cursor. This allows you to position the paste without using the mouse. Remember, you can move through fields in a dialog by pressing the TAB key (SHIFT+TAB to go back through fields) and press ENTER instead of clicking OK.

♦ When you paste a PCB that includes a polygon you will be prompted to repour polygons – generally there is no need to repour polygons as you have not modified the polygon, or any of the objects that it is poured around. If you do repour the polygons you will lose all the polygon connections unless you enabled the Keep Net Names option in the Paste Special dialog – this option should be disabled for a multi-PCB panel to prevent the net data from the different PCBs being merged.

Pasting an Array of Objects

When you use **Edit » Cut (or Copy)** you are placing a copy of the current selection in the clipboard. **Edit » Paste Array** provides a powerful way to place multiple copies of the clipboard contents back into the workspace.

To paste an array of the current clipboard contents select **Edit » Paste Special** from the menus. Set the Paste Special dialog as required and press the Paste Array button. The Paste Array dialog will appear.



Define the array criteria in the paste Array dialog

Placement Variables**Item Count**

The number of repeat placements to be performed. For example, typing 4 will place 4 of the current clipboard contents.

Text Increment

This option is used for designators on pads and components. Setting this to 1 (default) will increment the designators in series, for example U1, U2, U3 etc.

Both alpha and numeric increments other than 1 are also supported. By setting the designator of a pad prior to copying it to the clipboard and setting the Text Increment field, the following types of pad designator sequences can be placed: Numeric (1, 3, 5); Alphabetic (A, B, C); Combination of alpha and numeric (A1, A2, or 1A, 1B, or A1, B1 or 1A, 2A, etc).

To increment numerically, set the Text Increment field to the amount you wish to increment by. To increment alphabetically set the Text Increment field to the letter in the alphabet that represents the number of letters you wish to skip. For example, if the initial pad had a designator of 1A and the Text Increment field was set to C (the third letter of the alphabet), the pads would have the designators 1A, 1D (three letters after A), 1G (three letters after D), and so on.

Array Type**Circular**

Repeat placements are made in a circular array using the rotation and spacing values specified under Circular Arrays.

Linear

Repeated items are placed in a linear array, using the spacing values specified under Linear Array.

Circular Array**Rotate Item to Match**

Array items will be rotated by the same angular amount as their Spacing.

Spacing

Specifies the angular spacing between each pasted item. The PCB Editor has an angular resolution of 0.001 degrees.

Linear Array

These values specify the X and Y distance between each item as it is placed.

Enter the desired values into the Setup Paste Array dialog and click OK. Follow the prompts on the Status Bar to paste the array.

Global Editing

As well as being able to edit the attributes of a single object, the PCB Editor also allows you to apply these edits to other objects of the same type. These may be other objects in the current component or in the current document.

Additionally, you can further define conditions that either extend or restrict global changes. For example, changes can be applied to all objects that are selected or all objects that are not currently selected. If desired, you can create a complex set of conditions for applying changes.

Virtually every attribute of an object can be globally edited. A simple example would be changing the size of pads associated with a specific component. In another instance you may wish to change the width of tracks for a particular net. These options (and many more) are possible with global editing. The possible applications for global changes are limited only by the imagination of the designer.

Each object's dialog may contain different options since each object type may have unique attributes.

♦ The large number of global change options may make this feature appear somewhat complex at first. However, the principles of applying global changes are reasonably simple, once understood. When mastered, this feature can be an important productivity tool that can save a great deal of manual editing of a PCB.

Global Editing Strategies

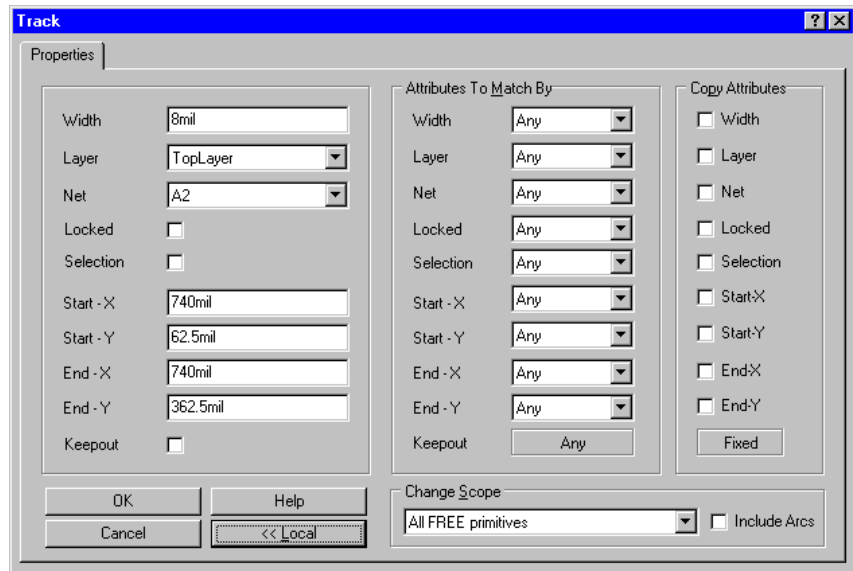
While the presentation of global change options may appear differently in the various object dialogs, the strategy used is always the same. This description will outline the approach to global editing.

Current Attributes

When you double-click on an object, you are presented with the Change dialog for that type of object. This dialog contains the *current* values or settings of the object's attributes.

Change the attributes you would like to alter.

Pressing the Global button will extend the dialog. It will now contain three distinct regions; Attributes, Attributes to Match By and Copy Attributes.



Note the three columns or regions each dialog has when the Global button is pressed.

Attributes to Match By

In the center of the dialog there will now be a column titled Attributes To Match By. In the Attributes To Match By column you define how to identify the other objects in the design which the global change is to apply to.

The Attributes To Match By column will contain either a choice field for each attribute or a text field which you can type in.

The choice field has three options: Same (apply global changes if this attribute is matched in the target object); Different (apply global changes if this attribute is not a match in the target object) and Any (the default) which applies the change irrespective of whether the attribute has the same value in both objects.

Use combinations of Match By attributes to define a particular set of objects to apply the change to.

- ◆ If the Match By attributes are all set to *any*, and the text fields contain the wildcard symbol (*), then the global change will apply to all objects of this type.

Copy Attributes

The third column in the dialog is titled Copy Attributes. This column will contain either a check box for each attribute or a text field which you can type in.

In this column you specify which of the attributes in the matched objects you want to copy the changes to, and if the attribute has a text field what new text value to copy to the matched objects.

Change Scope

The last parameter to set is the change scope. There are two options here; “All primitives” or “All FREE primitives”. The PCB Editor identifies any primitive which is not part of a group object as a *free primitive*.

Examples of Global Changes

The following examples will give you some idea of the potential scope of global changes that can be performed on components and primitives:

Example 1 – Swapping Track Layers

To move all Top layer tracks to the Bottom layer, regardless of track width or selection status:

1. Double-click on any Top layer track to open the Change Track dialog.
2. Set the Layer attribute to Bottom Layer.
3. Click the Global button to display the global edit parameters.
4. Under Attributes to Match By set Layer to Same. All other attributes should be set to Any.

This tells the PCB Editor to apply this change to *all* tracks on the *same* layer. Note that the match by condition is based on what the Layer was, not what you just changed it to.

5. In the Copy Attributes column the Layer attribute will have been automatically activated.

This tells the PCB Editor to copy the change made to the Layer attribute to all tracks that meet the Match By criteria.

6. Set the Change Scope to All FREE primitives. If your routing includes arcs then enable the Include Arcs option. Click OK or press ENTER.

The initial track that was edited will be changed to the Bottom layer first. The Confirm Global Change dialog will then pop up.

7. Click YES to accept the global change.

All Top layer tracks will be swapped to the bottom layer. You may need to redraw the screen (press END) to refresh the display.

If you wish to change a particular net to another layer, select the **Edit » Select » Physical Net** menu item and click anywhere on the desired net. Now repeat the global edit process as described, except in the Attributes To Match By column set the Selection attribute to Same and all others to Any. Only the selected net will move to the Bottom Layer.

Example 2 – Changing Via Sizes

To change all vias on a board to 40 mils:

1. Double-click on any via to open the Change Via dialog.
2. Type 40 in the Diameter field under Attributes.
3. Click on the Global button.
4. Set all the Attributes To Match By to Any (all vias will be changed).
5. In the Copy Attributes column the Diameter attribute will have been automatically activated.

Leave the other attributes disabled.

6. Set the Change Scope to All primitives and click OK or press ENTER.

The initial via selected will be redrawn with the new diameter and the Confirm Global Change dialog will open.

7. Click OK.

All vias in the document window will change to reflect the new size setting.

Example 3 – Locking a Net

To prevent the autorouter from modifying a particular net that was manually routed, lock it in place. To do this:

1. Select the net. To select a net choose the **Edit » Select » Physical Net** menu item and click on the net.

All the track segments that make up the net will highlight in the selection color.

2. Double-click on one of the selected track segments.

The Change Track dialog will pop up.

3. Enable the Locked attribute in the Change Track dialog.
4. Press the Global button.
5. In the Attributes To Match By column set the Selection attribute to be Same.
6. Set the Change Scope to All FREE primitives. If your routing includes arcs then enable the Include Arcs option.

The global change will be applied to all selected track segments that are not part of a group object.

7. In the Copy Attributes column the Locked attribute should be set.

This instructs the PCB Editor to copy the changes made to the Locked attribute of this track segment to all track segments that meet the Match By criteria.

8. Click OK.

The initial track selected will be changed and the Confirm Global Change dialog will open.

9. Click OK to lock the remaining selected track segments.

Summary

The three examples above show the most basic application of the global change options. With care and planning you will experience significant productivity benefits from this powerful feature. However, the very power of these options can contribute to some unanticipated results – particularly when complex selections are globally edited.

When in doubt, it's always safest to DeSelect All (X, A), then, rather than using the global edit to change the target objects, set it up to simply select the target objects. Visually confirm that the match by criteria has targeted the correct objects, then re-do the global edit, using Selection as the match-by criteria.

Use the auto-backup feature included in Protel 99 SE, and always archive your design, particularly if you intend to perform complex changes. Finally, remember that the Undo/Redo features can allow you to recover several operations, if required.

Moving and Dragging

To *move* an object is to reposition it without regard to objects in contact with it. For example, if you move a component, all the tracks connected to its pads will not move. If you *drag* the component, all the tracks will remain connected to the pads as you reposition it. Move operations can be performed on a selection or on individual objects.

♦ The concept of dragging is only applicable when the objects are part of a net.

Move Shortcut

To select and move any object or selection:

1. Position the cursor over the object to be moved.
2. Click and hold LEFT MOUSE.
The Status Bar will report what is being moved.
3. Move the object to the new location.

Drag Shortcut

To drag an object that is part of a net:

1. Position the cursor over the object to be dragged.
2. Click LEFT MOUSE to focus the object.

If the object is part of a net the entire net will highlight. If no other objects highlight then this object does not have the same net attribute as the joining objects and can not be dragged.

3. Click LEFT MOUSE on the object again.

If you click on one of the sizing handles of the focused object, you will drag that handle rather than the entire object. For more information on the behavior of a focused object refer to the *Focus* topic in the *Editing* section of this chapter.

4. Drag the object to the new location.

Dragging a Component

To drag a component:

1. Select the **Edit » Move » Drag** menu item (shortcut: M, D).
2. Click on the component you wish to drag and move it to the new location.

The behavior of the tracks that connect to and pass under the component are influenced by the Component Drag option in the Preferences dialog (**Tools » Preferences**). **Edit » Move » Drag** can also be used to drag any primitive object.

Selection Moves

Once selected, an individual item or a complex selection containing many items can be moved as a single entity. Select **Edit » Move** or press the M shortcut key to pop up the Move sub-menu. This sub-menu includes:

Move Selection

This option allows you to select a new location for the selection, which will move as a block (shortcut: M, L). When you invoke this process you will be prompted for a reference point.

Flip Selection

This option flips the selection around the vertical axis. Pressing the X or Y keys during a Move Selection can also be used to flip a selection.

Rotate Selection

Selecting this option will pop the Rotation Angle dialog. Enter the rotation angle, which has an angular resolution of .001 degrees. You will then be prompted to select a reference point, about which to pivot the rotation.

Selections can also be rotated during a Move Selection by pressing the SPACEBAR. The spacebar rotation angle is set in the Options Tab of the Preferences dialog.

For Gerber plot final artwork, the target photoplotter may not allow rotated primitives. Rotated rectangles (such as rectangular component pads) are automatically painted using a round aperture during Gerber generation.

Moving Individual Items

The other **Edit » Move** process launchers work with items that have not been previously selected. Because these Move process launchers manipulate specified items, it is easy to control the selection in a dense layout, where many items overlap.

Moving or deleting one or more items can leave a “hole” in the display under the moved/deleted primitives. This is because the PCB Editor does not continuously redraw the screen during moves or deletions, as this would significantly slow system performance. Click the Redraw button on the Tool bar, or press END to refresh the screen.

◆ Rotatable objects, such as components, pads and text strings, can be rotated in anti-clockwise steps during moves by pressing SPACEBAR, or in clockwise steps by pressing SHIFT+SPACEBAR. The Rotation Step is set in the Preferences dialog.

Break Track

Break Track converts a single track segment into two connected segments. To break a track:

1. Choose the **Edit » Move » Break Track** menu item (shortcut: M, B).

You will be prompted to “Choose a track”.

2. Position the cursor over the track segment and press ENTER or click LEFT MOUSE.

The track will be displayed in draft mode.

3. Move the cursor to drag the break point to a new location.
4. Press ENTER or click LEFT MOUSE again to complete the move.

You will be prompted “Choose a track” again. During the drag, you can abort the move by clicking RIGHT MOUSE or pressing ESC once. Note that the “Select track” prompt is still displayed.

5. Select another track or press ESC (or click RIGHT MOUSE) a second time to quit from the break track mode.

Polygon Vertices

The boundary of a polygon plane can be reshaped by moving the vertices. For information on moving the polygon vertices refer to the Refer to the *Polygons* topic in the *PCB Design Objects* chapter.

Deleting

Select the **Edit » Delete** menu item to remove objects from the PCB workspace. Delete differs from the Cut or Clear processes described previously. With Cut or Clear you identified the objects first (selected them), then picked the action (Cut or Clear). To Delete you pick the action first (Delete), then click on the object.

If more than one object is under the cursor when you click to delete, a pop up menu will appear allowing you to choose exactly which object to delete.

Delete is also independent of selection. For example, when deleting individual tracks, tracks that are part of the current selection will be left undisturbed.

All deletions can be restored by using **Edit » Undo** (or ALT+BACKSPACE). If you have deleted a series of items, they will be restored one-at-a-time starting with the last deleted item. **Edit » Redo** uses the same first-in/last-out logic. Redo reverses the Undo operations, one-at-a-time.

- ◆ Press the DELETE key to delete the focused object.
- ◆ Press the CTRL+DELETE keys to delete the current selection.

Measuring Distance in the PCB Workspace

Measure Distance

Use this feature when you need to perform an accurate measurement in the PCB workspace. After selecting the **Reports » Measure Distance** menu item you will be prompted to Select Measure Start Point. Position the cursor, click LEFT MOUSE, move the cursor to the end point and click LEFT MOUSE again. A dialog will report; the point-to-point distance measured, the X distance and the Y distance.

Tips on using the Measure Distance feature:

- Change the snap grid if you cannot accurately position the cursor at the required points (shortcut: G).
- You may need to temporarily disable the Electrical Grid if you find that the cursor snaps to the center of electrical objects (shortcut: SHIFT+E to toggle the electrical grid on and off).
- Toggle the workspace units to do a measurement in metric (shortcut: Q to toggle between imperial and metric).

Measure Primitives

As you are designing your PCB you will often want to know the clearance between two primitives. The Measure Primitives feature provides a quick and easy way of finding the shortest distance from the edge of one primitive to the edge of another primitive.

Select **Reports » Measure Primitives**, click on the first primitive, then click on the second primitive. A dialog will report; the type, coordinates and layer of each primitive, and the shortest distance between their closest edges.

PCB Editing Shortcuts

Re-entrant Editing

The PCB Editor includes a powerful feature which allows you to perform a second operation, without having to quit from the operation you are currently carrying out. This facility is known as re-entrant editing.

Re-entrant editing allows you to work more flexibly and intuitively. For example, you start placing a track, then realized that another track segment must be deleted. There is no need to drop out of Place Track mode. Simply press the Delete shortcut keys (E, D), delete the required track segment and click RIGHT MOUSE or ESC to terminate the Delete process. You are now back in the Place Track process, ready to place the new segment.

◆ Accessing another process while executing a process is only possible using the shortcut keys.

The number of times another process can be launched before the current process is complete depends on the demands each of these incomplete processes is placing on the software. For graphical type processes approximately ten processes can be nested. A dialog will pop up if the limit has been reached.

Canceling a Screen Redraw

Whenever you change the size and/or position of your view of the screen, the contents of the workspace will be redrawn to reflect the change. You can terminate the redrawing process by pressing the SPACEBAR anytime while the redraw is in progress. This saves time whenever you wish to immediately scroll or zoom again, without waiting for the redraw to complete. Use the solid (not transparent) layer display option to make redraws faster (Options Tab of the Preferences dialog). Working with the Display mode options set to Draft for all primitives will also slightly improve redraw speed (Show/Hide Tab of the Preferences dialog).

◆ Use the ALT+END shortcut keys to redraw the current layer only.

Mouse Shortcuts

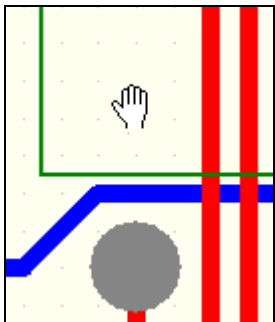
As you read through this Handbook, you will notice several mouse and keyboard shortcuts that are used to speed-up or simplify frequently performed operations. For example, pressing P, P allows you to place a pad without having to go to the Place menu and choose the Pad menu item. Using the left mouse button for ENTER and the right mouse button for ESC will allow you to perform many operations without using

the keyboard. The opposite can also be done, press ENTER or ESC on the keyboard rather than clicking OK or Cancel in a dialog. Sometimes keyboard actions provide the only practical way of performing an operation when you do not wish to move the mouse in the workspace, such as setting a new grid while placing an object, or changing the zoom level while moving a selection.

You can also create your own custom shortcut keys. Refer to the *Design Explorer* section for clues on creating your own shortcut keys.

- ◆ If you double-click on any placed item, the Change dialog for that item will be opened, allowing you to edit its attributes.
- ◆ To move an item simply click and hold LEFT MOUSE, hold on the object and drag the mouse to the new position.
- ◆ To delete a primitive from the design, click on the primitive you wish to delete to focus it, then press the DELETE key.

Slider Hand



Use the Slider hand to quickly change your view of the PCB

The PCB Editor includes a powerful mouse shortcut for changing your view of the workspace, the Slider Hand. Click-and-hold the right mouse button, and the cursor will change to a hand symbol. You can now slide your view of the PCB design around in the window.

Keyboard Shortcuts

There are two ways of creating shortcuts invoked through the keyboard. The first is through the Keyboard Shortcut Editor (**Client menu » Edit Shortcuts**). These are known as Keyboard Shortcuts. They launch a process directly. For example, pressing CTRL+G will pop up the Snap Grid dialog, allowing you to change the snap grid.

Processes can also be launched through the keyboard via the menu keyboard shortcuts. Underlined menu items which lead to a sub-menu will pop up that sub-menu, underlined menu items which do not pop up a sub-menu will launch the process tied to that menu item. For example, press P to pop up the Place menu, then press V to present the current via on the cursor, ready for placing. Press T to pop up the Tools menu, followed by R to pop up the Auto Route sub-menu.

♦ If the same keyboard key has been assigned as a keyboard shortcut and also as a menu shortcut the keyboard shortcut will take precedence.

Menu Shortcuts include:

A	Auto Route menu
B	View » Toolbars sub-menu
D	Design menu
E	Edit menu
F	File menu
G	Snap Grid pop-up menu
H	Help menu
I	Interactive placement options sub-menu
J	Edit » Jump sub-menu
M	Edit » Move sub-menu
N	Netlist pop-up menu
O	Options pop-up menu
P	Place menu
R	Reports menu
S	Edit » Select sub-menu
T	Tools menu
U	Tools » Unroute sub-menu
V	View menu
W	Window menu
X	Edit » DeSelect menu
Z	Zoom pop-up menu

Keyboard Shortcuts include:

L	Layers Tab of Document Option dialog
Q	Toggle Units
CTRL+D	Display Tab of the Preferences dialog
CTRL+G	Snap Grid dialog
CTRL+H	Edit » Select » Connected Copper
CTRL+L	Layers Tab of Document Option dialog
CTRL+M	Measure Distance
CTRL+O	Options Tab of the Preferences dialog
PGUP	View » Zoom In
PGDN	View » Zoom Out
CTRL+PGUP/PGDN	Zoom maximum / minimum
SHIFT+PGUP/PGDN	Zoom at 0.1 zoom step rate
HOME	View » Pan
END	View » Refresh
CTRL+HOME	Jump Absolute Origin
CTRL+END	Jump Current origin
CTRL+INS	Edit » Copy
CTRL+DEL	Edit » Clear
SHIFT+INS	Edit » Paste
SHIFT+DEL	Edit » Cut
ALT+BACKSPACE	Undo
CTRL+BACKSPACE	Redo
SHIFT+E	Toggle Electrical grid on/off
SHIFT+S	Toggle single layer mode on/off
SHIFT+R	Cycle through the Routing Modes
SHIFT+F4	Cascade Windows
SHIFT+F5	Tile Windows
*	Toggle active signal layers
+ or -	Next / previous active layer
F1	Help Index
UP, DOWN	Move one snap grid point, vertically
SHIFT+UP, DOWN	Move 10 snap grid points, vertically
LEFT, RIGHT	Move one snap grid point, horizontally
SHIFT+LEFT, RIGHT	Move 10 snap grid points, horizontally

Special Mode-Dependent Keys

TAB	Opens a dialog for the object currently being placed. Allows you to edit attributes when placing any object. Use the TAB key to “edit on-the-fly”.
SPACEBAR	Toggles between Start and End track placement modes; rotate item anti-clockwise during move (set step value in Preferences dialog); abort screen redraw; change the MiniViewer magnification level.
SHIFT	Control acceleration during Autopan (set mode in Preferences dialog).
SHIFT+SPACEBAR	Toggle track placement modes; rotate item clockwise during move.
N	Hide the connection lines while moving a component.
CTRL	Temporarily suspend the Electrical grid while editing.
CTRL+SPACEBAR	Cycle through each connection line on a pad during routing.

Locating Components

Often you will know what component you wish to edit or move, but can not currently see it on the screen. For example, you might want to place a particular component where you are currently working but do not want to scroll or zoom to find it. Select the **Edit » Move » Move Component** menu item (shortcut: M, C). Click somewhere in the workspace where there is no component under the cursor and the Component Designator dialog will pop up. If you know the designator type it in and click OK. If the component is off screen the view will start to scroll so move the mouse to bring the cursor and component into view.

When the Component Designator dialog pops up you can also leave the ? and click OK. This will pop up the Components Placed dialog, allowing you to select any component that is in the workspace.

Undo and Redo

The PCB Editor includes a full multi-level Undo and Redo facility. Each procedure is stored in a stack-like arrangement. When Undo is chosen, the last operation is undone. Choosing Undo again will undo the next-to-last operation, and so on.

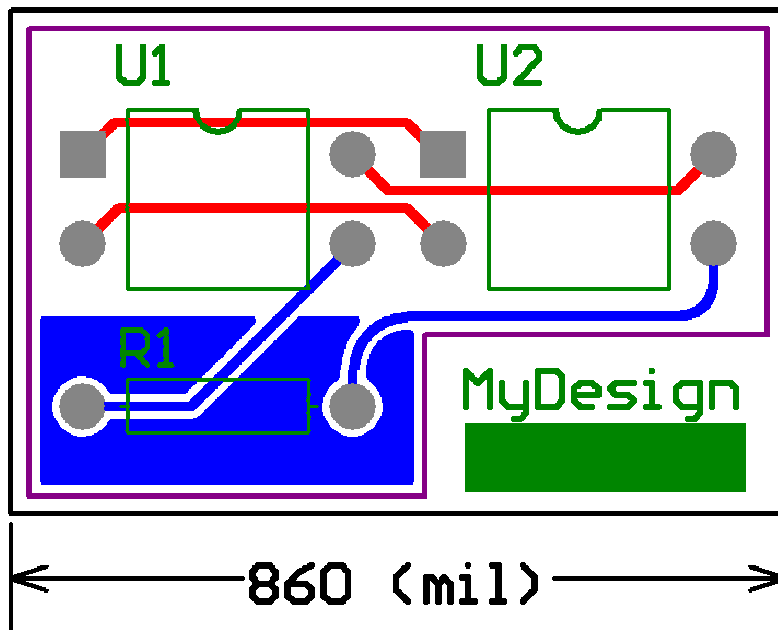
The Redo facility will reverse a previous Undo. Every time you undo something, the operation is stored in memory, in a stack-like arrangement. If you then select Redo the last undo operation that you did will be reversed, then the next-to-last, and so on.

Check in the Edit menu to confirm exactly what action will take place when you select Undo or Redo. While Protel 99 SE's PCB Editor has been optimized to ensure that Undo and Redo operations are not memory intensive, you can clear the Undo stack by temporarily setting the Stack Size to zero in the Preferences dialog.

PCB Design Objects

Protel 99 SE's PCB layout environment, whether creating and editing PCBs, or working in the library editor, consists of two basic features: *objects* that are placed in the workspace to build up the design, and *processes*, which are used by the system or the user to create, modify, save and report on the objects.

There are two types of objects in the PCB Editor, *Primitive Objects* and *Group Objects*. Primitive objects are the most basic elements, and include; tracks, pads, vias, fills, arcs and strings. Anything that is made up of primitives and identified as a design object is a group object. Examples of group objects include; components, dimensions, coordinates and polygons.



Your design is built up from the set of PCB design objects

PCB Primitive Objects

Tracks

Tracks are the basic PCB line element. Tracks can be placed on any layer, with any width from 0.001 to 10000 mils wide. Tracks can be used to route connections on signal layers, to define the board outline on a mechanical layer, to create a component outline on the silkscreen layer, or to define “no-go” regions on the keepout layer. Use tracks whenever you need to define a straight line in the PCB workspace.

◆ If you are placing tracks on a non-electrical layer, select **Place » Line** from the menus instead of **Place » Interactive Routing**. You will still be placing track segments, the difference is that the Line parameters are constrained by the Default Primitives (Preferences dialog), whereas the Track parameters are constrained by the Design Rules.

Activity - Placing Tracks

A track is a layer-dependent object. Click on the required Layer Tab at the bottom of the PCB workspace before you start placing tracks. To place track segments on the current layer:

1. Select **Place » Interactive Routing** from the menus (shortcut: P, T or click the Track button on the PlacementTools toolbar).

The prompt “Choose start location” is displayed on the Status Bar.

◆ When you are already in track placement mode you can:

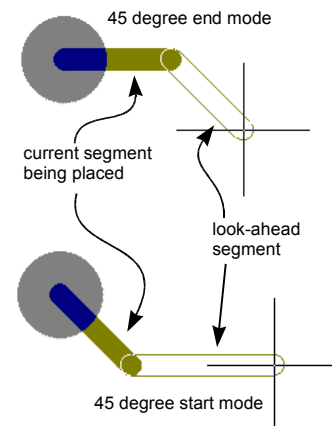
- Press * on the numeric keypad to toggle *signal* layers
- Press the + or – keys on the numeric keypad to toggle all layers

2. Click LEFT MOUSE (or press ENTER) once to define a start point for the track.

The Status Bar will display the net assigned to this track, the current segment length and the total track length in brackets.

3. Move the cursor in any direction. As you do you will notice two track segments, one solid track and one outline track. The solid track is the segment you are currently placing, the outline segment is the “look-ahead” segment. Refer to the chapter, *Manually Routing the PCB*, for more information on how to use the look-ahead feature.

By default you will be in the *45 Degree End* track placement mode. In this mode the first segment you place will be vertical or horizontal, the second at 45 degrees. If you press the SPACEBAR the mode



will toggle to *45 Degree Start*, where the current segment is diagonal and the look-ahead is vertical or horizontal. Press the SPACEBAR a second time to revert to the 45 Degree End mode.

4. Move the cursor until the end of the solid track segment is where you would like it, and click LEFT MOUSE (or press ENTER) to place this first segment.
5. Move the cursor to continue with a new track segment, which will extend from the placed track segment. Experiment with the behavior by moving the cursor and pressing the SPACEBAR. Click LEFT MOUSE each time you wish to define a track segment.
6. Click RIGHT MOUSE to end this series of connected track segments.

◆ If you make a mistake press BACKSPACE to remove the last track segment.

Note that “Choose start location” is still displayed on the Status Bar. This allows you to end one series of connected tracks segments and begin a new series of track segments elsewhere in the workspace without having to choose **Place » Interactive Routing** again.

7. To exit track placement, press ESC or RIGHT MOUSE a second time.

◆ To edit the track attributes during placement press the TAB key to pop up the Track dialog. If the Track is part of a net, you can only change the width between the Min and Max values specified in the Width Constraint design rule that applies to that net.

Track Placement Modes

The PCB Editor provides six *track placement modes*. The mode specifies how the corner is created when placing tracks.

The track placement modes include:

45 Degree (Start and End)

Constrains track placement to a 45 degree segment and a horizontal/vertical segment.

45 Degree with Arc (Start and End)

Constrains track placement to a 45 degree segment and a horizontal/vertical segment, joined by a 45 degree arc.

90 Degree (Start and End)

Constrains track placement to a horizontal segment and a vertical segment.

90 Degree with Arc (Start and End)

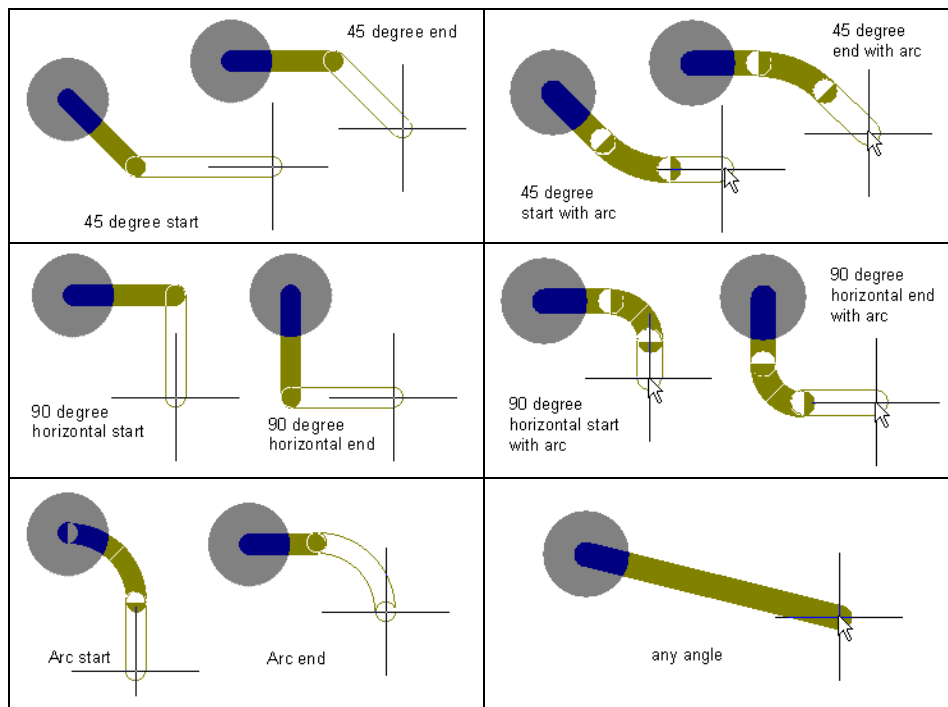
Constrains track placement to a horizontal and vertical segment, joined by a 45 degree arc.

Any Angle

Allows track segment to be placed at any angle.

Arc Start and Arc End

Constrains track placement to a 90 degree arc and a horizontal/vertical segment.



The six track placement modes

- ◆ Press the SPACEBAR to toggle between the Start and End placement modes while placing a track. Hold the SHIFT key and press the SPACEBAR to toggle through the six different types of track placement modes.
- ◆ Press the “.” shortcut key to increase the arc size in the “with arc” modes, press the “,” key to decrease the arc size.

Placing Tracks to Route a Connection

When you place a track that starts on an object with a net name, such as a pad, you are *routing* that net. The track you are placing will adopt the net name of the pad and the design rules that apply to that net will be observed. Placing tracks to route a net is supported by a number of features, such as the track placement modes, that simplify this task. For a complete discussion of these refer to the chapter, *Manually Routing the PCB*.

Changing Tracks

 Click on the What's This Help icon at the top of the dialog for information on each attribute in the Track dialog.

Tracks can be changed both individually and globally. Select the **Edit » Change** menu item to edit an existing track. Click on a track to pop up the Change Track dialog, where you can change the attributes.

Default Track

The attributes of the track currently being placed can be changed during placement by pressing the TAB Key. This will pop up the Track dialog, where you can edit the track attributes. The default track attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option in the Defaults Tab is not set, changes made during placement become the new defaults.

Note: if the track is part of a net, you can only change the width between the Min and Max values specified in the Width Constraint design rule that applies to that net.

Pads

Pads can be either multi-layer or placed on any individual layer. For example, surface mount components and edge connectors have single layer pads on the Top and/or Bottom layers. Pads shapes can be circular, rectangular, rounded rectangular (circular with different X and Y sizes), or octagonal with X and Y size definable from 1 to 10000 mils. Hole size can range from 0 (SMD) to 1000 mils. Pads can be identified with a designator up to 4 characters long.

On a multi-layer pad the Top layer, Mid layers and Bottom layer pad shape and size can be independently assigned to define a pad stack. Pads can be used individually as free pads, or they can be incorporated with other primitives into components.

Placing Pads

Free pads (pads that are not grouped in a library component) can be placed anywhere in your design. Through-hole pads (and vias) are multi-layer objects which occupy each signal layer of the PCB and can be placed without regard to the current layer setting. Single layer pads can be placed on any layer.

To place a pad select **Place » Pad** from the menus (shortcut: P, P or click the Pad button on the PlacementTools toolbar).

◆ Press TAB to change the default pad attributes during placement.

Pad Designator

Pads can be labeled with a designator (usually representing a component pin number) of up to four alphanumeric characters. Spaces are not allowed but the designator can be left blank if desired.

Pad designators will auto-increment by 1 during placement if the initial pad has a numeric designator. To set the designator prior to placing the first pad, press the TAB key while the pad is floating on the cursor.

To achieve alpha or numeric increments other than 1, use the Paste Array feature. By setting the designator of the pad prior to copying it to the clipboard and setting the Text

Increment field in the Paste Array dialog, the following types of pad designator sequences can be placed;

numeric (1, 3, 5)

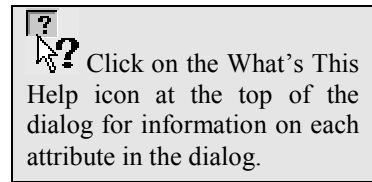
alphabetic (A, B, C)

combination of alpha and numeric (A1, A2, or 1A, 1B, or A1, B1 or 1A, 2A, etc).

To increment numerically set the Text Increment field to the amount you wish to increment by. To increment alphabetically set the Text Increment field to the letter in the alphabet that represents the number of letters you wish to skip. For example, if the initial pad had a designator of 1A and the Text Increment field was set to C (the third letter of the alphabet), the pads would have the designators 1A, 1D (three letters after A), 1G (three letters after D), and so on.

Changing Pads

Both free pads and component pads can be individually and globally edited. To edit a pad select the **Edit » Change** menu item. The Status Bar will prompt “Change Any Object”. Click on a pad to pop up the dialog, where you can edit the pad attributes.



Default Pad

The attributes of the pad currently being placed can be set by pressing the TAB Key as soon as you select the **Place » Pad** menu item. The default pad attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Vias

When tracks from two layers need to be connected vias are placed to carry a signal from one layer to the other. Vias are like round pads, which are drilled and usually through-plated when the board is fabricated.

Vias are either multi-layer, blind or buried, and can be any diameter from 2 to 10000 mils wide. Vias can be placed manually using the Via button on the PlacementTools toolbar or by selecting the **Place » Via** menu item; they can be placed automatically by the auto via feature when placing tracks, or by the autorouter. The hole size can be set from 0 to 1000 mils.

Via Type

Vias can be multi-layer, blind or buried. A multi-layer via passes from the Top layer to the Bottom layer and allows connections to all internal signal layers. A blind via connects from the surface of the board to an internal layer, a buried via connects from one internal layer to another internal layer.

Blind and Buried Vias

Before using blind or buried vias it is important to establish the level of support provided by the manufacturer. Most manufacturers support blind and buried vias between what are termed *layer pairs*. Using this technology, a multi-layer board is fabricated as a set of thin double-sided boards which are then “sandwiched” together. This allows blind and buried vias to connect between the surfaces of these thin double-sided boards, which become the layer pairs. The layer pairs are defined by the layer stack you configure in the Layer Stack Manager dialog. It is important to note that the layers pairs are dependant on the layer stackup style – read the chapter *Defining the Board*, and contact your manufacturer to ensure you select the correct stackup style, before you start designing with blind and buried vias.

Once you have established the correct stackup style, you should define the valid drill pairs. Drill pairs are set up in the Drill Pair Manager dialog, click on the Drill Pair button in the Layer Stack Manager dialog to open the Drill Pair Manager dialog. If you define a drill pair for each layer pair in your design the PCB Editor will automatically insert the correct via type (thru-hole, blind, or buried) as you toggle layers during routing.

Default Via

The attributes of the via currently being placed can be set by pressing the TAB Key as soon as you select the **Place » Via** menu item. The default vias attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Placing Vias

Vias can be placed by the Autorouter, by the Auto Via feature, or manually.

Autorouter Vias

The autorouter will obey the via defined by the Routing Via Style design rule which has a scope set to board (select **Design » Rules**). Refer to the chapter, *Specifying the PCB Design Requirements*, for more information on design rules.

Auto Via Feature

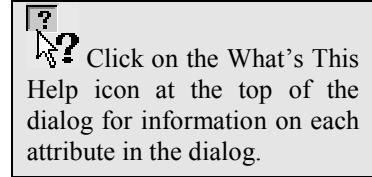
If the * key is used to toggle to another signal layer when a track is being placed, a via is added automatically. This via will obey the appropriate Routing Via Style design rule and the drill pair definitions. The routing via style parameters can be changed during routing by pressing the TAB key while routing.

Manually Placed Vias

To manually place a via select the **Place » Via** menu item. The current default via will appear on the cursor. Click to place a via in the workspace.

Changing Vias

To edit a via, select the **Edit » Change** menu item and click on the via. The Change Via dialog will pop up, where you can change the via attributes.



Connecting Vias to Power Planes

Like pads, vias automatically connect to a internal power plane layer that is assigned the same net name. The via will connect in accordance with the applicable Power Plane Connect Style design rule.

If you do not want vias to connect to power planes, add another Power Plane Connect Style design rule with a connection style of No Connect. Refer to the topic *Examples of Using Design Rules* in the *Specifying the PCB Design Requirements* chapter for an example of how to set this rule up.

Fills

Fills (or *area fills*) are rectangles which can be placed on any layer. When placed on a signal layer they become areas of solid copper and can be used to provide shielding or to carry large currents. Fills of varying size can be combined to cover irregularly shaped areas. They can be combined with track or arc segments and be recognized as electrically connected when running the design rule check (DRC) feature.

Fills can also be placed on non-electrical layers. For example, place a fill on the Keep Out layer to designate a “no-go” area for both autorouting and auto component placement. Place a fill on a Power Plane, Solder Mask, or Paste Mask layer to create a void on that layer.

Activity - Placing a Fill

1. To Place a fill, select the **Place » Fill** menu item.
The Status Bar will prompt “Select First Corner”.
2. Position the cursor and click LEFT MOUSE.
3. Move the cursor to position the diagonally opposite corner.
4. Click LEFT MOUSE to define the second corner.
5. Continue placing fills, or click RIGHT MOUSE to stop.

◆ A fill will “adopt” a net name if the first corner is placed on an object which has a net name.

Changing Fills

To edit a fill, select the **Edit » Change** menu item and click on the fill. The Fill dialog will pop up, where you can edit the fill attributes.

Default Fill

The attributes of the fill currently being placed can be set by pressing the TAB key as soon as you select the **Place » Fill** menu item. The default fill attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Arcs

Arcs are essentially circular track segments. They can be placed on any layer with a radius between 0.001 to 16000 mil and width from 0.001 to 10000 mils wide. The angular resolution is 0.001 degrees. Arcs can be placed using the Arc button on the PlacementTools toolbar, the **Place » Arc** menu items, or as part of a track using the Place Track process. Arcs are also used when generating polygon fills.

Arcs have a variety of uses in PCB layout. For example, they can be used to indicate component shapes on the Overlay layers, or on a mechanical layer to indicate the board outline, cut outs, and so on. Arcs can be open, or closed to create a circle.

♦ Arcs can also be placed on signal layers as part of a track. These arcs can be generated on-the-fly while placing tracks if the Track Placement Mode is set to one of the modes that includes arcs. Press SHIFT+SPACEBAR during track placement to toggle through the placement modes. Once you have selected the desired arc mode, press the SPACEBAR to toggle between the Start and End modes.

Activity - Place Arc (starting at the center)

To place an arc on the current layer using the arc center as the starting point:

1. Select **Place » Arc** (Center) (shortcut: P, A) or the Arc tool button.

The prompt “Select Arc Center” is displayed on the Status Bar. You can change layers at any time during this operation by pressing * (to toggle active signal layers); + or – (to toggle up and down through all active layers).

2. Position the cursor to set the center of the arc and click LEFT MOUSE. As you move the mouse a highlighted arc will be displayed.
3. Position the cursor to set the *radius* and click LEFT MOUSE.
4. Position the cursor to define the *start point* of the arc and click LEFT MOUSE.
5. Position the cursor to define the *end point* of the arc and click LEFT MOUSE.

♦ To render the arc in the other direction, press the SPACEBAR before defining the end point.

If you are drawing a 360 degree arc, click to define the Start and End points without moving the cursor.

6. Start a new arc, or press ESC or RIGHT MOUSE to stop placing arcs.

Activity - Place Arc (starting at the edge)

To place an arc on the current layer starting at one of the arc end points:

1. Select **Place » Arc** (Edge) (shortcut: P, E).
2. Position the cursor to set the *start point* of the arc and click LEFT MOUSE once.

3. Position the cursor to set the *end point* of the arc and click LEFT MOUSE.
4. Start a new arc, or press ESC or RIGHT MOUSE to stop placing arcs.

Activity - Place Arc of any angle (starting at the edge)

You can also place an arc starting at an end point, of any angle:

1. Select **Place » Arc (Any Angle)** (shortcut: P, N).
2. Position the cursor to set the *start point* of the arc and click LEFT MOUSE once.
3. Position the cursor to set the *radius* and the *center point* of the arc and click LEFT MOUSE.
4. Position the cursor to set the *end point* of the arc and click LEFT MOUSE.
5. Start a new arc, or press ESC or RIGHT MOUSE to stop placing arcs.

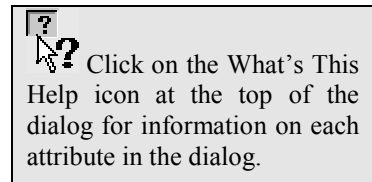
Activity - Place Circle

To place a full circle (closed arc):

1. Select **Place » Full Circle** (shortcut: P, U).
2. Position the cursor to set the *center point* of the circle and click LEFT MOUSE once.
3. Position the cursor to set the *radius* of the circle and click LEFT MOUSE once.
4. Start a new circle, or press ESC or RIGHT MOUSE to stop placing circles.

Changing Arcs

To edit an arc, select the **Edit » Change** menu item and click on the arc. The Arc dialog will pop up, where you can edit the arc attributes.



Default Arc

The attributes of the arc currently being placed can be set by pressing the TAB key as soon as you select one of the **Place » Arc** menu items. The default arc attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Strings

Text strings of up to 254 characters (including spaces) can be placed on any layer with any height from 0.010 to 10000 mils. Strings can be placed with the Text String button, or by the selecting the **Place » String** menu item.

Text is rendered using one of three special fonts. The Default style is a simple vector font which supports pen plotting and vector photoplotting. The Sans Serif and Serif fonts are more complex – and will slow down vector output generation, such as Gerber. These fonts are built into the software and cannot be changed. All fonts have the full

IBM extended ASCII character set that supports English and other European languages.

All text strings (component designators, component comments, and free text strings) have the same attributes, and can be moved and edited in the same way. Free text can be placed on any layer. Component text is automatically assigned to the Top or Bottom Overlay layer when the component is placed, but can be moved to any layer.

Free Text strings can be moved or edited like other primitives. Component text can be moved independently of the component (**Edit » Move**). If the component is moved, component text will move relative to the component.

The PCB Editor includes “Special Strings”. These are strings which are interpreted when output is generated. Special strings are discussed below.

Default String

The attributes of the string currently being placed can be set by pressing the TAB key as soon as you select the **Place » String** menu item. The default string attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Activity - Placing a String

To place a free string:

1. Select the **Place » String** menu item. The current default string will appear floating on the cursor.
2. Press the TAB key to pop up the Change String dialog.
3. Type the string into the Text field or select one of the special strings from the drop down list.
4. Set the Height, Width and Font as required.
5. Click OK. The string will appear floating on the cursor. Click to place the string.
6. Click RIGHT MOUSE or ESC to stop placing strings.

◆ While placing a string press the X or Y keys to mirror *along* these axes, press the SPACEBAR to rotate the string.

Changing Strings

Free text strings and component text (designators and comments) can be changed both individually and globally. Double-click on a string to pop up the string dialog, where the string attributes can be edited.

 Click on the What's This Help icon at the top of the dialog for information on each attribute in the dialog.

Special Strings

Special strings allow you to place generic, non-specific text which is interpreted when printing, plotting or generating Gerber files. For example, the string .PRINT_DATE will be replaced by the current date when output is generated. The available special strings are:

.PRINT_DATE
.PRINT_TIME
.PRINT_SCALE
.LAYER_NAME
.PCB_FILE_NAME
.PCB_FILE_NAME_NO_PATH
.PLOT_FILE_NAME
.ARC_COUNT
.COMPONENT_COUNT
.FILL_COUNT
.HOLE_COUNT
.NET_COUNT
.PAD_COUNT
.STRING_COUNT
.TRACK_COUNT
.VIA_COUNT
.DESIGNATOR
.COMMENT
.LEGEND
.NET_NAMES_ON_LAYER

◆ Add the .DESIGNATOR and .COMMENT special strings to the component in the library. Use these if you need to pre-define the location and layer of these strings for the component. The standard designator and comment can be hidden if desired.

◆ Place the .LEGEND string on the Drill Drawing layer. It will be replaced by a drill table when the output is generated.

◆ To interpret special strings on screen enable the Convert Special Strings option in the Display Tab of the Preferences dialog. Note that not all special strings can be interpreted on screen.

Group Objects

A group object is any set of primitives which has been defined to behave as an object. These may be user defined, such as components and polygons, or system defined, such as coordinates and dimensions. A group can be manipulated as one object – they can be placed, selected, copied, changed, moved and deleted.

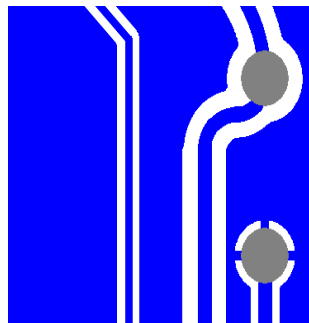
◆ **Select Tools » Convert » Explode Polygon to Free Primitives** to convert a polygon into a set of tracks and arcs.

Polygons

Polygons are special areas of copper formed when you use the **Place » Polygon Plane** process. Polygon planes (or copper pours) can fill irregularly shaped areas of a board and can connect to a specified net as they are poured.

Although polygons consist of tracks and arcs, polygons can be manipulated as a unit. Polygon boundaries can be re-shaped and polygons can be re-poured around new obstacles after placement, and any of their attributes, such as grid and track size, can be changed. By adjusting the grid and track size, a polygon plane can be either solid (copper) areas or a cross-hatched “lattice”.

When placed in occupied board space polygon planes “pour” copper around any tracks, pads, vias, fills or text while maintaining the clearances specified in the design rules. If you are working with a netlist-based layout the plane can automatically connect to any component pads on the specified net that are within the polygon plane.



Polygons pour around existing objects, in accordance with the design rules

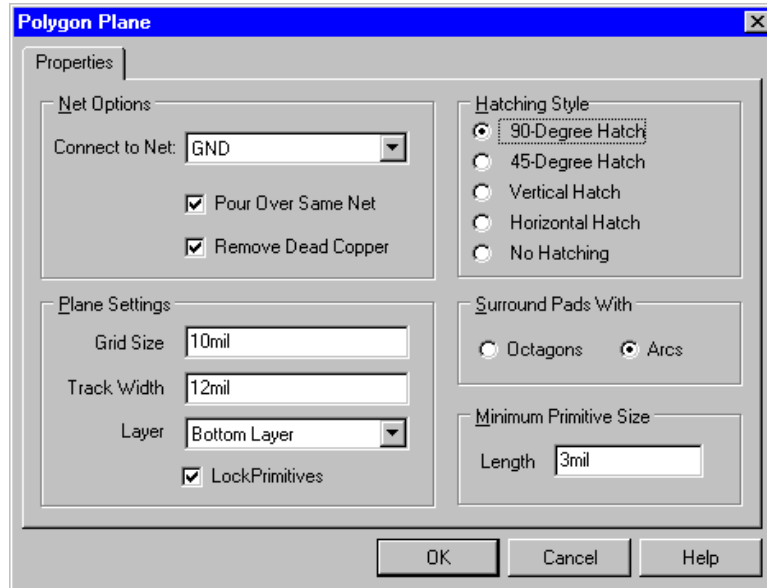
Polygons can be poured to create a multi-sided shape on any layer. If a polygon is placed on a non-signal layer it will not be poured around existing objects as these objects are not assigned to a net and therefore do not “belong” to anything.

Activity - Placing a Polygon Plane

To place a polygon plane:

1. Choose **Place » Polygon plane** (shortcut: P, G).
The Place Polygon Plane dialog will pop up. Set the attributes as required. Each attribute is defined below.
2. Click to define the starting point of the polygon.
3. Continue to click at each vertex of the polygon until the boundary of the polygon plane is defined. Press the SPACEBAR to change polygon boundary track placement modes.

The polygon will pour when it is closed. If you do not actually close the polygon, when you press ESC or RIGHT MOUSE the polygon will automatically be closed, from the last vertex to the initial vertex.



Options in the Place Polygon Plane dialog include:

Net Options

Connect To Net

If a netlist has been loaded, one of the nets in the netlist can be selected in the Connect To Net drop down. If the polygon is being connected to a net, the other two Net Options can be applied.

Pour Over Same Net

If the Pour Over Same Net option is enabled any existing tracks within the polygon which are part of the net being connected to will be covered by the polygon.

Remove Dead Copper

Dead copper is copper placed by the Place Polygon Plane process which can not be connected to the selected net. Regions of dead copper are created when existing tracks, pads and vias prevent the plane pouring as one continuous area. These can be removed if desired. If this option is enabled and the polygon does not enclose a pin on the selected net, the entire polygon is removed as it is all dead copper.

Plane Settings

Grid Size

This is the grid on which the tracks within the polygon are placed. To allow the most effective placement of the polygon tracks this will ideally be a fraction of the component pin pitch.

Track Width

This is the width of the tracks which are placed to form the polygon. If the track width is smaller than the grid size the polygon will be hatched. If the track size is equal to or greater than the grid size the polygon will be solid. For a solid plane set the track width slightly larger than the grid size.

Layer

This is the layer the polygon is to be placed on. Polygons can be placed on both copper and non-copper layers.

Hatching Style

90 Degree Hatch

Fill the polygon with tracks running both horizontally and vertically.

45 Degree Hatch

Fill the polygon with tracks running at 45 degrees (in both directions).

Vertical Hatch

Fill the polygon with tracks running vertically.

Horizontal Hatch

Fill the polygon with tracks running horizontally.

No Hatching

Define all the outlines of the polygon but do not place any tracks inside the polygon. Use this option if you want to place the polygon, but do not want it to slow system performance. It can be re-poured later with the desired hatching.

Surround Pads With

Pads can be surrounded with either Arcs or Octagons. Octagons give smaller Gerber files and faster photoplotting.

Minimum Primitive Size

Length

The length field allows you to limit the minimum size of primitives used in the polygon. When polygons are poured they can contain many short pieces of tracks and arcs, placed to create smooth shapes around the existing objects on the board. By limiting the length of primitives used you will get faster pour times, screen redraws and output generation. This will be at the expense of the smoothness of the polygon edges.

How the Polygon will Connect to Pads

To control how a polygon connects to pads when the Connect To Net option is used, include a Polygon Connect Style design rule (select **Design » Rules**). This rule allows you to select between a direct connection and a thermal relief connection. It also allows you to set the conductor width and connection angle if you select relief connection. Refer to the chapter, *Specifying the PCB Design Requirements*, for more information on the Polygon Connect Style design rule.

Activity - Re-pouring a Polygon

To re-pour a polygon select the **Edit » Change** menu item. Click on the polygon you wish to re-pour and the Place Polygon Plane dialog will pop up (shortcut: double-click on the polygon). Change the attributes as desired and click OK. You will be asked if you wish to re-pour the modified polygon. Click Yes to re-pour the polygon with the new settings.

Activity - Changing the Shape of the Polygon Boundary

The boundary of a polygon plane can be reshaped by moving the vertices, deleting the vertices, and breaking the boundary tracks.

To modify the shape of the polygon:

1. Select the **Edit » Move » Polygon Vertices** menu item.

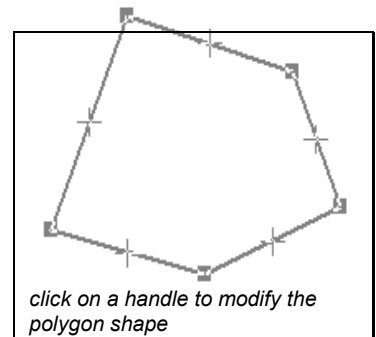
The Status Bar will prompt “Choose a Polygon”.

2. Click on the polygon to be edited.

The internal polygon tracks will disappear, replaced by the initial boundary that you defined for the polygon.

3. Each track segment that defines the boundary will have 3 handles, 2 small squares at each end, and a cross in the middle. Click on a corner handle to reposition the corner, click on the center handle to break the polygon boundary track.
4. Press DELETE to remove the vertex you are currently moving.
5. Click RIGHT MOUSE or press ESC when you have finished modifying the polygon.

After modifying the polygon you will be asked if you wish to re-pour the modified polygon. Click Yes to re-pour the polygon with the new settings.



Dimensions

Dimensions are special entities consisting of text and track segments. They are automatically generated when you indicate the starting and ending points after choosing the **Place » Dimension** menu item. Imperial or metric units will be calculated, depending upon the current Snap grid setting.

Activity - Placing a Dimension

1. Select the **Place » Dimension** menu item.
2. Click to define the start point.

The Status Bar will prompt “Select Measure End Point”.

3. Click to define the end point.

The dimensioning information will then be placed.

◆ To modify a placed dimension select **Tools » Convert » Explode Dimension to Free Primitives** to convert a dimension into a set of tracks and strings.

Changing Dimensions

Dimension attributes such as string height, font, etc, can be changed during placement (press the TAB key), or after the dimension has been placed. Select the **Edit » Change** menu item and click on the dimension to pop up the Change Dimension dialog, where the attributes can be edited.

Moving a Dimension

Dimensions can be moved, or “adjusted”, after they have been placed. To move or adjust a dimension, click once anywhere on the dimension to bring it into focus (the square focus handles will appear). A second click anywhere on the dimension moves the dimension, a second click on a focus handle allows you to adjust the dimension. If you re-size a dimension the distance will be automatically updated.

Default Dimension

The attributes of the dimension currently being placed can be set by pressing the TAB key as soon as you select the **Place » Dimension** menu item. The default dimension attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

Coordinates

Use a coordinate marker to indicate the coordinates of a specific point in the workspace. A coordinate marker includes a point marker (cross made of two tracks) and the X, Y coordinates of the position. They can be placed on any layer.

Default Coordinate

The attributes of the coordinate currently being placed can be set by pressing the TAB key as soon as you select the **Place » Coordinate** menu item, or press the button on the PlacementTools toolbar. The default coordinate attributes are set in the Defaults Tab of the Preferences dialog. If the “Permanent” option is not set in the Defaults Tab of the Preferences dialog then changes made during placement will become the new defaults.

◆ To move the coordinate string without moving the marker select **Tools » Convert » Explode Coordinate to Free Primitives** to convert the coordinate into a pair of tracks and a string.

PCB Component Footprints and Libraries

Comprehensive libraries of over 1600 predefined through-hole and SMD component footprints are included with Protel 99 SE. PCB footprints are created and modified in the PCB Library Editor. Refer to *The PCB Library Editor* topic later in this chapter, for information on using the PCB Library Editor to create component footprints and libraries.

What is a Footprint and what is a Component?

In the Protel 99 SE's PCB layout environment, a "footprint" is what exists in the PCB library. When this footprint is placed from the library into the PCB workspace, it is assigned a designator and comment. It is then referred to as a component.

Where are the PCB Footprint Libraries?

In Protel 99 SE, the PCB footprint libraries are stored within a set of Library Design Databases. The Protel 99 SE PCB Library Databases are in the \Program Files \Design Explorer 99 SE\Library\Pcb folder.

◆ In Protel 99 SE you can still access the PCB footprints in the older .LIB format libraries. These libraries can be added to the list of available libraries in the normal way.

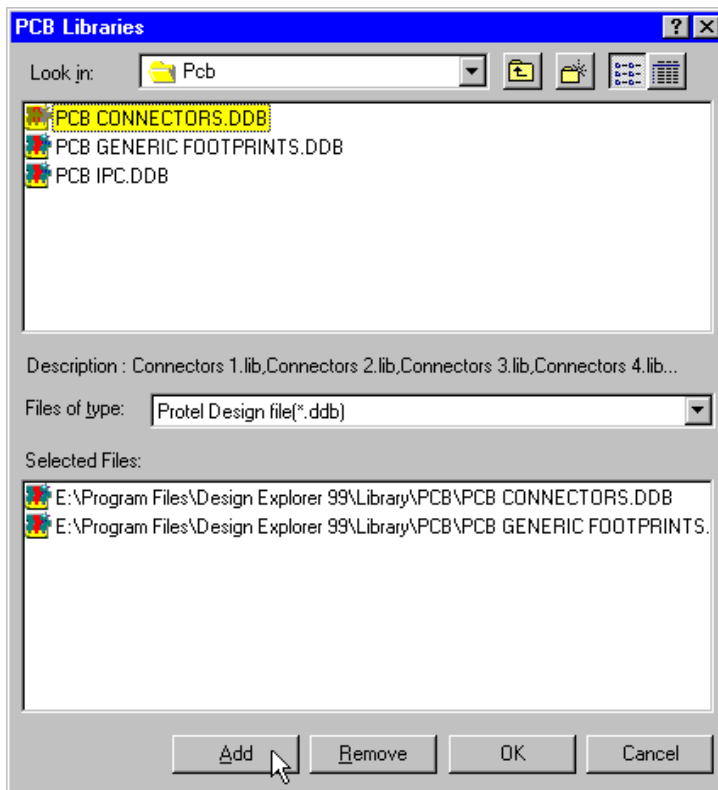
Accessing Component Footprints

To access the component footprints in libraries, the libraries must first be added to the *current library list* in the PCB Editor. Libraries are added and removed by selecting **Add/Remove Library** in the **Design** menu, or by pressing the Add/Remove button in the PCB Editor Panel (when the Browse mode is set to Libraries). This pops up the PCB Libraries dialog, where new libraries can be added, and open ones removed from the Selected Files list.

Once a library has been added, footprints from that library can be placed in the workspace. The only limit to the number of libraries that can be added is the memory available in your computer.

Adding and Removing Libraries

Use the PCB Libraries dialog to access components in the libraries. The Selected Files window at the bottom of the dialog lists all the currently added libraries.



After locating the required Library Database, double-click on it to add it to the list. Double-click on a Library Database in the Selected Files list to remove it.

Use the Look in field at the top of the dialog to browse to the folder where the Library Databases are located. The Protel 99 SE PCB Library Databases are stored in the \Program Files\Design Explorer 99 SE\Library\Pcb folder.

- ◆ Libraries are stored in standard Protel 99 SE Design Databases, which means you can easily create your own Library Databases. These databases can even be a mix of schematic and PCB libraries.
- ◆ To access components in libraries that are stored inside a Project Design Database, simply add the Project Design Database to the Selected Files list in the PCB Libraries dialog.

Finding and Placing Components

To browse the components in a library, set the Browse mode in the PCB Editor Panel to Libraries and click on the desired library in the list. When you click on a component in the Components list it will be displayed in the MiniViewer.

Placing in the PCB Editor

To place a component footprint:

1. Set the Panel Browse mode to Libraries.
2. Select the desired library from the list.
3. Select the component from the list in the Panel and press the Place button (shortcut: double-click LEFT MOUSE).

The Component will appear floating on the cursor.

4. Press the TAB key to edit the designator and comment prior to placing the component, then click OK to return to the component floating on the cursor.

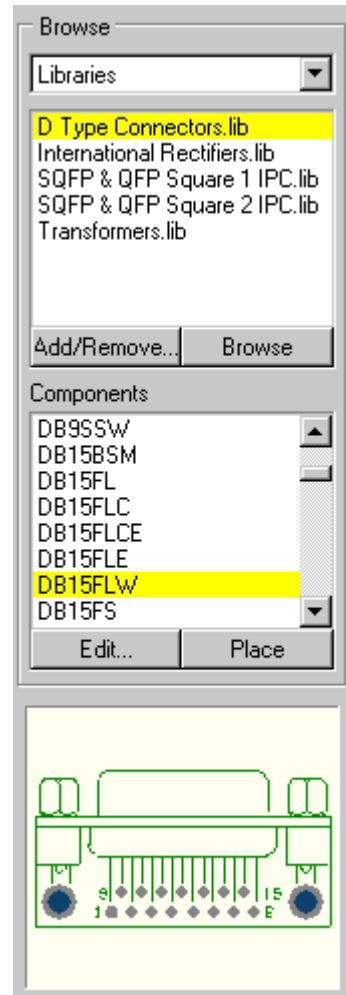
Press PAGEUP to zoom in. Press the SPACEBAR to rotate and the L shortcut key to flip the component to the bottom of the board. Use the Jump shortcut keys to jump to an exact location.

5. Click LEFT MOUSE to place the component.

Placing from the PCB Library Editor

Components can also be placed from the PCB Library Editor. The Library Editor Panel has a Place button. Select the component from the list and press Place.

The component will be placed in the last active PCB window. Placement then follows the same sequence as placing from within the PCB Editor.



PCB Components Attributes

Once a footprint is placed in the workspace, it becomes a component. It has a designator, such as R3 and a comment, say 10K. It also behaves as a single object rather than a collection of tracks, pads, arcs, etc, and can be moved, flipped and rotated as such. To edit a component, select the **Edit » Change** menu item or double-click inside the component outline. The Change Component dialog will pop up.

The Change Component dialog is divided into three Tabs. Component attributes which can be edited include:

Properties Tab

This Tab contains the commonly used

Designator/Comment

The value of the component designator and comment can be changed here. Use the designator and comment Tabs to change the text attributes such as the font, text height and text width.

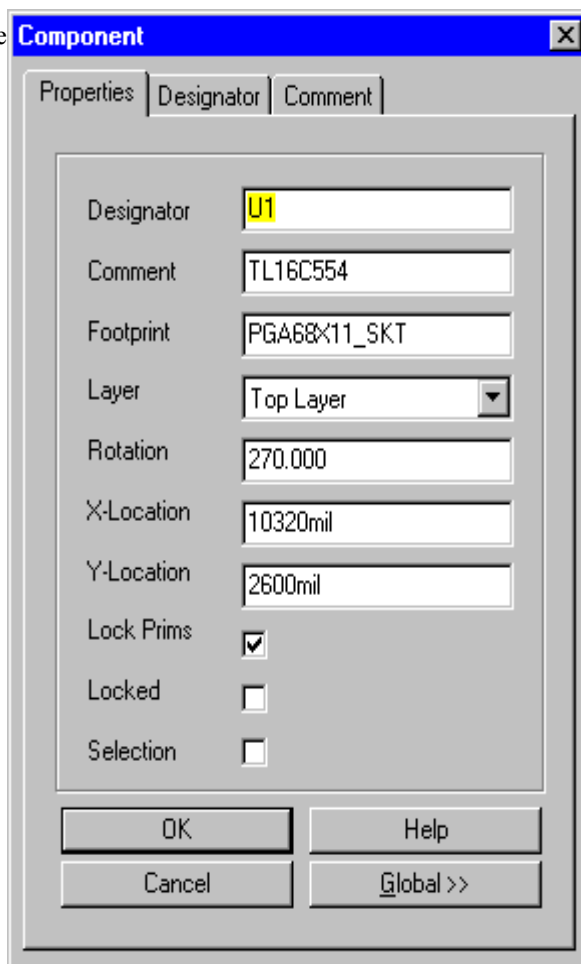
Footprint

The current footprint of the component can be changed to any other available footprint in any open component library. If you type the name of a different footprint in the Footprint field, when you exit the dialog the PCB Editor will search the current open libraries to try and locate the new footprint.

Layer

Components can be assigned to either the top or bottom layer of the PCB. To change the layer assignment, click in the Layer box and choose Top layer or Bottom layer.

Changing the layer status swaps the component to the



opposite layer. For example, when moving a Top layer component to the Bottom layer, primitives on the Top Overlay layer will be automatically reassigned to the Bottom Overlay layer. The orientation of the component will be flipped along the x axis and the component overlay text will read from the bottom. Single layer pads are also swapped between the Top layer and Bottom layer. You can extend this to do global swaps of components from one layer to another.

Rotation

Components can be rotated to any angle, with an angular resolution of 0.001 degrees.

Lock Prims

Normally it is desirable to manipulate a component as one object. In this state the primitives that make up a component are “locked” together. However, if required you can unlock the primitives and modify or manipulate them. Component primitives should then be re-locked. Component pad attributes can be modified without un-locking the primitives.

Locked

The Locked attribute determines whether a component is fixed in the workspace or is free to move. If the Locked attribute is set the autoplacer will not move the component. If you attempt to manually move the component the warning message “Object is locked, continue?” will pop up, allowing you to move the component without unlocking it. The locked attribute remains set after this move.

Selection

Components can be selected or de-selected. Use this attribute to help qualify components for a global edit operation.

Designator and Comment Tabs

These two Tabs contain the text attributes

Text

The component designator or comment. The designator attribute is not globally editable as each component must have a unique designator. The Comment attribute is globally editable. Designator and comment strings can be a maximum of 255 characters in length.

Height

Text size can be set in mils (.001 in) or mm. Range is 0.01 to 10000 mils. The character width used to display / print the text is automatically proportioned to the height. A minimum text height of 36 mils will allow strings to be legibly photoplotted.

Width

The text stroke width can be set in mils (.001 in) or mm. Range is 0.001 to 255 mils.

Font

Three fonts are available. Click the Font button to choose the Default font, Sans Serif font or Serif font.

Layer

Component text can be assigned to any layer. Click the Layer button to scroll the selection bar through the layer options. The selected layer will be displayed in the Layer box.

Rotation

Component text can be moved and rotated independently of the component. Click and hold on the component text and drag the mouse to reposition the text. Press the SPACEBAR to rotate the text while it is floating on the cursor.

X, Y Location

Location of the text in the workspace, relative to the current (relative) origin.

The screenshot shows the 'Component' dialog box with the 'Designator' tab selected. The fields are as follows:

Field	Value
Text	U1
Height	60mil
Width	6mil
Font	Default
Layer	Top Overlay
Rotation	360.000
X-Location	11000mil
Y-Location	3640mil
Hide	☐
Mirror	☐

Buttons at the bottom: OK, Cancel, Help, Global >>

Hide

Component text can be displayed or hidden. Hidden text will not be printed.

Mirror

Component text can be mirrored independently of the component.

- ◆ To flip a component during placement, press the L key while the component is floating on the cursor. This will mirror the footprint, convert top layer pads to bottom layer pads and mirror the overlay onto the bottom overlay layer. Do not use the X or Y keys, as these will flip the component, but not change its layer.

Changing the Footprint a Component is Using

To change an existing component from one footprint to another, double-click inside the component outline to pop up the Component dialog. Type the new footprint name into the Footprint field of the Attributes Tab.

The current component footprint can be changed to any other available footprint in any open library, or library currently listed in the Panel. When you type a different name in the Footprint field and click OK to exit the dialog, the PCB Editor will search the current open libraries, then the libraries listed in the panel, to locate the new footprint.

Component footprints can be changed freely. However, if there are netlist connections to the pads the new footprint must have the same used pin numbers available as the previous one. If it does not the warning message “cannot match pads with new footprint” will be displayed and the substitution will be aborted. For example, changing a DIP16 to an SMD16A is a legitimate change as the pin numbers match. Changing a DIP16 to a TO-3 would generate a warning and the change would be aborted. If the change is successful the connection lines will also be updated to remain connected to the appropriate pads.

Modifying a Component Footprint on the Board

Generally, if a component footprint requires modification the footprint is edited in the library and then all instances of the footprint on the PCB are updated – click the Update PCB button in the Library Editor panel to do this.

Component primitives can also be added, modified and deleted from a component footprint in the PCB workspace. To modify component primitives on the board you must first unlock them – to do this double-click on the component and disable the Lock Prims option. Once you close the dialog you can edit and delete the existing primitives. Re-enable the Lock Prims check box when you have finished.

You can also add new primitives to a component footprint – to do this first disable the Lock Prims option as before. Now place the new primitives as required, select the new primitives (but not the existing component primitives), then select **Tools » Convert » Add Selected Primitives to Component** from the menus. You will be prompted to click on

the component that you want to add the selected primitives to – when you click on the component the new primitives are added, and then deselected. When you have finished adding primitives re-enable the Lock Prims option. Note that these changes only affect this component, they do not affect the component footprint in the library.

Polygon primitives can also be included in a component footprint. To include polygon primitives in a component footprint the polygon must be added in the PCB workspace. The method of adding primitives to a component is described above. There is an extra step in adding polygon primitives to a component – after placing the polygon and selecting it (shortcut: SHIFT+click), select **Tools » Convert » Explode Polygon to Free Primitives** from the menus and click on the polygon to explode it. Now add these polygon primitives as you would add any new primitive to a component.

Including Routing in Component Footprints

Library components can include routing primitives. If your components include routing primitives there is an option that automatically updates the net name of these primitives as the netlist is transferred from the schematic to the PCB. Enable the Assign Net to Connected Copper option in the Update Design dialog when you select **Design » Update PCB** from the Schematic Editor menus.

Net names can also be applied to routing primitives that are built into components at any time by selecting **Design » Netlist Manager** from the PCB Editor menus. Select the **Update Free Primitives from Component Pads** option from the Menu button at the bottom of the dialog to reapply the pad net names to all connected copper.

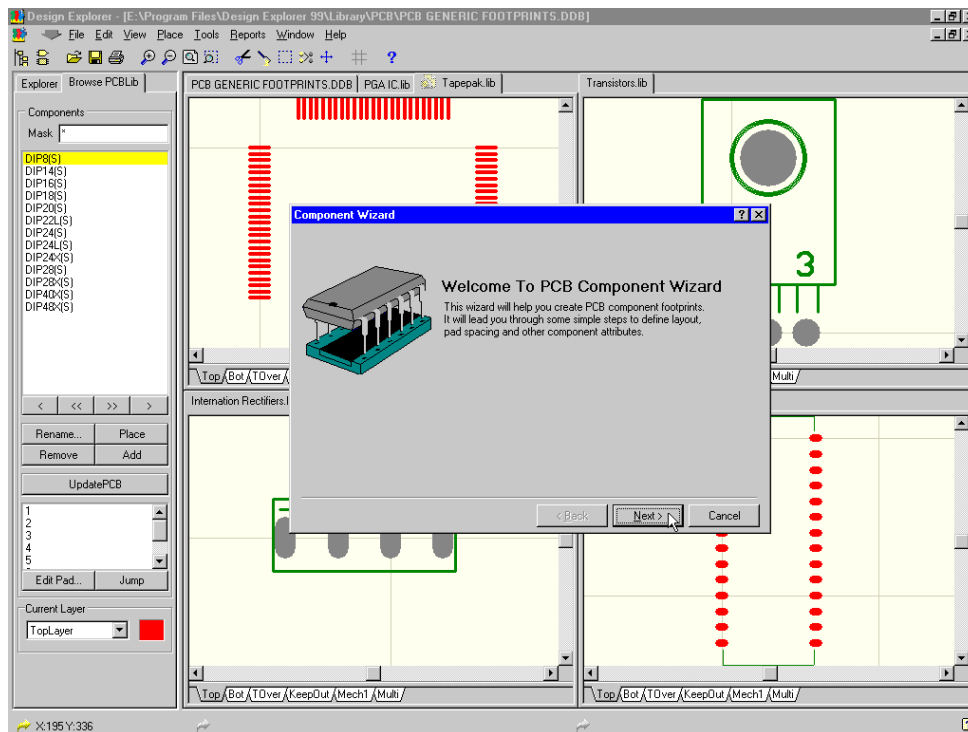
Un-Grouping a Component

If necessary, a placed component can be converted back into the original set of primitive parts. Select the **Tools » Convert » Ungroup Component** menu item. When you launch this process you will be prompted, “Select Component”. The prompt “Confirm convert Component To Primitives” will be displayed. If you click YES the component designator and comment will be removed from the component and it will become a set of primitives. This is a one-way process, it is not possible to re-group an un-grouped component. Un-group has no effect on the component footprint stored in the library – only on the individual instance of the component placed in the document window.

Copying Components from the PCB to a Library

Components can be copied directly from the PCB workspace into an open library – first select the component(s), choose **Edit » Copy** from the PCB Editor menus, click when you are prompted to select a reference location, change to the target library, right-click in the component list in the Browse PCBLib panel, and select **Paste** from the floating menu. Each of the selected components will be added to the library.

The PCB Library Editor



The PCB Library Editor is the second PCB Document Editor provided by Protel 99 SE's PCB server. Where the PCB Layout Editor is used to design the printed circuit board, the PCB Library Editor is used to create and modify the component footprints used on those PCBs. It is also used to manage the PCB libraries.

The PCB Library Editor includes a complete set of processes for creating, editing and placing library footprints. Custom libraries can be created and any number of component libraries can be opened at the same time, limited only by available memory. There is no limit to the number of component footprints that each library can hold.

Components generally include one or more pads (corresponding to component pins and numbered accordingly) plus track and/or arc segments on the overlay (silkscreen) layer to define the component body.

◆ The Altium Library Development Center is constantly developing new libraries – check www.protel.com to download the latest PCB footprint libraries.

Working with Libraries

In Protel 99 SE PCB footprint libraries are stored within a Design Database, just like all the other design documents. In this Handbook they are referred to as PCB Library Databases, but these databases are exactly the same as the Design Databases you store your design documents in.

The Protel 99 SE PCB Library Databases are stored in the \Program Files\Design Explorer 99 SE\Library\Pcb folder.

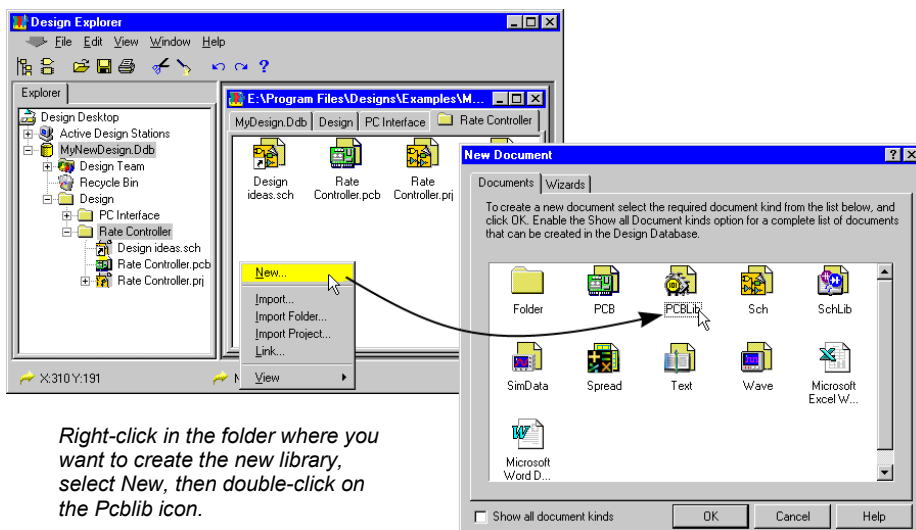
Opening an Existing Library

Libraries are opened in the PCB Library Editor in the same way all documents are opened in the Design Explorer, by first selecting **File » Open** to open the PCB Library Design Database, then browsing through the database and opening the library for editing. Each open library appears on a separate Tab in the integrated Design Window.

Creating a New Library

Before you create a new library you must open the Design Database that you want to store the library in, then browse through the database and open the folder that you want to create the library in.

To create the new PCB library right-click in the folder window, and select **New** from the floating menu that appears. The New Document dialog will pop up, double-click on the PcbLib icon to create a new library.



An icon for the new library will appear in the folder, with a name like PcbLib1. To rename the new library click once to select the icon, then press F2 to highlight the

name, ready for typing. Type in the new name and press ENTER on the keyboard. Double-click on the icon to open the new library.

Creating a Component Footprint with the Component Wizard

The PCB Library Editor includes a powerful component creation Wizard. This Wizard will ask a few questions, and then build the component footprint for you, from a simple two pin resistor through to a Pin Grid Array with hundreds of pins.

To launch the Component Wizard press the Add button on the Library Editor panel or select the **New Component** menu item in the **Tools** menu.

Manually Creating a Component Footprint

Footprints are created in the PCB Library Editor using the same set of design objects available in the PCB Editor. Anything can be saved as a PCB footprint, including corner markers, phototool targets, mechanical definitions, and so on. The typical sequence for manually creating a component footprint is:

1. New component – Open the desired library in the Library Editor. Select the **Tools » New Component** menu item. The Component Wizard will automatically start, press Cancel to manually create a component. You will be presented with an empty component footprint workspace, called PCBComponent_1. Select **Tools » Rename Component** to change this to the required name, of up to 255 characters.
2. Place the pads – place the pads according to the component requirements. When a pad is floating on the cursor select **Edit » Jump » Reference** (shortcut: J, R) to position the cursor at the workspace 0, 0 coordinate. Prior to placing the first pad, press the TAB key to define all the pad attributes.

◆ Always build surface mount footprints on the top layer. Use the L shortcut key to flip them to the bottom layer during placement.

3. Component outline – Use the track tool to create the component outline on the Top Overlay layer. Use the SPACEBAR to change between the Start and End placement modes. Press SHIFT+SPACEBAR to change track placement modes.
4. Save the library. You can now return to the PCB Editor and place this component.

◆ Always build the component around the workspace 0,0 reference point. The Reference is the point you will be “holding” the component by when you place it. Use the **Reference** options in the **Edit** menu to move the Reference if it needs to be changed.

◆ The special strings, .DESIGNATOR and .COMMENT can be added to the component in the Library Editor if you require control over their layer, location and text attributes prior to placing the component. These will be in addition to the standard designator and comment which can be hidden if required.

Copying Components between Libraries

Components can be copied and pasted between libraries, from a library to a PCB, and from a PCB to a library. To copy between libraries, or from the library to a PCB, select the component(s) in the Library Editor panel using the standard Windows selection keys (left-click, SHIFT+click and CTRL+click). Once the components have been selected click the right mouse button to pop up the floating menu and select **Copy**. Change to the target library, right-click in the Library Editor panel, and select **Paste** to add them to the target library. If you are pasting directly onto a board select **Edit » Paste** from the PCB Editor menus.

Updating a Footprint

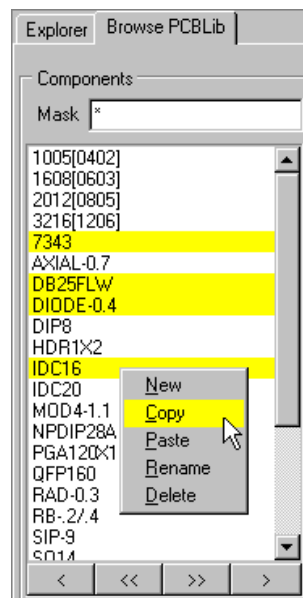
After editing a footprint in the PCB Library Editor, use the Update PCB button in the PCB Library Editor Panel to update all instances of this footprint in all open designs.

Checking the Components

Select **Reports » Component Rule Check** to pop up the Component Rule Check dialog. The Component Rule Checker tests for duplicate primitives, missing pad designators, floating copper and inappropriate component references.

Creating a Project Library

The PCB Editor can create a library of all components currently placed in the workspace. To use this feature select the **Design » Make Project Library** menu item. Enter the library name in the Save Project Library dialog and click OK. All components in the workspace will be added to the library.



Use the floating menu in the panel to copy and paste components between libraries

Defining the Board

The first step in the PCB design process is to define the board boundaries and layers. While the finished board is a 3 dimensional object, it is effectively designed and fabricated as a set of flat, 2 dimensional layers. To manufacture the board these layers are “sandwiched” together (with layers of insulation separating them), to create the 3 dimensional PCB.

Everything you design in the PCB workspace is done by placing design objects (from the **Place** menu) on the various layers available in the workspace – the physical outline of the board is created with tracks on a mechanical layer, components and routing are placed on the top and bottom signal layers, and so on.

♦ If the board outline has been defined in a mechanical package, such as AutoCAD, you can transfer it into the PCB Editor using the Import DWG/DXF feature.

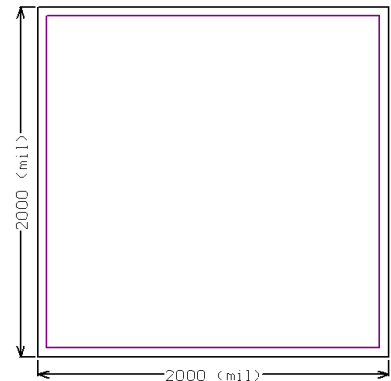
Protel 99 SE includes a powerful Board Wizard that guides you through the complete process of creating a new PCB document and board definition. The Wizard include a number of pre-defined board templates, and also allows you to create your own templates.

Creating the mechanical definition of the PCB

The exact detail required for the mechanical definition of a board will depend on company and manufacturer requirements, and could include a mechanical outline, keepout boundaries, cutout details, dimension details, photo tool targets and so on. Generally, manufacturers require board corner markers, a reference hole location, and external dimensions as a minimum. Contact your PCB manufacturer for details of their requirements.

The actual physical boundary of the board is defined by placing tracks on one of the mechanical layers. Select **Design » Mechanical Layers** to enable and name the required mechanical layers.

To create the board boundary first make the appropriate layer the active layer by clicking its Tab at the bottom of the PCB Window (select **Design » Options** from the menus to display/hide a layer). Then select **Place » Line** from the menus to start defining the boundary, refer to the *Tracks*



Example board outline – the outer box is the physical board boundary, defined on a mechanical layer, the inner box is the placement and routing outline on the keepout layer

topic in the *PCB Design Objects* chapter for more information on track placement techniques. It is good practice to design the physical board outline starting in the lower left region of the workspace. One inch in, one inch up from the absolute origin is often used as a position for the lower left corner of the board.

◆ The content of any mechanical layer can be added to all output layers during output generation – use this feature to include common information such as boundaries, dimensioning, photo tool targets and a title block on each of the output layers.

Defining the Placement and Routing Outline

The placement and routing outer limits are defined by placing tracks and arcs on the Keepout layer to define the electrical outline of the board. Typically this boundary is set slightly in from the physical edge of the board, ensuring that tracks and components do not get too close to the edge. This boundary is used by the Design Rule Checker, the autoplacer and the autorouter to limit placement and routing within the boundary. Refer to the example files supplied with Protel 99 SE for examples of how the physical and keepout boundaries can be defined.

You can also define "no go" regions within this outline where components and/or tracks are to be excluded. This can include zones for mounting hardware and regions required for board profiling. These zones are created by placing design objects such as tracks, arcs and fills on the Keepout layer, within the outer boundary.

Keepout regions defined on the Keepout Layer apply to all signal layers. The basic rule when using the Keepouts is that routing on signal layers will not cross over design objects on the Keepout layer.

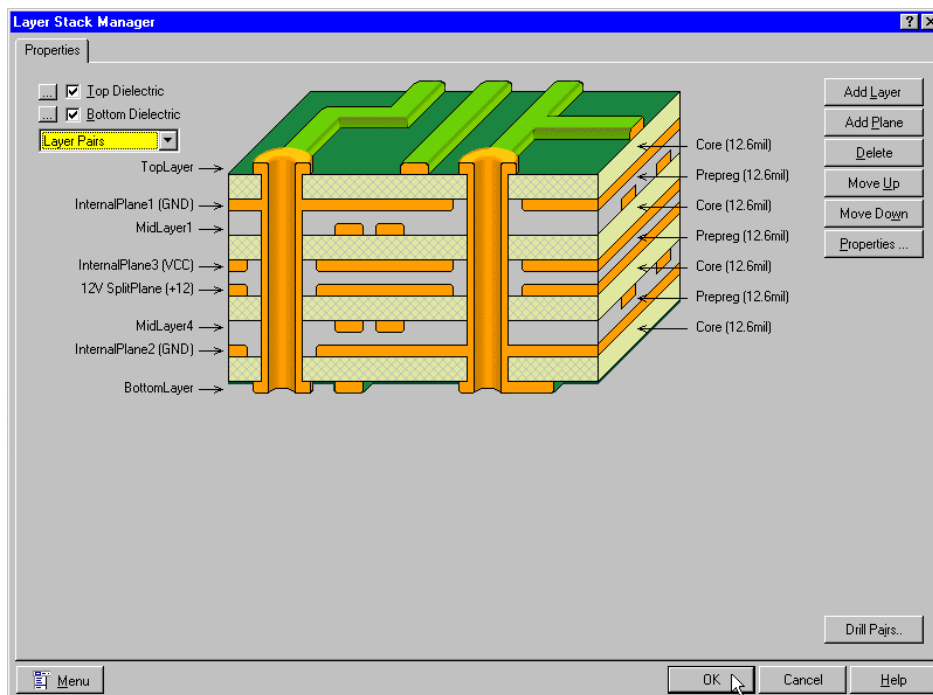
Layer-specific keepout regions can also be defined, these are created by placing Keepout objects from the **Place » Keepout** sub-menu on the required layer.

Defining the PCB Layer Stack

Once the mechanical definition is complete the next step is to define the set of electrical layers that will be used to make up the actual board – this set of layers is referred to as the *layer stack*.

There are 2 types of electrical layers – signal layers, which contain the signal interconnect paths, and power planes, which are layers of unbroken copper used to distribute current to power the components.

The layer stack is defined in the Layer Stack Manager dialog (**Design » Layer Stack Manager**). The image in the center of the dialog shows the current layer stack, the default is for a double-sided board. New layers can be added to the design by clicking on the Add Layer and Add Plane buttons. Each new layer is added below the layer that is currently selected. Double-click on a layer name to edit the properties of that layer.



The electrical layers are added and their order defined in the Layer Stack Manager – right-click on the image to copy it to the Windows clipboard and include in your documentation

The Menu button at the bottom of the dialog includes a number of pre-packed example layer stacks. Note that these example layer stacks are not fixed, you can start with one of these and easily modify it. Once the required layers have been added, use the Move Up and Move Down buttons to configure the layer stack. New layers can be added at any point in the design process.

There are a total of 32 signal layers available (top layer, bottom layer, and 30 mid-layers) and 16 plane layers. Layer visibility is controlled in the Document Options dialog (**Design » Options**).

Selecting the Layer Stack-up Style

As well as the electrical layers, the stack-up includes the non-electrical insulation layers. There are typically 2 kinds of insulation used in the fabrication of a PCB, these are often referred to as core and prepreg layers.

The stack-up style refers to the order of the insulation layers through the layer stack. Three default stack-up styles are supported – layer-pairs, internal layer-pairs, and build up. Changing the layer stack-up style changes the way that the core and prepreg layers are distributed through the layer stack.

Select the preferred stack-up style at the top left of the Layer Stack Manager dialog. Defining the stack-up style is only required if you plan to use blind and buried vias, and for signal integrity analysis. If you are planning to use blind and buried vias you must consult with your PCB manufacturer to ensure that they can fabricate the design, and that the correct stack-up style is selected.

Defining the layer properties

There are 3 kind of layers added to the layer stack in the Layer Stack Manager; signal layers, plane layers and insulation (substrate) layers. The information in these dialogs must be correctly specified if you intend to perform a signal integrity analysis (**Tools » Signal Integrity**).

Signal Layers

Name – user-defined layer name

Copper thickness – this value is required for signal integrity analysis

Plane Layers

Name – user-defined layer name

Copper thickness – this value is required for signal integrity analysis

Substrate and Dielectric Layers

Material – material kind, this is entered for reference

Thickness – the dielectric (substrate) thickness is required for signal integrity analysis

Dielectric constant – dielectric constant of the substrate, required for signal integrity analysis

Defining the drill pairs

Part of defining the layer stack-up is to specify the drill-pairs. The term drill-pairs refers to the 2 layers that a drilling operation starts from, and stops at. Unless the board includes blind and buried vias only one drill-pair is required, comprising the Top and Bottom layers. This drill-pair is on by default and can not be deleted or modified.

Drill-pairs are defined in the Drill-Pair Manager dialog, click on the Drill-Pairs button in the Layer Stack Manager dialog to display this dialog.

If the design includes blind and buried layers then the drill pairs must be defined to suit the layer stack-up style. This should be done in consultation with your board manufacturer to ensure that your design matches their fabrication technology.

Using The Board Wizard

The PCB Editor includes a board Wizard, which allows you to select from a large number of industry-standard board templates. The templates include; a title block, alignment markers, reference rulers, dimensions, and standard edge connectors. The

Wizard also enters text into the title block, and lets you specify the number of routing layers and the track/pad technology. Start the PCB Wizard from the Wizards Tab of the New Document dialog.

The Wizard includes an option to create a custom board. If you select this you define the size and shape of the outline, and can also include cutouts. On the last setup page of the Wizard there is a check box to save this board as a template – if you do, this template will appear at the bottom of the list of pre-defined templates the next time you run the Wizard.



Use the PCB Wizard and select an industry-standard template, or create and save a custom template

If you enable the Save the Board as a Template option the board is saved in the \Program Files\Design Explorer 99 SE\System\Templates.Ddb design database. You can edit this PCB at any time to modify the template. You can also include a custom bitmap for the template – create the image as a 64x32 pixel (16 color) bitmap file, save it with the same name as the PCB template, then import it into the \Boards folder in the Templates.Ddb design database.

Transferring the Design from the Schematic

Once you have defined the board outline and mechanical details, you are ready to transfer the design information from the schematic, to the PCB. Refer to the chapter, *Transferring the Design Information to the PCB*, in the *Schematic Capture* section of the Handbook, for details on how to do this.

Specifying the PCB Design Requirements

PCB design is no longer a matter of simply placing tracks to create connections. High speed logic combined with smaller and more complex packaging technologies place new demands on the PCB designer. It is no longer possible to satisfy all the design requirements by only considering the clearance between tracks, pads and vias. Today's designs can also require that you apply specific requirements to individual nets, components or regions of the board as well as considering such issues as crosstalk, reflections and net lengths. To successfully complete designs such as these you need a tool that can be configured to monitor and test for these requirements. Protel 99 SE allows you to do this by defining *design rules*.

The PCB Editor incorporates a large set of design rules. These include clearances, object geometry, placement, parallelism, impedance control, signal integrity, routing priority and routing topology.

Each rule has a *rule scope* that defines how it is applied. The rule scope defines exactly what objects that rule is to apply to – this scope can be as broad as the whole board, or as narrow as an individual pad. Multiple rules of the same kind can be defined – for example a clearance constraint that applies to the whole board, then a second clearance constraint that applies to an individual net – the rule system resolves these rules in a hierarchical fashion, only applying the highest priority rule to the objects covered by that rule.

What are Design Rules?

You design your PCB by placing components, tracks, vias and other design objects. These objects must be placed in the workspace with regard to each other. Components must not overlap, nets must not short, power nets must be kept clear of signal nets, different nets must be routed at different widths, certain nets must have equal lengths, and so on.

To allow you to remain focused on the task of designing the board, the PCB Editor can monitor these design requirements for you. You instruct the PCB Editor of your requirements by setting up a series of design rules. The PCB Editor monitors the placement of objects in the workspace, and as soon as an object is placed in violation of a design rule it is highlighted.

◆ The PCB Editor allows you to create classes (sets) of objects, which are very handy for setting up and managing the design rules. Refer to the *Creating Object Classes* topic later in this chapter for details on how to create a class of Nets, Components, or From-Tos.

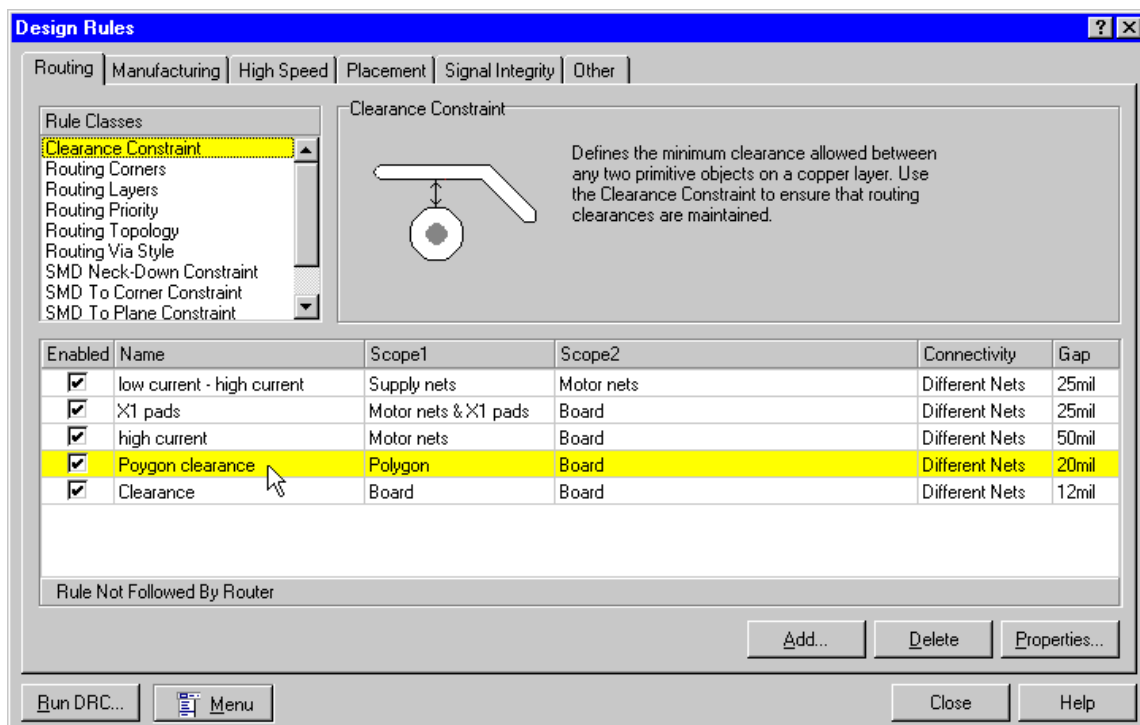
Defining the Design Rules

Where are Rules Defined?

Rules are listed, added and edited in the Design Rules dialog. Select the **Design » Rules** menu item to pop up this dialog.

The set of rules available in the PCB Editor is divided into six groups, with each group on a separate Tab in the Design Rules dialog. On each Tab there is a set of rules listed on the left – click to select a different rule. Next to this list is a description of the currently selected rule.

The lower half of the dialog lists all instances of that rule that have been defined for this PCB (including rules that are automatically created for a new PCB). Note that rules that have been created can be individually disabled – use this feature when you do not want a rule to be applied, but you do not want to delete it.



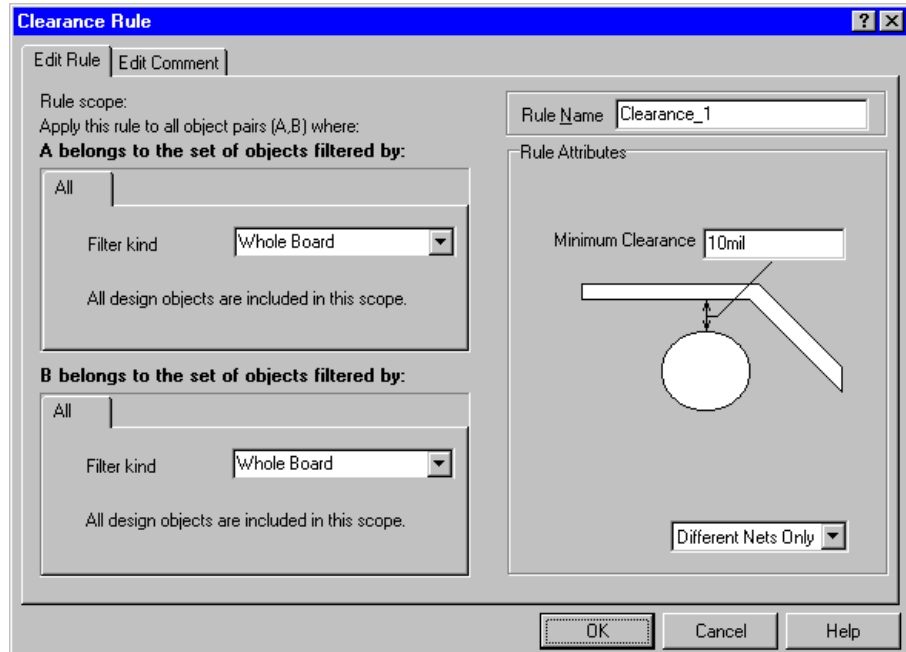
Examine and edit the rules in the Design Rules dialog

- ◆ Current rules in the lower half of the Design Rules dialog are listed from the highest priority rule, down to the lowest priority rule.

Adding a Rule

Locate and select the rule you require in the Design Rules dialog and press the Add button (shortcut: double-click LEFT MOUSE on the rule). After pressing the Add button a dialog for that particular rule will pop up. This is where you configure the rule and set the scope.

Note that each rule has a Rule Name, when a new rule is created it is assigned a default name. The rule name is displayed in the panel when the Browse mode is set to Rules, use the name to simplify the process of managing the rules.



Each rule is configured in its own Rule dialog

What is the Rule Scope?

The scope, or extent of each Rule is determined by the Rule Scope. The rule scope defines the set of target objects that a particular instance of a rule is to be applied to. By setting the scope you could apply a rule to the whole board, or you could target a particular net, a set (class) of nets, a component, or a pad.

♦ The rule scope defines exactly what objects the rule is to apply to.

Description of each Scope

Following is a summary of each rule scope.

Whole Board

This is the lowest priority scope available – it targets all objects on the board that this rule can be applied to.

Layer

Targets all objects on the specified layer. Note that objects are not targeted by this scope. An Example of when this scope could be used includes defining different paste mask openings for the top layer surface mount components from the bottom layer surface mount components.

Object Kind

Targets the enabled object types. Examples of when this scope could be used include when you need a different track to pad/via clearance, or when you need different clearance for polygons from all other routing.

Footprint

Targets all components that use the specified footprint. An advantage of this scope is that as new components that use the specified footprint are added to the design, they are automatically covered by the rule. Examples of when this scope could be used include footprint-specific rules such as mask and paste expansions, or the component clearance rule.

Component Class

Targets all components that belong to the class. Examples of when this scope could be used include grouping components for placement (using the Room Definition rule), or for specifying allowable component orientations (Component Orientations rule).

Component

Targets the specified component. Examples of when this scope could be used include solder and paste mask rules, and placement rules.

Net Class

Targets all the objects that belong to each of the nets in the class (including tracks, arcs, fills, pads and vias). Examples of when this scope could be used include clearance constraints and width constraints (applies to the tracks and arcs), as well as high speed and signal integrity rules.

Net

Targets all the objects that belong to the net routing (including tracks, arcs, fills, free pads and vias). Examples of when this scope could be used include clearance constraints and width constraints (applies to the tracks and arcs), as well as high speed and signal integrity rules.

From-To Class

A from-to is a user-defined connection in a net, going *from* a specific pad *to* another specific pad (**Design » From-To Editor**). This scope targets all the routing tracks, arcs and vias that belong to the from-tos in the class. Examples of when this scope could be used include clearance constraints and width constraints (applies to the tracks and arcs), as well as high speed and signal integrity rules.

From-To

A from-to is a user-defined connection in a net, going *from* a specific pad *to* another specific pad (**Design » From-To Editor**). This scope targets all the routing tracks, arcs and vias that belong to the from-to. Examples of when this scope could be used include clearance constraints and width constraints (applies to the tracks and arcs), as well as high speed and signal integrity rules.

Pad Class

Targets all pads that belong to the pad class. Examples of when this scope could be used include mask expansions, polygon connection styles, plane connection styles, and hole size constraints.

Pad Specification

Targets all the pads that satisfy the enabled pad criteria. Click the Specification button in the Rule dialog to display the Pad Specification dialog. Each of the enabled attributes is used in the scope, disable the attributes that are not required and configure those attributes that will be used to test against. Examples of when this scope could be used include mask expansions, polygon connection styles, plane connection styles, and hole size constraints.

Via Specification

Targets all the vias that satisfy the enabled via criteria. Click the Specification button in the Rule dialog to display the Via Specification dialog. Each of the enabled attributes is used in the scope, disable the attributes that are not required and configure those attributes that will be used to test against. Examples of when this scope could be used include mask expansions, polygon connection styles, and hole size constraints. Another example could be a power plane connection style rule, a via specification can be used to prevent vias connecting to a power plane. Refer to the examples later in this chapter for an example of how to do this.

Footprint Pad

This scope allows a design rule to target a specific pad (or pads if wildcards are used) in the specified footprint.

Pad

Targets the specified pad. Examples of when this scope could be used include mask expansions, polygon connection styles, plane connection styles, and hole size constraints.

Region

Targets a specific region of the board. The region is defined by clicking the small Define button that appears on the rule dialog when you set the scope to region. An object is considered to be within the region if any part of the object's bounding rectangle (the smallest rectangle that could be drawn to completely encompass the object) lies within the region. Examples of when this scope could be used include polygon and power plane connection styles, and width constraints.

Unary and Binary Rules, and Setting Their Scope

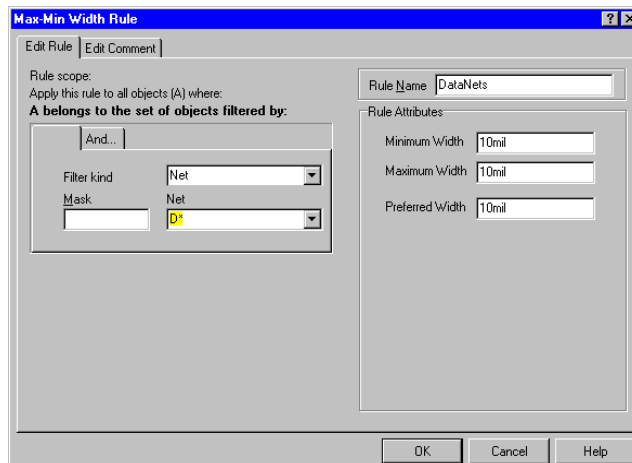
There are two types of design rules, *unary rules* and *binary rules*. Unary rules apply to one object, or each object in a set of objects. Binary rules apply *between* two objects, or between any object in one set to any object in the second set.

An example of a unary rule is the solder mask expansion rule. This rule applies individually to each pad identified by the rule scope. An example of a binary rule is the clearance constraint, which applies *between* any copper object in the first set and any copper object in the second set, as identified by the two rule scopes. When you configure a unary rule you set up one rule scope, when you configure a binary rule you set up two rule scopes.

Using Classes and Wildcards in the Rule Scope

There are many ways of using the design rules to satisfy your design requirements. To help keep the set of rules manageable there are 2 powerful ways of defining rule scopes that target sets of objects.

You can create user-defined sets of objects, referred to as Classes, and then configure the design rule to target that class of objects. Supported classes include net class, component class, pad class and from-to class. Refer to the *Creating Object Classes* topic later in this chapter for details on how to create a class of objects.



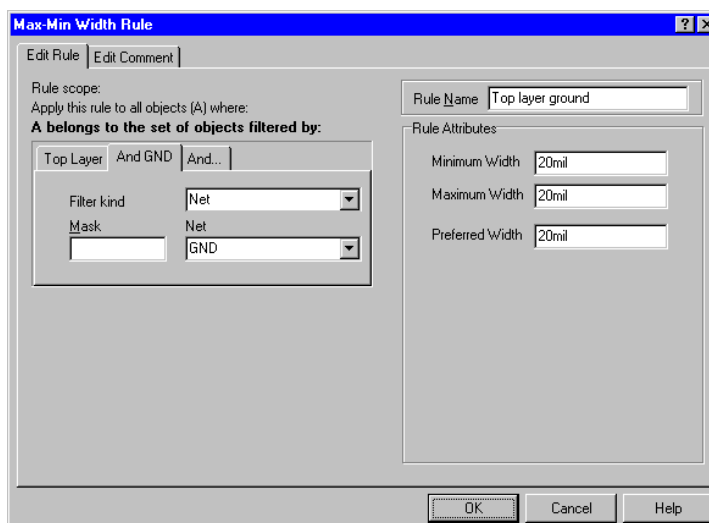
Using a wildcard to identify a set of nets

Wildcards can also be used to define a set of objects. Both the any single character (?) wildcard and the any characters (*) wildcard are supported. In the adjacent figure the Width requirement for all the Data nets (starting with the letter D) is set to 10 mils. The rule specifies that all nets, whose net name starts with the letter D, must have a width of 10 mils.

Compound Rule Scopes

Sometimes you will need to target a set of objects that are a subset of what you can target with a single rule scope. An example would be a net that must be routed at a certain width, except on one layer, where it must be routed at a different width.

To allow more specific rules to be created, each design rule allows you to create a *compound rule scope* for that rule. A compound scope is where you logically AND together more than one scope, thereby narrowing down the set of objects that the rule will target.



Using a compound scope to set the width of the GND net to 20 mils on the top layer

If you are attempting to set up a rule with a compound scope, it is important to remember that for the rule to apply, *all* of the scope conditions must apply to the object, if one does not, then the rule does not apply. To check if a rule is correctly defined to apply to an object use the **Applicable Rules** options in the right-click menu – if the rule is not listed then it is not applicable, and must be reconfigured. Refer to the topic Working with Design Rules layer in this chapter for more information on the Applicable Rules feature.

Refer to the *Examples of Using Design Rules* topic later in this chapter for examples of compound rule scopes.

Multiple Rules of the Same Kind and their Order of Precedence

The Rule Scope allows you to identify exactly what you want to apply a particular rule to. For example, you can apply a clearance constraint to a net, to an individual connection on the net (a From-To), or to an individual pad.

As well as using the scope to define the set of objects that you want to apply the rule to, you can also use the scope to override one rule with another rule of the same kind.

Each rule can be applied as many times as you require. For example, you can apply a solder mask expansion rule to the whole board, apply a second solder mask expansion rule to a particular component, and apply a third solder mask expansion rule to an individual pad on the same component. So that the PCB Editor knows which of these three rules to apply to this pad, there is an *order of precedence* for rules of the same kind, which have different scopes. The order of precedence for rule scopes from highest priority to lowest is:

- Region (highest priority)
- Pad
- Footprint-Pad
- Via Specification
- Pad Specification
- Pad Class
- From-To
- From-To Class
- Net
- Net Class
- Component
- Component Class
- Footprint
- Object Kind
- Layer
- Whole Board (lowest priority)

◆ So that you do not have to remember the order of precedence when setting the scope, the rule scopes are ordered from lowest priority to highest priority in the drop down selection lists.

This allows you to use a “most-general to most-specific” strategy when you apply rules. Apply general rules to the whole board, then build up your requirements by applying more specific rules with a higher scope precedence. The PCB Editor will analyze the rules applied to each design object, and identify and apply only the rule with the highest precedence.

Contentions Due to Duplicate Rules

When an object is covered by more than one rule with the same scope (for example a pad covered by two solder mask expansion rules, both with the scope set to region, but whose regions overlap) a *contention* exists. The PCB Editor has pre-defined strategies to resolve each possible contention. The basic approach is to choose the rule that gives the greatest margin of safety (for example, the largest clearance). How this is interpreted for each rule is documented with each rule description.

How Rules are Applied

As mentioned previously, multiple design rules of the same kind can be defined. It is important to note that only one design rule of any kind can apply to an object – the rules system automatically resolves the applicable rules to determine which rule applies. Read the following topic, *Working with Design Rules* for information on strategies on determining which rule is being applied to an object.

It is also important to note that not all the design rules are applied in all situations. Certain rules are specifications for the autorouter (eg routing layers), others are specifications for creating manufacturing output (eg solder mask expansion), and neither of these rules are tested by the DRC feature. The PCB Editor applies the rules in the following situations.

On-line Design Rule Check (DRC)

A violation of the rule is flagged as soon as the violation occurs during placement. It is flagged by outlining the objects in violation in the current DRC color. The On-line DRC feature can be disabled in the Options Tab of the Preferences dialog.

Batch DRC

Selecting the **Tools » Design Rule Check** menu item will pop up the Design Rule Check dialog. Enable those Rule Types you wish to test and press the OK button. All instances of the enabled rule types will be tested.

Note that you can set the number of violations to report. Use this to keep the report manageable.

During a Software Operation

Certain rules are monitored during a software operation including; polygon pour, autorouting, autoplacement and output generation. Examples of these include; the mask expansion rule which is monitored during output generation and the routing via style rule which is monitored during autorouting.

Exported with the Design

Certain rules are included to support features in the SPECCTRA™ autorouter. The requirements specified by these rules are exported with the design.

◆ Refer to the Rule Definitions later in this chapter for details on when each rule is applied.

Working with Design Rules

Design Rules are the basis of design specification and control in the PCB Editor. During the different phases of the board design you will be focused on different design requirements – for example during component placement component clearances will be important, during routing the routing clearances will be important. The PCB Editor includes a number of features to help you manage the various design rules.

Monitoring the Design Rules as You Work

Many of the design rules can be monitored as you work. Enable the rules you want monitored on-line in the On-line Tab of the Design Rule Check dialog (select **Tools » Design Rule Check**). Apart from the Un-Routed Net Constraint, any design rule violation that is created during the design process is immediately highlighted in green (the default DRC error color). Each violation is also listed in the PCB panel, set the Browse mode to Violations to display them.

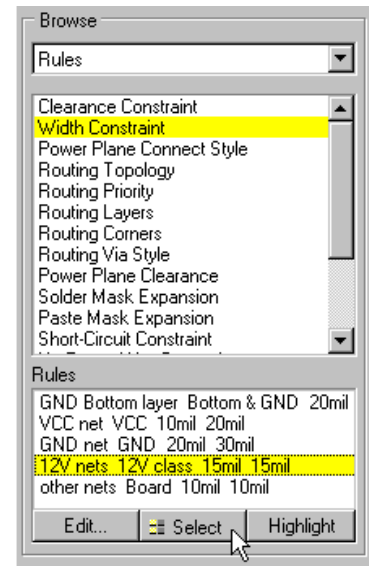
Read the chapter *Verifying the PCB Design* for more information on design rule checking and monitoring.

Working with Rules from the PCB Panel

The rules that are currently defined for the PCB can be displayed in the PCB panel, set the Browse mode to Rules to do this. Use the buttons on the panel to Edit a rule, Select all the objects that this rule applies to, or Highlight all the objects that this rule applies to. Note that the Select and Highlight features resolve the rule precedence and only select/highlight objects that this rule applies to.

Disabling Rules

Rules can be individually disabled in the Design Rules dialog. Disabling a rule has the same effect as deleting the rule in terms of how it is handled by the on-line and batch DRC routines. Disabled rules are displayed in the panel with a line drawn through their name.



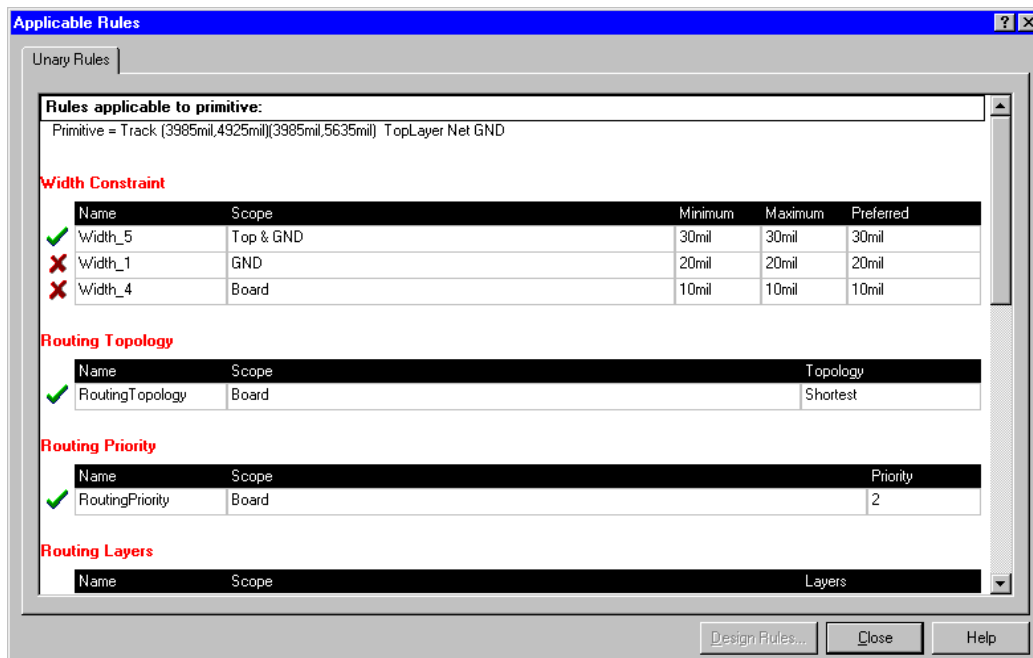
Set the Browse mode to Rules to easily examine and edit the rules

Importing and Exporting Rule Sets

Rule sets can be exported and imported, allowing you to store and retrieve favorite rule sets. To import or export a rule set click on the Menu button at the bottom of the Design Rules dialog. When importing rules you can either Overwrite or Add them by clicking the appropriate button in the rule Import Options dialog. Overwrite occurs when a rule with the same rule name is encountered.

Checking which Rules Apply

You can easily check which design rules are being applied to any object. There are 2 applicable rules options in the right-click floating menu, after selecting either **Applicable Unary Rule** or **Applicable Binary Rule** from the floating menu you will be prompted on the Status bar to select either one primitive (**Applicable Unary Rule**) or 2 primitives (**Applicable Binary Rule**) – click on the primitive(s) of interest.



The Applicable Rules dialog details all rules that apply to the primitive(s)

The Applicable Rules dialog then appears, displaying all design rules that apply to this primitive(s). Note that all current design rules that could be applied to the selected primitive(s) are analyzed and listed in the Applicable Rules dialog. Each rule that is listed in the dialog will have either a tick or a cross next to it. A tick indicates that this is the highest priority rule out of all applicable rules, and is the rule being applied. Lower priority rules of the same kind are listed with a cross next to them, indicating that they are applicable but as they are not the highest priority rule they are not currently applied.

On a more global level you can examine what objects each rule applies to by setting the browse mode in the Panel to Rules, and then using the Highlight and Select buttons to show which objects any rule applies to.

When you want to know why an object is flagged as being in violation right click on it and select **Violations** from the floating menu. The Violation Inspector dialog will appear, detailing which design rule(s) this object is not complying with.

Object Classes

A class is a set of objects that you wish to treat as a group. For example, you may wish to group all the power supply components into a class, and the memory chips into another class. You can then work with the entire group of components – targeting the class with various design rules, or selecting the class, then moving them to position them on the board.

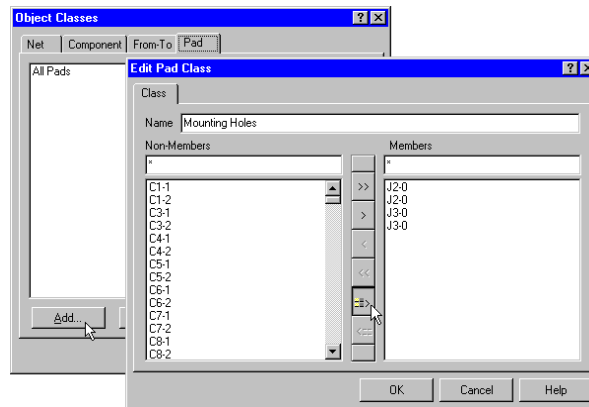
Four types of classes can be defined; net classes, component classes, from-to classes and pad classes. Component classes and net classes can be selected directly from the PCB Editor panel.

◆ Uses classes to easily identify a group of objects. Classes make design rules easier to set up and manage.

Classes are not mutually exclusive, an object can belong to more than one class. This allows you to put objects in more than one class, with each class being used for a different purpose. For example, you could have a component in one class to identify it as belonging to a group of components that must be placed in a certain region of the board (using a room definition), the same component in a second class where it is grouped by footprint, where that class of footprints have a rotation rule applied, and in a third class for a group of components that have a specific solder mask expansion.

Creating a Class

All 4 types of classes are created in the same way – in the Object Classes dialog (select **Design » Classes** from the menus). Click on the appropriate Tab at the top of the dialog to select the type of class, then click the Add button at the bottom of the dialog. When the Edit Class dialog appears select the required objects in the Non-Members list (multiple selections are supported), then use the Arrow buttons to transfer the objects to the Members list. Note that the dialog includes buttons to transfer objects that are currently selected on the board.

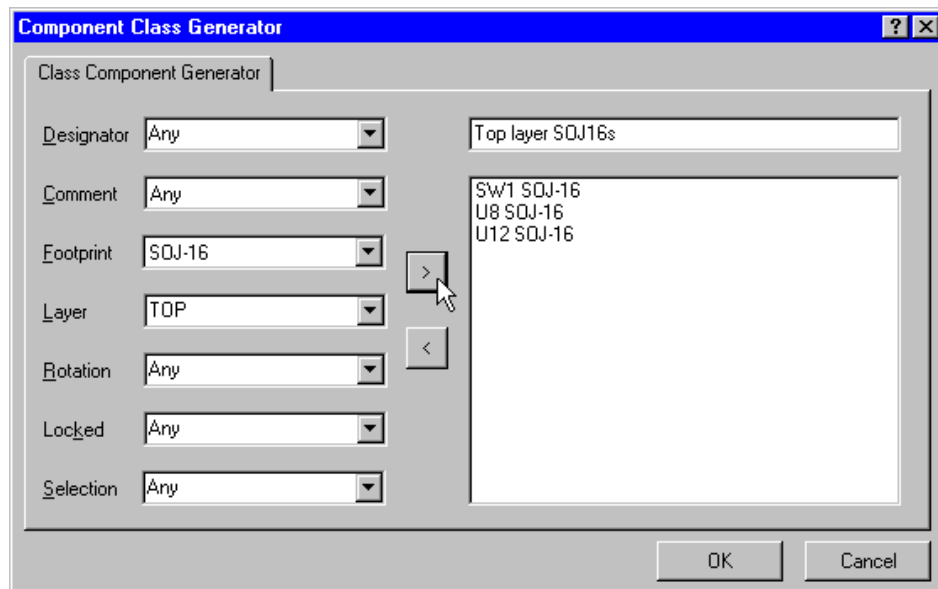


Creating a pad class from pads that are currently selected on the board

Component Class

Component classes can be easily created with the assistance of the Component Class Generator, which allows you to identify a set of components based on common component attributes.

Press the Class Generator button in the Edit Component Class dialog to display the Component Class Generator dialog. Use the filters on the left to identify components by attribute, then click the > button to add all the components that have that attribute value. Set the Class Name at the top of the dialog, then click OK to add this new Class.



Use the class generator to quickly create a class of components based on common component attributes

Routing Rule Definitions

Clearance Constraint

Definition

Defines the minimum clearance allowed between any two primitive objects on a copper layer. Use the Clearance Constraint to ensure that routing clearances are maintained.

Connective Checking

Typically this would be set to Different Nets. An example of when Any Net could be used is to test for vias being placed too close to pads or other vias on the same net, or any other net.

How Duplicate Rule Contentions are Resolved

The rule with the largest clearance is obeyed.

Rule Application

On-line DRC, Batch DRC and during autorouting.

Routing Corners Rule

Definition

Specifies the corner style to be used during autorouting. The corner style can be a 45 degree chamfer or rounded (using an arc). The setback specifies the minimum and maximum distance from the corner location to the start of the corner chamfer or arc.

How Duplicate Rule Contentions are Resolved

The order that duplicate rules are obeyed is; Rounded, 90/45 degrees, 90 degrees.

Rule Application

Export to SPECCTRA.

Routing Layers Rule

Definition

Specifies the layers to be used during autorouting.

How Duplicate Rule Contentions are Resolved

The rule with the minimum number of layers is obeyed.

Rule Application

During autorouting.

Routing Priority Rule

Definition

Assign a routing priority from 0 to 100. 100 is the highest priority and 0 is the lowest. The Routing Priorities are relative values which are used to set the order that the nets will be autorouted.

How Duplicate Rule Contentions are Resolved

The rule with the highest priority is obeyed.

Rule Application

During autorouting.

Routing Topology Rule

Definition

The topology of a net is the arrangement, or pattern of the pin-to-pin connections. By default the PCB Editor arranges the pin-to-pin connections of each net to give the shortest overall connection length. A topology is applied to a net for a variety of reasons; for high speed designs where signal reflections must be minimized the net is arranged with a daisy chain topology; for ground nets a star topology could be applied to ensure that all tracks come back to a common point. The following topologies can be applied with the Routing Topology rule:

Shortest

This topology connects all the nodes to give the shortest overall connection length.

Horizontal

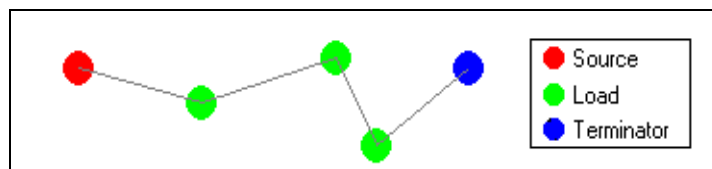
This topology connects all the nodes together, preferring horizontal shortness to vertical shortness by a factor of 5:1. Use this method to force routing in the horizontal direction.

Vertical

This topology connects all the nodes together, preferring vertical shortness to horizontal shortness by a factor of 5:1. Use this method to force routing in the vertical direction.

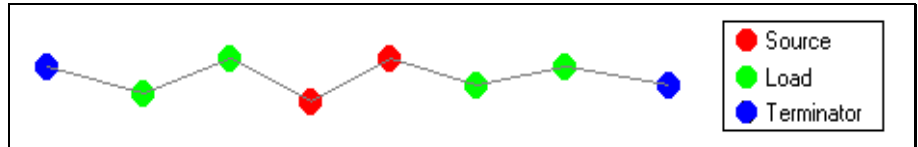
Daisy-Simple

This topology chains all the nodes together, one after the other. The order they are chained is calculated to give the shortest overall length. If a source and terminator pad are specified then all other pads are chained between them to give the shortest possible length. Edit the pad to set it to be a source or terminator. If multiple sources (or terminators) are specified they are chained together at each end.

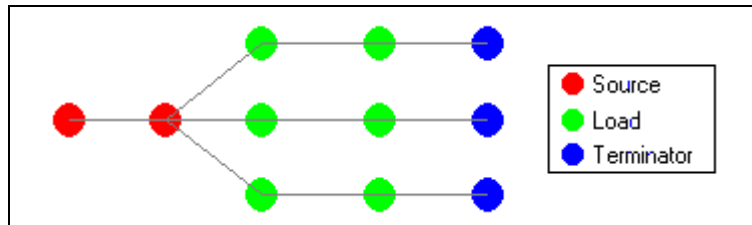


Daisy-Mid Driven

This topology places the source node(s) in the center of the daisy chain, divides the loads equally and chains them off either side of the source(s). Two terminators are required, one for each end. Multiple source nodes are chained together in the center. If there are not exactly two terminators a simple daisy topology is used.

**Daisy-Balanced**

This topology divides all the loads into equal chains, the total number of chains equal to the number of terminators. These chains then connect to the source in a star pattern. Multiple source nodes are chained together.

**Star**

This topology connects each node directly to the source node. If terminators are present they are connected after each load node. Multiple source nodes are chained together, as in the daisy-balanced topology.

How Duplicate Rule Contentions are Resolved

The rules are obeyed in the following order; Star, Daisy-Balanced, Daisy-Mid Driven, Daisy-Simple, Horizontal, Vertical, Shortest.

Rule Application

During autorouting.

Routing Via Style Rule

Definition

Specifies the routing via diameter and hole size.

How Duplicate Rule Contentions are Resolved

The rule with the largest via size is obeyed.

Rule Application

The preferred via attributes are used during manual routing when you press the * shortcut key and toggle routing signal layers, or when you press the / shortcut key to connect to a plane layer. The preferred settings can be changed on-the-fly during manual routing by pressing the TAB key.

The autorouter also uses the preferred via attributes (board scope rule only). The maximum and minimum via attributes are checked by the on-line and batch DRC.

SMD Neck-Down Constraint

Definition

Specifies the maximum ratio of the track width to the SMD pad width, expressed as a percentage.

How Duplicate Rule Contentions are Resolved

The first rule in the list is obeyed.

Rule Application

On-line DRC and Batch DRC.

SMD to Corner Constraint

Definition

Specifies the minimum distance from the center of the surface mount pad to the first routing corner.

How Duplicate Rule Contentions are Resolved

The rule with the largest distance is obeyed.

Rule Application

On-line DRC and Batch DRC.

SMD to Plane Constraint

Definition

Specifies the maximum routing length from the SMD pad center to the plane connection pad/via.

How Duplicate Rule Contentions are Resolved

The first rule in the list is obeyed.

Rule Application

On-line DRC and Batch DRC.

Routing Width Constraint

Definition

Defines the width of tracks and arcs placed on the copper layers.

How Duplicate Rule Contentions are Resolved

The rule with the tightest range is obeyed.

Rule Application

The preferred setting is obeyed during manual and auto routing. It can be changed on-the-fly during manual routing by pressing the TAB key.

The minimum and maximum settings are obeyed by the on-line and batch DRC.

Manufacturing Rule Definitions

Acute Angle Constraint

Definition

Specifies the minimum angle permitted at a track corner. Acute angles can be a problem when manufacturing, resulting in over-etching of the copper at the corner.

How Duplicate Rule Contentions are Resolved

The rule with the largest angle is obeyed.

Rule Application

On-line and Batch DRC.

Hole Size Constraint

Definition

Specifies the maximum and minimum hole size, expressed either as exact numeric values, or as a percentage of the pad/via size.

How Duplicate Rule Contentions are Resolved

The rule with the largest minimum and smallest range is obeyed.

Rule Application

On-line and Batch DRC.

Layer Pairs

Definition

This rule checks to ensure that the used layer-pairs match the current drill-pairs. The used layer-pairs are determined from the vias and pads found in the board, one layer-pair for each Start Layer-End Layer combination that is found.

How Duplicate Rule Contentions are Resolved

The first rule with the enforced layer pairs attribute enabled is obeyed.

Rule Application

On-line and Batch DRC.

Minimum Annular Ring

Definition

Specifies the minimum annular ring allowed on a pad. The annular ring is measured radially, from the edge of the pad hole to the edge of the pad.

How Duplicate Rule Contentions are Resolved

The rule with the largest annular ring is obeyed.

Rule Application

On-line and Batch DRC.

Paste-Mask Expansion Rule

Definition

The shape that is created on the paste mask layer at each pad site is the pad shape, expanded or contracted radially by the Expansion specified in this rule.

How Duplicate Rule Contentions are Resolved

The rule which specifies the smallest expansion is obeyed.

Rule Application

During output generation.

Polygon Connect Style

Definition

Specifies the style of the connection from a component pin to a polygon plane. Three connection options are available; direct connections (solid copper to the pin), thermal relief connections, or no connection.

If Relief Connect is selected you then define; how wide the thermal relief copper connections are, the number of connections and the angle of the connections.

How Duplicate Rule Contentions are Resolved

The first rule which specifies direct connection is obeyed first.

Rule Application

During polygon pour.

Power Plane Clearance

Definition

Specifies the radial clearance created around vias and pads that pass through but are not connected to a power plane.

How Duplicate Rule Contentions are Resolved

The rule which specifies the largest expansion is obeyed.

Rule Application

During output generation.

Power Plane Connect Style

Definition

Specifies the style of the connection from a component pin to a power plane. Three connection options are available; direct connections (solid copper to the pin), thermal relief connections, or no connection.

If Relief Connect is selected you then define; how wide the thermal relief copper connections are, the radial width of the expansion measured from the edge of the hole to the edge of the air-gap, and the width of the air-gap. Note that power planes are constructed in the negative, so a primitive placed on a power plane layer creates a void in the copper.

How Duplicate Rule Contentions are Resolved

The first rule which specifies Direct Connections is obeyed.

Rule Application

During output generation.

Solder-Mask Expansion Rule

Definition

The shape that is created on the solder mask layer at each pad and via site is the pad or via shape, expanded or contracted radially by the amount specified by this rule. To tent a via set the Expansion to a negative value equal to or greater than the via radius. To tent all vias when the design includes different size vias, set the Expansion to a negative value equal to or greater than the largest via radius.

How Duplicate Rule Contentions are Resolved

The rule which specifies the largest expansion is obeyed.

Rule Application

During output generation.

Testpoint Style

Definition

Specifies the allowable physical parameters of pads and vias that are flagged as testpoints.

The Find Testpoint feature and the Autorouter use the Allowed Side settings in the following order of preference (highest to lowest):

- Bottom (surface mount)
- Top (surface mount)
- Bottom via
- Top via

- Bottom thru-hole
- Top thru-hole

How Duplicate Rule Contentions are Resolved

The first instance of the rule is obeyed.

Rule Application

This rule is obeyed by the Find Testpoint feature, the autorouter, and the on-line and batch DRC. The on-line and batch DRC test all attributes of the rule except the Preferred Size and Preferred Hole Size - these settings are used by the autorouter to define the size of testpoint pads that the autorouter places.

Testpoint Usage

Definition

Specifies which nets require a testpoint. The DRC report is used to identify each net that fails this rule, and the Testpoint report feature in the CAM Manager is used to identify the location of valid testpoints.

How Duplicate Rule Contentions are Resolved

The first instance of the rule is obeyed.

Rule Application

This rule is obeyed by the Find Testpoint feature, the autorouter, and the on-line and batch DRC.

High Speed Rule Definitions

Daisy Chain Stub Length

Definition

Specifies the maximum permissible stub length for a net with a daisy chain topology.

How Duplicate Rule Contentions are Resolved

The rule which specifies the smallest stub length is obeyed.

Rule Application

On-line and Batch DRC.

Length Constraint

Definition

Specifies the minimum and maximum lengths of a net.

How Duplicate Rule Contentions are Resolved

The rule which specifies the tightest range is obeyed.

Rule Application

On-line and Batch DRC.

Matched Net Lengths

Definition

Specifies the degree to which nets can have different lengths. The PCB Editor locates the longest net (based on the scope) and compares it to each of the other nets specified by the scope.

The Matched Length Rules dialog also allows you to specify how you would like to match the length of nets which fail the matched length requirements. The PCB Editor will add accordion sections to the nets to equalize their lengths.

If you would like the PCB Editor to attempt to match net lengths by adding accordion sections, set up the Matched Length Rules dialog and then select the **Tools » Equalize Nets** menu item. The matched lengths rule will be applied to the nets specified by the rule and accordion sections will be added to those that fail. The degree of success depends on the amount of space available for the accordion sections and the accordion style being used. The 90 degree style is the most compact and the Rounded style is the least compact.

How Duplicate Rule Contentions are Resolved

The rule which specifies the smallest tolerance is obeyed.

Rule Application

On-line and Batch DRC.

Maximum Via Count

Definition

Specifies the maximum number of vias permitted.

How Duplicate Rule Contentions are Resolved

The rule which specifies the smaller number of vias is obeyed.

Rule Application

On-line and Batch DRC.

Parallel Segment Constraint

Definition

Specifies the distance two track segments can run in parallel, for a given separation. Note that this rule tests track segments, not collections of track segments. Apply multiple parallel segment constraints to a net to approximate crosstalk characteristics that vary as a function of length and gap.

How Duplicate Rule Contentions are Resolved

Duplicate rules do not create contentions for this rule.

Rule Application

On-line and batch DRC.

Vias Under SMT Constraint

Definition

Specifies whether vias can be placed under SMD pads during autorouting.

How Duplicate Rule Contentions are Resolved

The rule which specifies that vias are not allowed is obeyed.

Rule Application

On-line and Batch DRC.

Placement Rule Definitions

Component Clearance Constraint

Definition

Specifies the minimum distance that components must be from each other.

The Check Modes are:

Quick Check – use the components' bounding rectangle to define their shape. The bounding rectangle is the smallest rectangle that encloses all the primitives that make up the component.

Multi Layer Check – also uses the component bounding rectangle, but considers through-hole component pads on a board with components on both sides, allowing surface mount components to be placed under a through-hole component.

Full Check – use the exact shape that encloses all the primitives that make up each component. Use this option if the design includes a large number of circular or irregular shaped components.

How Duplicate Rule Contentions are Resolved

The rule with a tighter gap is obeyed.

Rule Application

On-line and Batch DRC (Quick Check and Multi-Layer Check modes). Also during autoplacing with the Cluster Placer.

Component Orientation Rule

Definition

Specifies allowable component orientations. Multiple orientations are permitted, allowing the autoplacer to use any of the enabled orientations.

How Duplicate Rule Contentions are Resolved

The rule which allows less rotation is obeyed.

Rule Application

During autoplacing with the Cluster Placer.

Nets to Ignore

Definition

Defines which nets should be ignored during autoplacing with the Cluster Placer. Ignoring power nets can assist in placement speed and quality. If the design has a large number of two pin components that connect to a power net, ignoring the power net will

result in these components being clustered based on their other net, rather than the power net.

How Duplicate Rule Contentions are Resolved

Contentions are not possible.

Rule Application

During autoplacing with the Cluster Placer.

Permitted Layers Rule

Definition

Specifies which layers components can be placed on during placement with the Cluster Placer. The Cluster Placer does not change the layer a component is on, you must set the component layer prior to running the placer.

How Duplicate Rule Contentions are Resolved

The rule with the smaller number of layers is obeyed.

Rule Application

During autoplacing with the Cluster Placer.

Room Definition

Definition

Specifies a rectangular region where components are either allowed in, or not allowed in. Rooms can be placed by selecting **Place » Room** from the menus. They can be modified by clicking once to focus, then clicking on a handle to resize.

Components can be moved into their room by clicking on the Arrange Components within Room button on the Component Placement toolbar. Once a component class is assigned to a room all components in that class will move when the room is moved, disable the Room Definition rule to stop this.

How Duplicate Rule Contentions are Resolved

All rules are obeyed.

Rule Application

On-line DRC, Batch DRC and during autoplacing with the Cluster Placer.

Signal Integrity Rule Definitions

To use the signal integrity analysis feature there must be a Signal Stimulus rule and appropriate Supply Nets rules defined, and the Designator Mapping must be configured in the Signal Integrity Tab of the Preferences dialog.

Impedance Constraint

Definition

Specifies the minimum and maximum net impedance allowed. Net impedance is a function of the conductor geometry and conductivity, the surrounding dielectric material (the board base material, multi-layer insulation, solder mask, etc) and the physical geometry of the board (distance to other conductors in the z-plane).

How Duplicate Rule Contentions are Resolved

The rule with the tightest impedance specifications is obeyed.

Rule Application

During Signal Integrity analysis.

Overshoot – Falling Edge

Definition

Specifies the maximum allowable overshoot (ringing below the base value) on the falling edge of the signal.

How Duplicate Rule Contentions are Resolved

The rule with the smallest allowable overshoot is obeyed.

Rule Application

During Signal Integrity analysis.

Overshoot – Rising Edge

Definition

Specifies the maximum allowable overshoot (ringing above the top value) on the rising edge of the signal.

How Duplicate Rule Contentions are Resolved

The rule with the smallest allowable overshoot is obeyed.

Rule Application

During Signal Integrity analysis.

Signal Base Value

Definition

The base value is the voltage that a signal settles to in the low state. Use this rule to specify the maximum allowable base value.

How Duplicate Rule Contentions are Resolved

The rule with the lowest allowable base value is obeyed.

Rule Application

During Signal Integrity analysis.

Signal Flight Time Falling Edge

Definition

Flight time is the signal delay time introduced by the interconnect structure. It is calculated as the time it takes to drive the actual input to the threshold voltage, less the time it would take to drive a reference load (connected directly to the output) to the threshold voltage.

This rule specifies the maximum allowable flight time on signal falling edge.

How Duplicate Rule Contentions are Resolved

The rule with the smallest flight time is obeyed.

Rule Application

During Signal Integrity analysis.

Signal Flight Time Rising Edge

Definition

Flight time is the signal delay time introduced by the interconnect structure. It is calculated as the time it takes to drive the actual input to the threshold voltage, less the time it would take to drive a reference load (connected directly to the output) to the threshold voltage.

This rule specifies the maximum allowable flight time on signal rising edge.

How Duplicate Rule Contentions are Resolved

The rule with the smallest flight time is obeyed.

Rule Application

During Signal Integrity analysis.

Signal Stimulus

Definition

Specifies the characteristics of the stimulus signal that is used in the signal integrity analysis. This is the signal that is injected at each output pin on the net under-test. The worst-case result is returned during design rule checking.

How Duplicate Rule Contentions are Resolved

The first rule in the list is obeyed.

Rule Application

During Signal Integrity analysis.

Signal Top Value

Definition

The top value is the voltage that a signal settles to in the high state. Use this rule to specify the minimum allowable top value.

How Duplicate Rule Contentions are Resolved

The rule with the highest allowable top value is obeyed.

Rule Application

During Signal Integrity analysis.

Slope – Falling Edge

Definition

Falling edge slope is the time it takes for a signal to fall from the threshold voltage (VT), to a valid low (VIL).

This rule specifies the maximum allowable slope time.

How Duplicate Rule Contentions are Resolved

The rule with the minimum slope value is obeyed.

Rule Application

During Signal Integrity analysis.

Slope – Rising Edge

Definition

Rising edge slope is the time it takes for a signal to rise from the threshold voltage (VT), to a valid high (VIH).

This rule specifies the maximum allowable slope time.

How Duplicate Rule Contentions are Resolved

The rule with the minimum slope value is obeyed.

Rule Application

During Signal Integrity analysis.

Undershoot – Falling Edge

Definition

Specifies the maximum allowable undershoot (ringing above the base value) on the falling edge of the signal.

How Duplicate Rule Contentions are Resolved

The rule with the smallest allowable undershoot is obeyed.

Rule Application

During Signal Integrity analysis.

Undershoot – Rising Edge

Definition

Specifies the maximum allowable undershoot (ringing below the top value) on the rising edge of the signal.

How Duplicate Rule Contentions are Resolved

The rule with the smallest allowable undershoot is obeyed.

Rule Application

During Signal Integrity analysis.

Other Rule Definitions

Short Circuit Constraint

Definition

Include this rule to test for short circuits *between* primitive objects on the copper (signal and plane) layers. A short circuit exists when two objects that have different net names touch.

How Duplicate Rule Contentions are Resolved

The rule that does not allow short circuits is obeyed.

Rule Application

On-line and batch DRC.

Un-Connected Pin Constraint

Definition

Detects pins that have no net assigned and no connecting tracks.

How Duplicate Rule Contentions are Resolved

The first instance of the rule is obeyed.

Rule Application

On-line and batch DRC.

Un-Routed Nets Constraint

Definition

The Un-Routed Nets Constraint tests the completion status of each net identified by the scope. If a net is incomplete then each completed section (sub-net) is listed along with the routing completion. The routing completion is defined as the (connections complete)/(total number of connections) x 100.

How Duplicate Rule Contentions are Resolved

The first instance of the rule is obeyed.

Rule Application

On-line and Batch DRC.

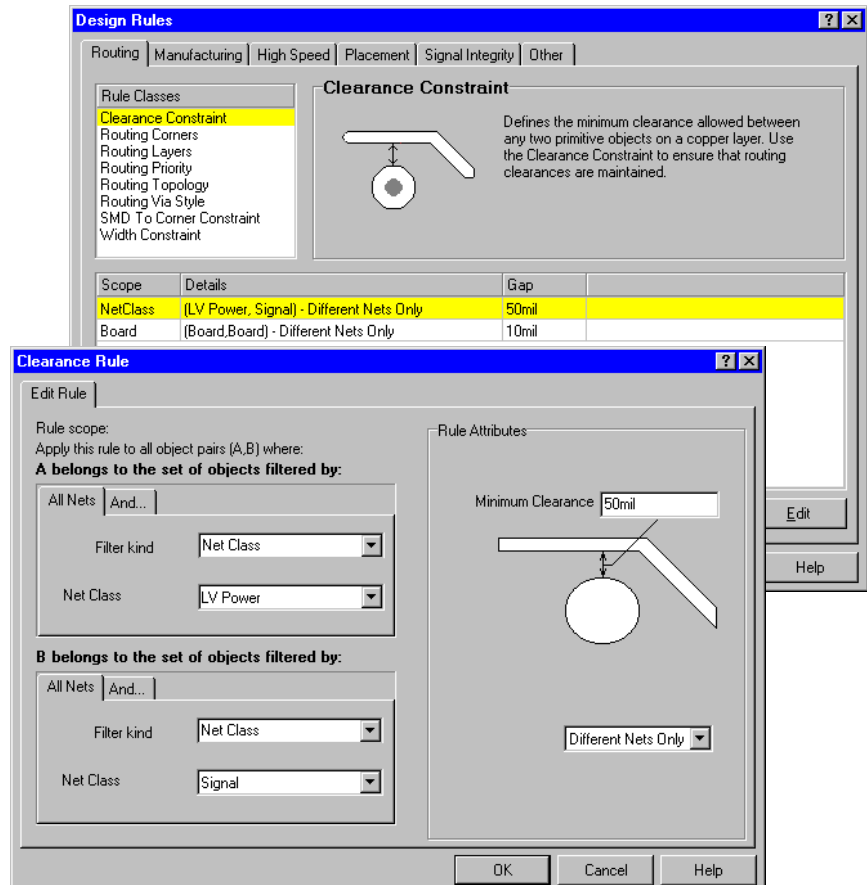
Examples of Using Design Rules

Using the Net Class Rule Scope

Consider the case where you have two groups of nets, signal nets (*Signal* class) and low voltage power nets (*LV Power* class). In this case you want to keep 50 mils *between* the two classes of nets, and 10 mils between any nets *within* a class.

Initially you might think that you could set up one 10 mils clearance rule with rule scopes of Whole Board-to-Whole Board, and a second 50mils clearance rule with rule scopes of Net Class-to-Whole Board. However, this will not work because it will maintain a clearance of 50 mils between all the nets in the *LV Power* class.

To overcome this you need to create two net classes, and setup the 50mils clearance rule with rule scopes of Net Class-to-Net Class. The first 10mils rule with scopes of Whole Board-to-Whole Board will create the 10 mils clearance between each net in the two classes, while the second rule with Net Class-to-Net Class rule scopes will create the clearance between the two net classes. The figure below shows these two rules.



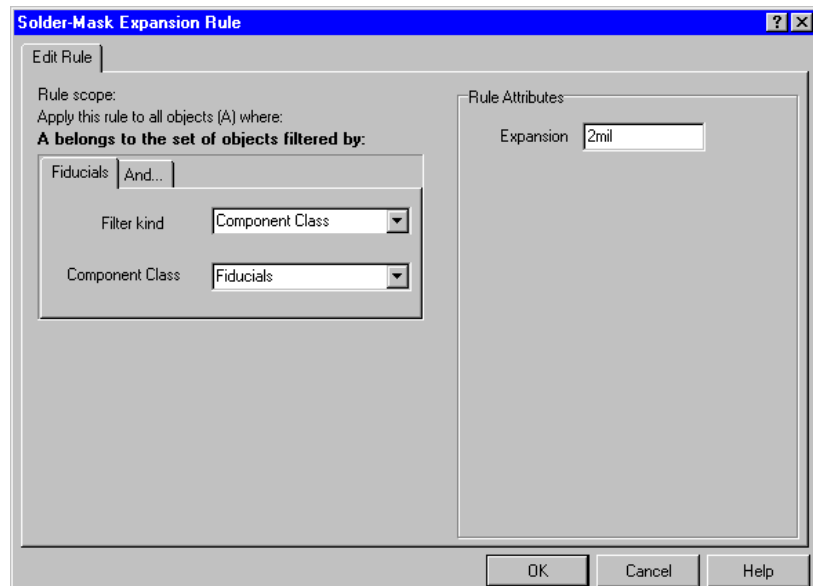
Handling the Mask Expansions for Fiducial Marks

Fiducial marks are copper features used for the optical alignment of a PCB by an automatic assembly machine. It is very important to ensure that the solder mask and paste masks are as per your assembly plant's requirements for fiducial marks.

To use fiducial marks in your design:

1. Create the fiducial mark as a component in the library. Use a single layer pad and give the pad the designator name "FID". Save the component with a name of your choice.
2. Place the fiducial components as required in your design.
3. Create a "Fiducials" component class, that includes all the fiducial components.
4. Add a Solder Mask Expansion rule in the Design Rules dialog. Set the rule scope Filter kind to Component Class, and select Fiducials in the Component Class field. Set the appropriate expansion value in the attributes section.

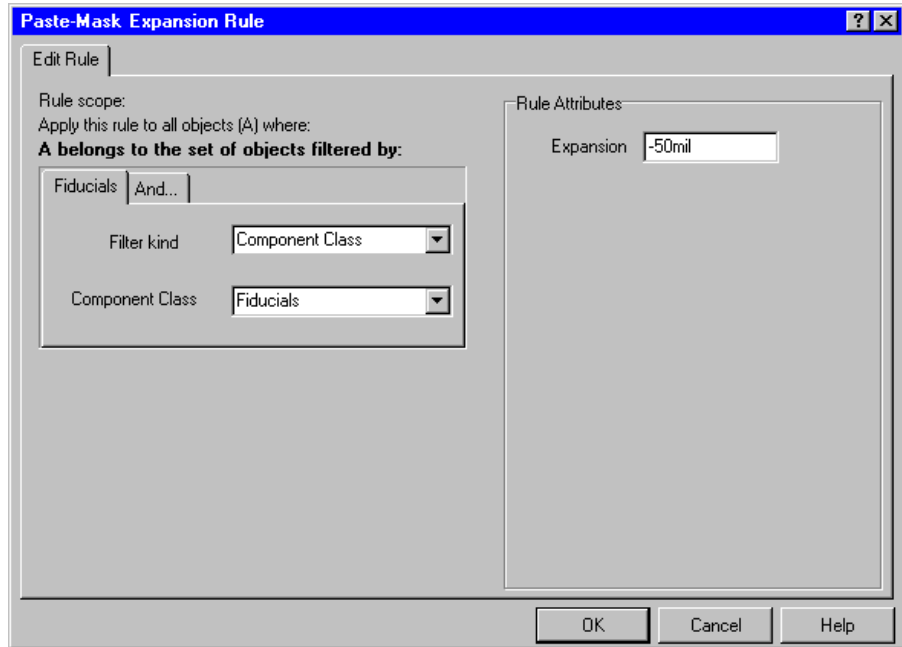
This will ensure that all components in the Fiducials component class will have this expansion applied to the solder mask. If you need a different expansion value for an individual fiducial component, remove this fiducial component from the Fiducials component class and apply another rule to just that fiducial component.



Setting the solder mask expansion for all fiducial pads

You will also need a Paste Mask Expansion rule for fiducials, to ensure that solder paste is not applied to them. To close the opening in the paste mask:

5. Add a Paste Mask Expansion rule in the Design Rules dialog. Using the same approach as before set the rule scope to component class and select the Fiducials class.
6. Set the expansion value to a large negative number, greater than the radius of the largest fiducial used in the design.



Setting the paste mask expansion for all fiducial pads

Using a negative number in the Expansion field instructs the PCB Editor to radially contract the opening in the mask by this amount. As long as you supply a contraction value greater than the radius of the largest fiducial, there will be no openings in the paste mask at the fiducials.

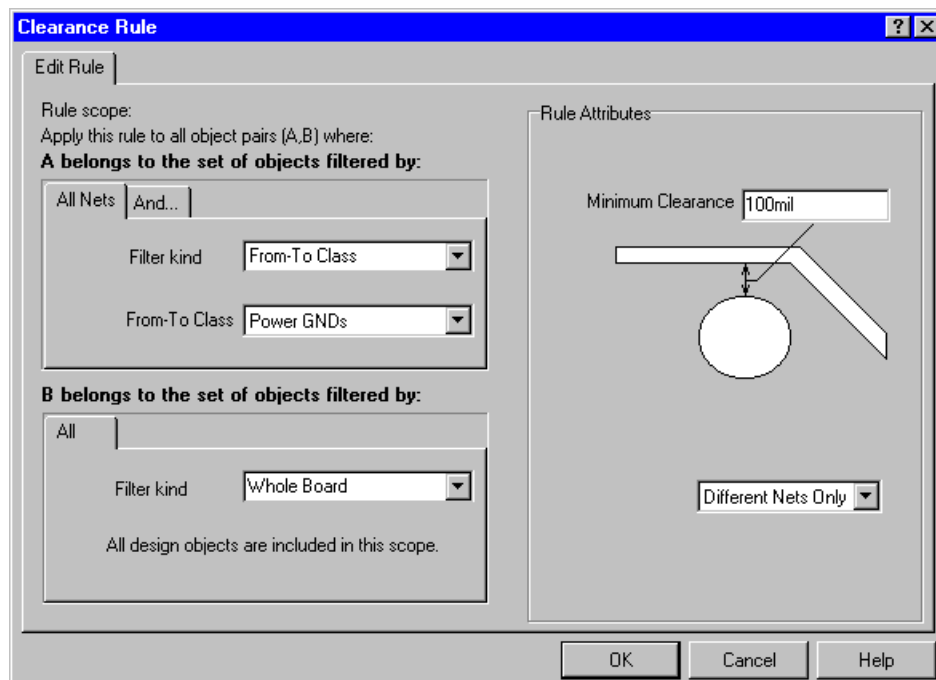
Applying a Clearance Rule to Part of a Net

Design rules can be applied to a particular part of a net.

1. Identify the critical parts of the net by defining From-Tos for those parts of the net.

To create a From-To select the Design-From-To Editor menu item. Define the From-Tos necessary to identify the critical parts of the net. For more information on defining From-Tos, refer to the chapter, *Working With a Netlist*.

If there is more than one From-To required to identify all critical parts of the net, create a From-To Class. Classes are created in the Object Classes dialog, select Design-Classes to pop up this dialog. Once you have identified the critical parts of the net you are ready to add the design rule.



Defining a separate clearance requirement for a critical part of a net

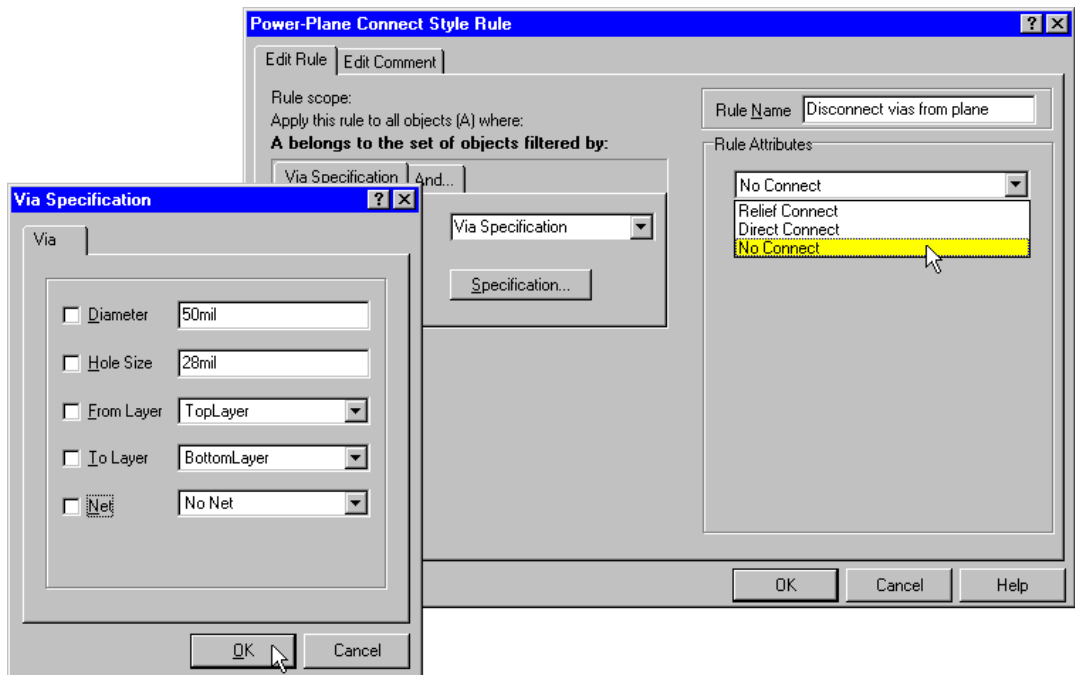
2. Add a Clearance Constraint in the Design Rules dialog.
3. Set the scope Filter kind to From-To, or From-To Class if you created a class.
4. Set the Clearance as required. This specifies the minimum distance allowed between any object in this part of the net, to any other object on the board.

Preventing Vias from Connecting to Plane Layers

Protel 99 SE supports connecting vias to plane layers. When a via is placed and it has a net name the same as the net assigned to a plane layer, it is automatically connected to the plane. You can prevent this behavior by setting up a Power Plane Connect Style design rule.

The rule scope is set to Via Specification, and in the Via Specification dialog all the attributes are disabled – meaning that this rule will apply to all vias of any diameter, with any hole size, from any layer, to any layer, on any net – in other words, all vias.

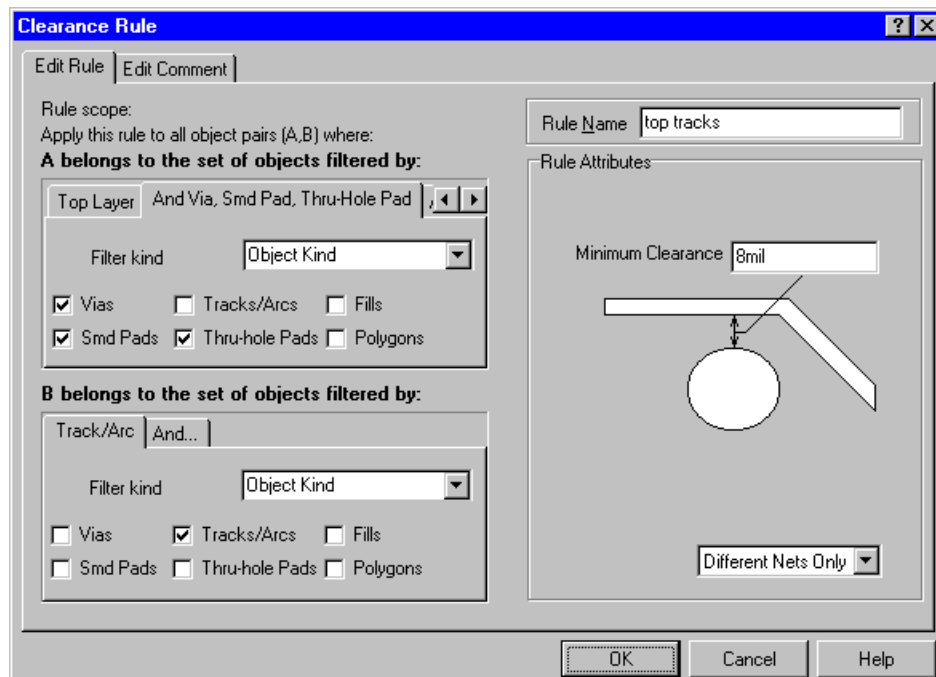
The connection style for the rule is then set to No Connect, meaning any via that satisfies the rule scope must not connect to a plane layer.



Disable all attributes in the via specification dialog for the rule to target no vias

Specifying Different Top Layer Object Clearances

Your design requires that top layer tracks can be 6mils away from each other, but they must be 8 mils away from pads and vias. The 6 mil track-to-track clearance requirement can be covered by the general Board scope clearance constraint. The specific top layer requirement is covered by the rule shown in the figure below.



Using a compound rule scope to set a larger track-to-pad/via clearance

Component Placement Tools and Techniques

Once the board boundaries and keep out requirements have been defined and the netlist successfully loaded into the workspace, the components can be laid out. When the netlist is initially loaded the components are arranged in rows to the right of the board outline (if one exists), or at the center of the workspace (50000, 50000). The components can then be arranged manually, with the aid of the interactive placement tools, via the autoplace tools, or by using a combination of techniques.

In this Handbook, the term *component placement* refers to the process of arranging or positioning components in the workspace.

♦ Good component placement is fundamental to a well designed board. The manufacturability and routability of the board is highly dependent on the component placement.

Important Placement Options

There are a number of options which affect the behavior of the PCB Editor during placement.

Snap to Center (Preferences dialog)

When this is enabled components will be “held” by their reference point during a move, with it disabled they will be held wherever the cursor is clicked. If the component reference point is off-screen when it is selected, the cursor jumps to the reference point if Snap to Center is on, otherwise it pulls the component to the cursor if Snap to Center is off.

Protect Locked Objects (Preferences dialog)

If this option is enabled you can not move components that have their Locked attribute enabled, any they are ignored during multiple-component moves. If the option is not enabled you will be prompted to confirm moving a locked object.

Rotation Step (Preferences dialog)

Amount a component (or selection of components) will rotate when the SPACEBAR is pressed during a component move.

Draft Thresholds (Preferences dialog)

The Strings threshold determines at what zoom level the component designators will change from text to an outline rectangle. To display designators as text when zoomed further out, set this to a smaller number.

Component X and Y Grids (Document Options dialog)

These grids define the points in the workspace that the component reference point will snap to as the component is moved.

Locking Components

Placement-critical components, such as edge connectors, can be locked in place. To lock a component in place, double-click on the component to pop up the Component dialog, then enable the Locked check box in the Attributes Tab.

Component Clearances

Set up suitable Component Clearance Constraint design rules (Placement Tab of the Design Rules dialog). This rule is obeyed by both on-line and batch DRC, enable these in the Design Rule Check dialog (**Tools » Design Rule Check**).

Moving Components

To arrange the components manually, select **Edit » Move » Component** from the menus. The Status Bar will prompt “Select Component”. Click on the component you wish to move. The component will float on the cursor and can now be positioned on the board. You can also click and hold to move a component.

Rotating and Flipping Components to the Other Layer

Components can be rotated in a number of ways. They can be rotated when they are floating on the cursor. Press the SPACEBAR to rotate anti-clockwise, hold SHIFT while pressing the SPACEBAR to rotate clockwise. The angle the component rotates is specified in the Options Tab of the Preferences dialog.

To rotate a group of components, first select the components. The selection can be rotated by choosing **Edit » Move » Rotate Selection** from the menus. This will first prompt for a rotation angle, and then for a Reference Point about which to rotate the selection. The selection can also be rotated by selecting the Move Selection process launcher and then rotated with the SPACEBAR.

To flip a component so that it can be placed on the bottom of the board, press the L shortcut key while the component is floating on the cursor. To flip a placed component, double-click to edit the component and change the Layer attribute.

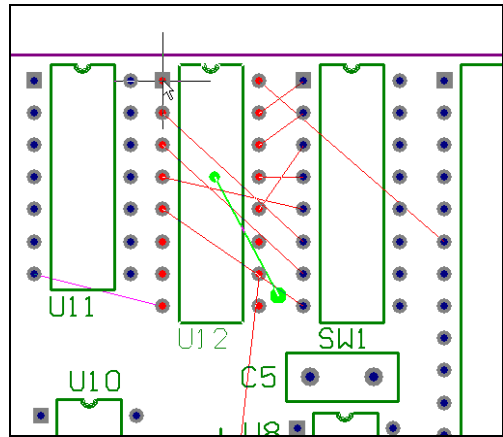
Dynamic Reconnect with Smart Connection Line Display

Your basic guide to selecting suitable positions for each component are the connection lines from the moving component to the other components on the board. Ideally each component is placed to minimize the overall length of the connection lines, helping minimize the finished routing paths.

However, on a dense board it can be difficult to make sense of the maze of connection lines, which are often referred to as ‘the ratsnest’. The other important fact to be aware of is that as you move a component around the board, the connection lines may no

longer accurately reflect the connectivity, it could be that the connection lines could connect to other pins on the same net that are closer to the new component location.

Protel 99 SE includes a dynamic reconnect feature with smart connection-line display. When you move a component, a group of selected components or a union of components, all connection lines are temporarily hidden except for those that connect from a moving component to a component on the board. These nets are analyzed and reconnected as you move the components across the board, making it easy to select the appropriate placement location.



Connection lines that are part of the move are automatically displayed

With this feature it is easier to hide all the connection lines, then as you move the components the appropriate connection lines are automatically displayed and updated. Connection lines can be hidden by selecting **View » Connections » Hide All** from the menus.

Note that you can also temporarily hide all the connection lines during component moves (including those normally displayed by the smart connection-line display feature) by pressing the **N** shortcut key. Doing this temporarily disables the connectivity analyzer.

As you move components the new dynamic connection length analyzer continually assesses placement quality based on connection lengths, and displays a green (strong) or red (weak) vector indicating current placement quality. The far end of the vector indicates a location for the component(s) that would minimize overall connection lengths.

- ◆ Select **View » Connections » Hide All** from the menus to hide all connection lines.
- ◆ During manual placement the netlist is automatically re-optimized as the component is moved, updating the connection lines to the current component position. If you are moving a component with a large number of pins (over 500) and you find the update too slow, you can temporarily disable the re-optimization by hiding the connection lines – press the **N** shortcut key to do this.

Working Between the Schematic and PCB

Transferring the design from the schematic editor to the PCB editor is more than a process of transferring component and connectivity information. There is also structural information about the design embodied in the schematic – a multi-sheet schematic breaks the design into logical groups of components, and the components are often arranged on these sheets in a manner that reflects how closely they connect to each other. This information makes the schematic an ideal platform from which to guide the PCB placement process. Protel 99 SE incorporates features that help you transfer these design requirements from the schematic to the PCB.

Creating Classes and Rooms from the Schematic

On many designs the way the components are grouped on the schematic sheets directly reflects the way they should be grouped on the PCB. Protel 99 SE includes features that help you transfer this grouping information from the schematic to the PCB.

PCB component classes can automatically be created from each schematic sheet when the design is transferred from the Schematic Editor to the PCB Editor. To do this enable the Generate Component Classes and Placement Rooms for all Schematic Sheets in Project option in the Update Design dialog (select **Update PCB** from the Schematic Editor menus).

A PCB component class is created for each sheet in the schematic, the class includes all the components on that schematic sheet. Each component class is given the same name as the schematic sheet it is created from, with any spaces removed. Multipart components that span more than one sheet are included in the class of the sheet that contains the first part of the component. A placement room is also created for each component class. For information on working with placement rooms refer to the topic *Working with Placement Rooms* later in this chapter.

Selecting PCB Components from the Schematic

Another feature that helps you transfer component relationship information from the schematic to the PCB is the ability to directly select PCB components from the schematic. To do this select the components on the schematic sheet, then select **Tools » Select PCB Components** from the Schematic Editor menus. These components will be selected on the PCB, and the PCB view zoomed to show the selection. With this set of selected components you can now create a component class, or a component union. Refer to the *PCB Selection* topic later in this supplement for details on how to create a component class from a selection. Refer to the following topic for information on creating component unions. You can also use the Arrange Within Rectangle feature to quickly pull the selected components out of a large group of components and arrange them in a group. Refer to the topic *Using the Interactive Placement Tools* later in *PCB Placement* section for information on how to do this.

Cross Probing from the Schematic to the PCB

You can also move from an individual schematic component to the corresponding PCB component using the Cross Probe feature. Click on the Cross Probe button on the main toolbar, then click on the component on the schematic sheet. The PCB component will be located and centered in the window. You can also cross probe from a component pin, and from a net identifier, as well as perform the same cross probe options from the PCB back to the schematic.



Using Component Unions

Create component Unions – unions are sets of components that you want to work with as a block. Components in a union maintain their relative positions within the union as they are moved, making it a valuable placement aid.

Creating a Component Union

To create a union select the components that you want in the union and click on the Create Union button on the Component Placement toolbar. Note that a component can only belong to one Union at a time.



Removing Components from a Union

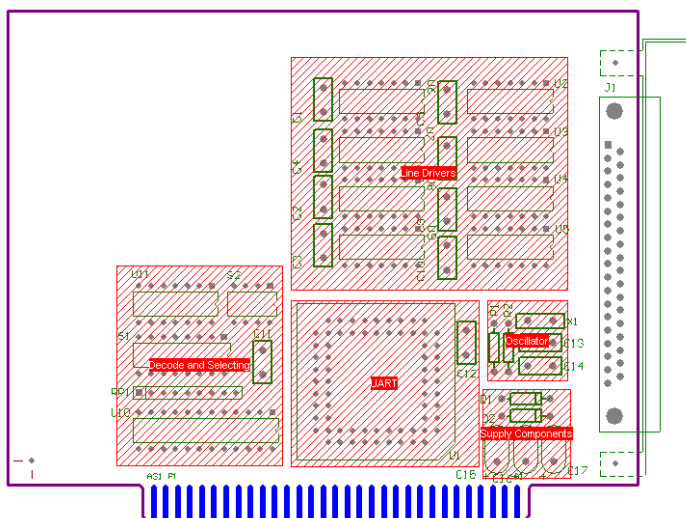
Unions can be broken by clicking on the Break Union button on the Component Placement toolbar, then clicking on one of the components in the union. Select the component(s) that you want to remove from the union in the Confirm Break Component Union dialog and click OK.



Breaking a Union

Select **Tools » Convert » Break All Component Unions** to break all current component unions.

Working with Placement Rooms



Create placement rooms to help with the placement process

Placement rooms are rectangular regions that assist in the placement of components. Components are assigned to rooms and can be automatically moved into their room, they also move with the room whenever the room is moved.

Room Definitions are part of the design rule system, and compliance with room definition rules can be checked by the on-line and batch DRC. Rooms can also be used by the cluster-based autoplacer, by selectively locking components you can also autoplacer on a room-by-room basis.

Creating Placement Rooms

Rooms can be placed from the **Place** menu, through the Placement Tab of the Design Rules dialog, or the Place Room button on the PlacementTools toolbar. Rooms can exist on the top or bottom layers, they are placed on the top layer by default. Once a room has been placed its properties can be defined by double-clicking on the room.



Assigning Components to a Room

The scope of the Room Definition design rule defines the set of components that are assigned to that room. Edit the rule to set the scope and assign the components to the room.

Positioning and Sizing Rooms

A room can be modified graphically at any time, click once to focus it, then click on a handle to resize it. The entire room can be moved by clicking and dragging.

Note that when a room is moved the components assigned to the room are automatically moved with it. To prevent this and move a room without moving the components, disable the Room Definition rule in the Design Rules dialog (disable a rule by unchecking the Enabled option). Rooms can also be locked in the Room Definition dialog, lock a room to prevent accidentally moving it.

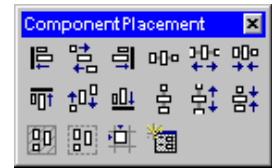
Placing within Rooms

You can quickly pull all the components that are assigned to a room into that room by clicking the Arrange Within Room button on the Component Placement toolbar.



Using the Interactive Placement Tools

The Component Placement toolbar includes a number of alignment and spacing tools to help you quickly arrange the components.



Alignment Tools

Align left edge of selected components

Using the left edge of the left-most component as a reference, slides the selected components to the left, packed as tightly as the component clearance rule allows.

Align right edge of selected components

Using the right edge of the right-most component as a reference, slides the selected components to the right, packed as tightly as the component clearance rule allows.

Align vertical centers of selected components

Places selected components in a single column, aligned by their vertical centers. After clicking the button you are prompted to select a reference component, the other selected components are placed above this component.

Align top edge of selected components

Using the top edge of the top-most component as a reference, slides the selected components up, packed as tightly as the component clearance rule allows.

Align bottom edge of selected components

Using the bottom edge of the bottom-most component as a reference, slides the selected components down, packed as tightly as the component clearance rule allows.

Align horizontal centers of selected components

Places selected components in a single row, aligned by their horizontal centers. After clicking the button you are prompted to select a reference component, the other selected components are placed to the right of this component.

Spacing Tools

Make horizontal spacing equal for selected components

Distributes selected components equally between the left-most and right-most components in the selection. Their vertical position is not changed.

Increase horizontal spacing for selected components

The horizontal distance between the component reference points is increased by the amount specified in the X component placement grid.

Decrease horizontal spacing of selected components

The horizontal distance between the component reference points is decreased by the amount specified in the X component placement grid.

Make vertical spacing equal for selected components

Distributes selected components equally between the top-most and bottom-most components in the selection. Their horizontal position is not changed.

Increase vertical spacing of selected components

The vertical distance between the component reference points is increased by the amount specified in the Y component placement grid.

Decrease vertical spacing of selected components

The vertical distance between the component reference points is decreased by the amount specified in the Y component placement grid.

Moving Tools

Move to Rectangle

This tool arranges the selected components within the defined rectangle. Click on the button, then click once to define the first corner of the rectangle, move the mouse and click a second time to define the opposite corner of the rectangle. The selected components are arranged within the rectangle, working across and down.

Move to Room

This tool arranges the components that are assigned to a room within that room. Click on the button, then click on the room – the components associated with that room by the rule scope are arranged within the room, working across and down.

Move to Grid

Move the components to the nearest point on the component placement grid. The component X and Y placement grids are defined in the Options Tab of the Document Options dialog. Locked components are not moved.

Automatic Component Placement

Placement is a critical process in the design phase of a printed circuit board. The effort to route your design, and its ultimate manufacturing cost, are highly dependent on placement quality.

Placement involves specifying the exact location and orientation of each component on the PCB, given the board size and the area of the board on which placement is permissible. The most important objective in placement, after the fitting of all the components on the board without violating any design rules, is to make the job of routing easier, or in certain cases, possible.

In short, the main goals in placement of a PCB are :

- Fitting all the components on the board.
- Avoiding design rule violations.
- Placing components to allow routing completion.
- Meeting any requirements for board assembly and board testing. It is advisable to always consult with your board assembler during the design phase to ensure that their requirements are met.

Selecting an Autoplacer

Protel 99 SE's PCB Editor has access to two autoplacement tools:

- The Cluster Placer groups components into clusters based on their connectivity, and then places these clusters geometrically. The algorithms in this autoplacer are more suited to designs with a lower component count (less than 100).
- The Global Placer uses a statistical algorithm to place the components in an attempt to minimize the connection lengths. As it uses a statistical algorithm it is best suited to designs with more than 100 components.

Board Area For Placement

Define the board area available for component placement prior to running the Global Placer Server. This is done by placing tracks on the keep-out layer to define a boundary within which all components are to be placed.

To keep certain regions of the board free of components, create keep-out zones. Place tracks, fills, arcs and polygons on the Keep Out layer to create these keep-out zones. Refer to the chapter, *Defining the Board*, for further information.

♦ Before performing an autoplacement, set the *current origin* back to the *absolute origin* by selecting the **Edit » Origin » Reset** menu item. This can be important because the autoplacement routines use the absolute origin as a reference point and may place components off grid relative to the current (relative) origin you created.

Setting up and Running the Cluster Placer

The Cluster Placer follows the Placement rules defined in the Design Rules dialog. The rules include; component clearances, component rotation rule, nets to ignore rule, and the permitted layers rule. The rules are explained in the chapter, *Specifying the PCB Design Requirements*.

Once the rules are defined, select **Tools » Auto Place** from the menus to pop up the Auto Place dialog. Click on the option to enable the Cluster Placer, then click OK to start the autoplacement process.

Setting up the Global Placer

The Global Placer is easy to use, despite the underlying complexity. All you need to know are some very basic facts about the design. No special configuration is required, as the system has been finely tuned over a wide spectrum of boards for optimal performance.

To set up the Global Placer, select the **Tools » Auto Place** menu item to pop up the Auto Place dialog. Click on the option to enable the Global Placer, then set up the Global Placer options, as described below.

Group Components

If this option is enabled the Global Placer will go through all the components on the board and group together those which are tightly connected. The main criteria for grouping is the number of connections between components. The weight given to this criteria is influenced by the number of pins these components have.

The system then performs a relative placement on each group. These groups are treated as “super components” and their relative placement is kept untouched during the main placement cycle.

Although this option is generally useful, it can be counter-productive if there is not going to be enough space on the board. This is because the relative placement within each group is not changed during the main placement cycle and space might be wasted in order to accommodate the group.

Rotate Components

If component rotation is allowed, components will be rotated in order to find an optimal orientation. Four orientations are considered, 90, 180, 270 and 360 degrees.

This option must be used with caution, as footprint rotation can have a direct impact on the manufacturability of the design. For example, some pick-and-place equipment cannot handle rotated components. Enabling Rotate Components also makes the placement job more difficult and time consuming – particularly if the design is very dense. Given that, it can also produce a superior result, so some degree of trial-and-error is normally required.

The PCB Editor also allows you to freely rotate components after placement to 0.001 degree accuracy, using the **Edit » Move**, **Edit » Paste Array** or the Rotation field in the Change Component dialog.

Automatic PCB Update

The Global Placer will automatically pass the current component positions to the board in the PCB editing window each time the optimization status is updated (approximately every 10 seconds). You can also manually update at any time while the Placer is running by selecting **File » Update PCB**.

Placement Grid

This is the grid each component reference point will be placed on. It is typically set to a fraction of the common component pin pitch, and/or a multiple of the intended routing grid. The PCB Editor includes a tool to move all components to a new grid if the component placement grid needs to be altered at a later stage (**Tools » Interactive Placement » Move To Grid**).

Power Nets

The Power Nets option performs two functions:

1. Nets that are specified in the Power Nets fields are no longer considered by the placement algorithm, which can greatly speed the placement process.
2. To associate a bypass capacitor with each large component, specify the names of the nets which the capacitors are across. For example, if your design uses the nets VCC and GND as the power nets, enter VCC in the first field and GND in the second. The Global Placer will attempt to associate a two pin component that is connected across the specified power nets (VCC and GND) with each large component (14 pins or more).

◆ More than one power net can be specified in each text box. Separate net names by a single blank space (total of 28 characters per line maximum).

Running the Global Placer

Once you have configured the Global Placer options, press the OK button to start the placement process.

The Global Placer Window

The Global Placer Server displays the placement progress in its own window. This includes the components and the keep-out areas.

The main menu for the Global Placer is very short. Options include: File, View, Window and Help. As the Global Placer has its own data structure the PCB database is not changed during the placement process. The **File » Update PCB** menu item can be used to pass the current placement back to the PCB Editor. This allows you to periodically switch back to the PCB editing window to check the placement quality and use the Density Map tool to examine the “routability” of the design.

The placement window has its own Status Bar, within which the following information is available:

Elapsed time

Time since start of placement.

Optimization

There are a total of 70 cycles in any placement task. The first 40-50 cycles are very fast as most moves are accepted. However, as the “temperature” decreases, more and more moves are made in order to satisfy the requirements of a cycle. This means the cycles get slower towards the completion. Optimization refers to the percentage of completion toward an “ideal” set of costs. Refer to the *Theory* topic at the end of this chapter for a discussion of optimization.

Number of Moves

The Number of Moves, displayed in the status bar, is the total number of times that the system has moved a component to a new position in order to improve the routability of the board.

◆ During the placement process small purple squares will appear on the board. The size of these squares reflect the connection density in that region.

Placement Results

The placement process will make smaller and smaller moves as it progresses towards its optimal solution. It is not necessary to run it to completion. Select **File » Close** to terminate the placement process. You will be asked if you wish to update the PCB before closing the placement window.

Tips for Better Autoplacement Results

Pre-Placing Components

You can pre-place any component before running one of the Autoplacers. Enable the Locked attribute in the Component dialog to prevent these components from being moved.

Apart from those components which have to be placed in certain locations on the board, such as edge connectors, heat sinks, or a group of analog components, it can be useful to pre-place components which need no restriction on their placement. For example, it might be desirable to pre-place the memory chips. This could facilitate the placement of the other components.

Use of Keep-Out Zones

To keep certain regions of the board free of components, create keep-out zones. These could be placed next to connectors, or in regions which must be kept clear for mechanical reasons. Place tracks, fills, arcs and polygons on the Keep Out layer to create these keep-out zones.

Autoplace and Larger Nets

Large nets can affect the speed and quality of the Autoplacers. The reason for this is that the computation involved with rearranging a net is exponentially proportional to the net size. An interesting observation is that large nets, such as power and ground, can play an insignificant role in the overall placement process. Therefore, it can help the placement process by instructing the Autoplacer to ignore these large nets.

To do this for the Cluster Placer you need to define a Nets to Ignore Design Rule in the Placement Tab of the Design Rules dialog.

To do this for the Global Placer you must specify the nets in the Power Nets region of the Auto Place dialog.

- ◆ Remember, automatic component placement is a productivity tool – not a replacement for the judgment and experience of the designer. A little guidance from the designer – for example, pre-placing important components and locking them, and realistic placement grids and clearances, can all go a long way toward ensuring that the Autoplacer will both speed up and ease the design process.

Post Autoplacement Tools

The outcome of the Global Placer is a board in which the relative positions of the components are optimal. Due to its global nature, the Global Placer often produces boards which are not entirely “polished”. For instance, there could still be some overlaps after the placement is completed, or some components might not be aligned properly. The interactive placement tools are specifically designed to facilitate the process of tidying up the placement. Refer to the *Interactive Placement* topics earlier in this chapter for information on using the interactive placement tools.

Understanding Connectivity and Topology

Electronic design tools provide the platform on which an electronic design can move from the conceptual stage in the designer's mind, right through to producing the files required to manufacture the printed circuit board. By capturing the design electronically, the power and efficiency of a computer can be harnessed to transfer the design rapidly and accurately through the various design phases.

Through these phases the design is held in a number of forms. It begins as a schematic, a collection of components which are “wired” together. It ends as a printed circuit board, manufactured from a set of files produced from the PCB design tool.

The essence of the design in each of these phases is the information about the components, and the information about how the components are wired (connected).

To transfer the design from the schematic editing phase to the PCB layout phase there is a mechanism that extracts the component and connective information from the schematic, and passes it into the PCB workspace. The component information that is passed includes; the designator, the value, and the physical component package (footprint). The connective information is transferred as a set of *nets*, where each net is a list of component pins that are electrically connected.

In the PCB Editor you translate this component and connective information into a “physical” layout, with “physical” connectivity. This includes; placing the components to meet any mechanical constraints, and translating each logical net connection into a physical connection, fulfilling the electrical parameters required for that connection.

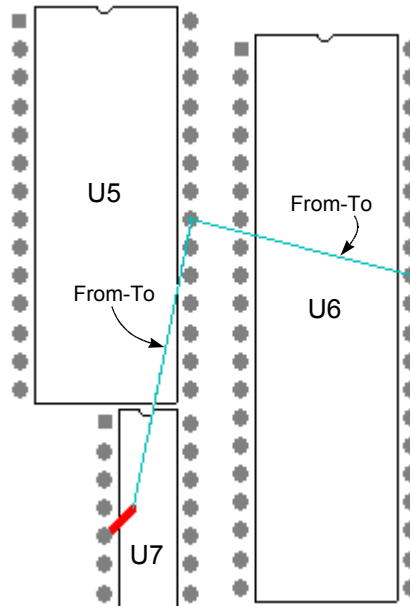
How the Component and Connectivity Information is Transferred

Protel 99 SE includes a powerful design synchronization tool, that makes it very easy to transfer design information from the schematic to the PCB, and back again. The synchronizer analyzes both the schematic and PCB, and identifies any differences.

When you run the synchronizer you nominate which is the target document by selecting either **Update PCB**, or **Update Schematic**, from the menus. This can be done from either the PCB Editor or the Schematic Editor. When you do this the synchronizer attempts to update the design data in the target document, to match the data in the source document. It produces a Synchronization report, which details each change, as well as listing any problems that were encountered.

Refer to the chapter, *Passing Design Information between the Schematic and the PCB*, for information on using the Synchronizer.

How the Net Connectivity is Displayed



A net with one connection unrouted and the other partially routed.

When the Synchronizer loads components and connective information into the PCB workspace, the PCB Editor displays the pin-to-pin connections in each net as a series of thin lines. The line that connects each pin in the net to another pin in the net is called a *From-To* – going *From* one pin in the net *To* another pin. The complete set of From-Tos are commonly referred to as the *Ratsnest*.

The pattern or arrangement of the From-Tos in a net is called the *net topology*. If a net has not been assigned a user-defined topology, then the PCB Editor arranges the From-Tos to give the shortest possible connection distances for the entire net, based on the current arrangement of the components.

If the net has a topology applied, the From-Tos are added to maintain the topology. When you route a net that has a topology applied, the From-To is shown as a dotted line, indicating that the net should be routed between these two points to maintain the topology.

A topology is applied to a net either through a Topology Rule, or by defining fixed From-Tos. There is more information on net topology and fixed From-Tos later in this topic. For more information on the Routing Topology design rule refer to the chapter, *Specifying the PCB Design Requirements*.

Net Topology

When the components and connective information has loaded, the pin-to-pin connections are displayed for each net. The arrangement, or pattern of the pin-to-pin connections is called the net topology. By default, the PCB Editor arranges the pin-to-pin connections of each net to give the shortest overall connection length (this topology is called Shortest). A different topology can then be applied to a net.

The topology of a net can be re-defined for a variety of reasons. High speed designs require that signal reflections must be minimized. To achieve this the high speed nets are arranged with a daisy chain topology, where all the pins are connected one after the other, with the source pin at one end and a terminator pin at the other end of the chain. Another requirement of your design may be that all ground pins in the ground net connect back to a common point. A star topology could be applied to the ground net to ensure this.

Changing the Net Topology

There are two techniques to apply a particular topology to a net in the PCB Editor. The first is at the net level and the second is at the individual pin-to-pin connection, or From-To level. If your design contains nets that require a certain topology, apply a Routing Topology design rule to those particular nets. Refer to the chapter, *Using Design Rules*, for further information about the Routing Topology design rule. If you wish to manually specify part or all of the topology of a particular net, then define From-Tos for that part of the net you wish to control the topology of.

User-defined From-Tos

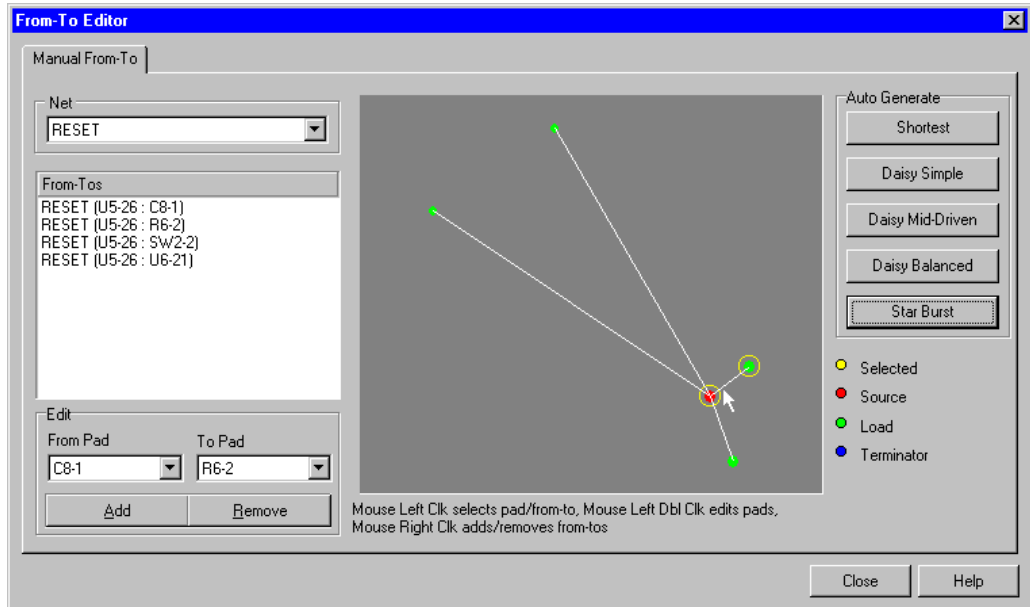
To give you total control of the arrangement, or pattern, of the pin-to-pin connections in a net, the PCB Editor allows you to define your own set of From-Tos. A From-To instructs the PCB Editor, “I want to connect *From* this pin *To* that pin”.

You can define one From-To for a net, a few From-Tos for a critical part of the net, or specify the entire topology of the net by defining From-Tos for all the pin-to-pin connections. If you create From-Tos for only part of a net the PCB Editor will set the remaining pin-to-pin connections to the shortest topology.

As well as using From-Tos to create a specific net topology, From-Tos can also be used as the scope for a design rule. The scope of a design rule designates exactly what the rule is to apply to. Using a From-To as the scope allows you to apply a rule to an individual pin-to-pin connection. This gives you total control over how rules apply to a net. You could specify that a net be routed at 25 mils, except for one From-To which you want to have routed at 40 mils. Refer to the chapter, *Using Design Rules*, for more information about design rules and their scope.

Creating From-Tos

To specify From-Tos for a net select the Design-From-To Editor menu item. The From-To Editor will be displayed.



Create and remove From-Tos in the From-To Editor.

Select the Net you wish to specify From-Tos for. All the pins in this net are displayed in the graphical window to the right. Any existing From-Tos are displayed as a thin line connecting the two pins in the graphical window, and they are also listed below the net name. Below the graphical window there are tips on how to quickly add and remove From-Tos.

Auto-Generated From-Tos

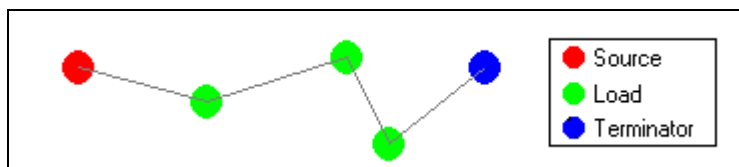
To quickly create a set of From-Tos for the entire net, use the Auto-Generate buttons at the bottom of the From-To Editor. These buttons create a set of From-Tos for the entire net, arranged in that particular topology.

Shortest

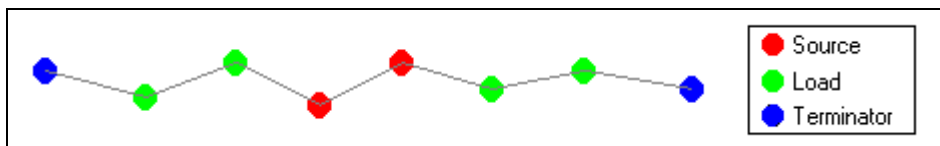
By default, the PCB Editor arranges the pin-to-pin connections in the net to give the shortest overall connection distance. Pressing this button will remove any user or auto-generated From-Tos, instructing the PCB Editor to arrange the pin-to-pin connections with the shortest topology.

Daisy-Simple

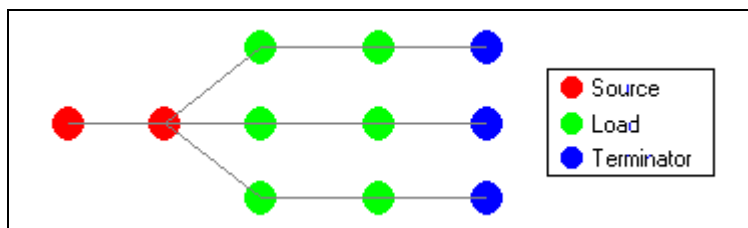
In a simple daisy chain topology all the nodes (pins) are chained together, one after the other. The order they are chained is calculated to give the shortest overall length. If a source and terminator pad are specified then all other pads are chained between them to give the shortest possible length. Edit the pad (double-click on it) to set it to be a source or terminator. If multiple sources (or terminators) are specified they are chained together at each end.

**Daisy-Mid Driven**

In a mid driven daisy chain topology the source node(s) are placed in the center of the daisy chain and the loads are divided equally and chained off either side of the source(s). Two terminators are required, one for each end. Multiple source nodes are chained together in the center. If there are not exactly two terminators a simple daisy topology is used.

**Daisy-Balanced**

In a balanced daisy chain topology all the loads are divided into equal chains, the total number of chains equal to the number of terminators. These chains then connect to the source in a star pattern. Multiple source nodes are chained together.

**Star**

This topology connects each node directly to the source node. If terminators are present they are connected after each load node. Multiple source nodes are chained together, as in the balanced daisy topology.

Displaying Pin-to-Pin Connections

To make working with the ratsnest more manageable, the PCB Editor allows you to selectively show and hide the pin-to-pin connection lines. Select the **View » Connections** sub-menu;

Net

Show/Hide the entire set of pin-to-pin connections for the selected net. When you choose this option, a cross hair cursor appears. If you know the location of a pad on the net, click on that pad. If you do not, click in free space and a dialog will pop up, prompting for the net name. If you are unsure of the net name type ? and click OK to list all loaded nets.

Component Nets

Show/Hide the entire set of pin-to-pin connections for all nets which connect to the selected component.

All

Show/Hide all currently loaded (unrouted) connections.

◆ If your design includes a large net with many nodes, or there are a large number of primitives (tracks, vias, etc) on the net, you may find that the automatic connectivity monitoring will take a long time to analyze this net. To disable the connectivity monitoring for a particular net you can “hide” the net. When you select **View » Connections » Hide Net** to hide the net connection lines, it is also hidden from the connectivity checker.

Managing the Netlist

During the course of the design process there may be times when you need to make changes to the netlist. Netlist changes are made in the Netlist Manager dialog – select **Design » Netlist Manager** to display the dialog. Click the Menu button to display a menu of options available in the Netlist Manager.

Adding and Removing Pads from a Net

To add or remove pads from a net, locate the net in the Nets in Class column, then double-click on it to open the Edit Net dialog. Use the buttons in the center of the dialog to add or remove pads from this net.

Update Free Primitives From Component Pads

Use this option when you have performed design changes on the schematic that change the netlist. When you transfer the changes to the PCB, DRC violation may occur if the routing net names no longer match the pad net names. When you select this option the net attribute for each primitive is set back to no-net, then starting at each pad, the routing is traced and the pad net name applied to all the routing primitives.

Note that this process is performed starting from every pad, if there is routing which joins 2 pads that belong to different nets then the last pad that is analyzed will have its net name applied to the routing. To control this behavior, delete a track segment to break the routing from any pads that the net should not connect to.

Export Netlist from PCB

Export the internal netlist from the PCB to a netlist file. The internal netlist is the list of nets displayed in the Netlist Manager dialog.

Create Netlist from Connected Copper

Analyze the connectivity created by the routing (copper) and create a netlist listing this connectivity. If a net name is found on a pad this is used to name that net.

Compare Netlists

You will be prompted to select a netlist, then prompted again to select a second netlist. The 2 netlists are compared, and nets are matched based on them having common nodes. The report file details any differences found in the 2 netlists.

Compare Netlist File to Board

Performs a netlist compare on the internal PCB netlist against a user-selected netlist. After performing a DRC to ensure that the routing matches the internal netlist, use this option to confirm that the PCB matches the original schematic netlist.

Changing Net Attributes

Like pads, tracks, vias and other design objects, each net has a set of attributes. To edit the attributes of a net set the Browse mode of the PCB Panel to Nets, select the net and press the Edit button. The Change Net dialog will pop up, allowing you to edit the attributes of the selected net. The editable attributes include:

Color – Each net will be assigned the default net color when the netlist is loaded. The color for this net can be changed here.

Hide – Selectively hide the connection lines for this net, or combine with a global edit to hide the connection lines for a number of nets.

- ◆ The Layers Tab of the Document Options dialog includes a check box to turn the Connect layer on or off. If this is off no connections are shown, regardless of the Hide attribute of each net.

Identifying Nets

Comprehensive information about each object on the board, including the physical parameters, and the net name, is displayed on the Status Bar when the cursor is positioned over the object. This Status Info can be disabled in the Display Tab of the Preferences dialog.

Manually Routing the PCB

Routing your design is the process of translating the logical connections into physical connections. These physical connections can include; tracks, vias, pads, arcs, fills, polygons and power planes. Typically, the majority of physical connections are created with tracks and vias.

Protel 99 SE's PCB Editor includes features specifically tailored to speed this process of translating a logical connection into a physical connection. These features include;

Intelligent manual routing

Place the tracks to create the connections where you choose, you do not have to route the connections along the path shown by the connection lines. Route to a different pin on the net, or create a T-Junction. When you terminate a track the net is analyzed and connection lines are added and removed as required.

Electrical grid

To ease the accurate placement of electrical objects such as tracks and vias, the PCB Editor includes an electrical grid. The electrical grid defines a range within which a moving electrical object (such as a track, pad or via) will attract to another electrical object. The electrical grid overrides the snap grid, allowing you to easily connect to an off grid object.

Violation-free object placement

The PCB Editor includes a routing mode where you can only place primitives such that they do not violate any clearance design rules. This feature allows you to route hard up against existing objects, without fear of violating any clearance rules.

On-line Design Rules

Most of the design rules can be monitored as you route. Enable the rules that you want monitored during routing in the On-line Tab of the Design Rule Check dialog (select **Tools » Design Rule Check**). Violations are flagged immediately.

Automatic Loop Removal

Existing tracks can be quickly re-routed. Simply route new track segments and the redundant segments are automatically removed.

Multiple track placement modes with look-ahead

The track placement mode defines the way track corners are placed, and includes arcs and 45 degree tracks. Each mode includes a look-ahead segment which you can use to predict the placement of the next segment and accurately terminate the current segment.

Automatic Via Insertion

Toggle to another copper layer while routing by pressing the *, + or – shortcut keys. A via is inserted automatically.

How the PCB Editor Manages the Connectivity as you Route

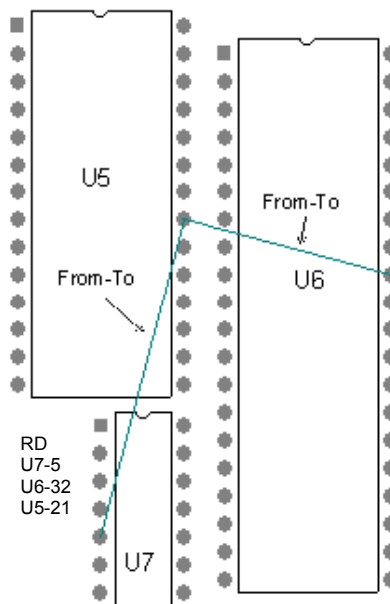
Protel 99 SE's PCB Editor is a connectivity driven design environment. At all stages of routing your design the PCB Editor monitors and manages the netlist connectivity.

Consider the net RD shown in the figure above. RD is the net name given to the connections between pin 5 of U7, pin 32 of U6 and pin 21 of U5. On the schematic this connectivity is represented with wires which join these three pins together. When the netlist is loaded into the PCB workspace the PCB Editor analyzes the nets, and creates two From-Tos for the net RD, represented by two thin lines on the Connection Layer, as shown in the figure above.

Your task as the designer is to translate these two From-Tos from symbolic connections on the Connection Layer into “physical” connections on the signal layers. You do this by placing tracks to create the physical connections. Whenever you stop placing tracks, the PCB Editor examines the entire net to determine what parts of the net are complete, and if the net is still broken. If any breaks are found it

inserts From-Tos to join the sub-nets, maintaining the connectivity of the net.

Because the PCB Editor monitors the completion status of the net you are routing automatically, you can route without regard to the arrangement of the From-Tos. For example, you might have commenced routing at U7-5 and then decided to route to U6-32 instead of U5-21. Once you complete this connection, the PCB Editor analyzes the entire RD net and adds a new From-To from the unrouted pin to the closest point on the other sub-net, as shown in the figure below.

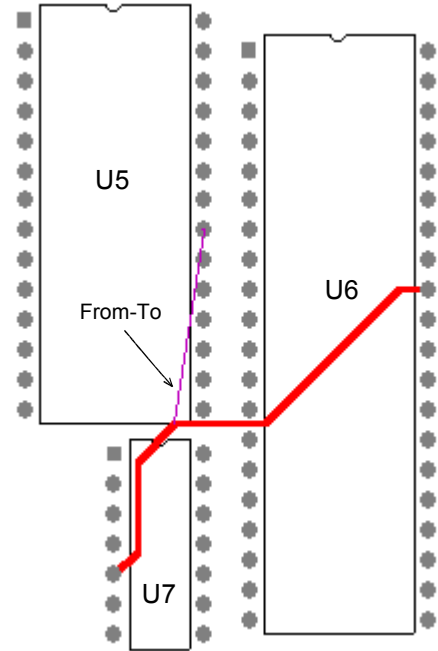


The nodes in net RD, and how that net is displayed on the unrouted PCB

To review, your task is to translate connections from symbolic *connection lines* to “physical” tracks, and the PCB Editor’s task is to monitor your progress and update the From-Tos as required.

There are two distinct advantages to this methodology. The first is that you can route a track to any primitive on the net, you do not have to route between the two pins connected by the From-To. The PCB Editor monitors your progress and adds and removes the From-Tos automatically. The second is that the net connectivity is “unbreakable”, you cannot accidentally break it into two unconnected parts. If you delete a track segment the PCB Editor detects the break and immediately adds a From-To to restore the net connectivity.

When the PCB Editor analyzes the net and determines that it must add a From-To, it adds it based on the topology of the net. By default, all nets have their topology set to *shortest*. For these nets the From-To is added where the two sub-nets are closest.



After analyzing the net, the PCB Editor adds the Broken Net Marker

If the net has a topology applied, the From-To is added to maintain the topology, and is shown as a dotted line, indicating that the net should be routed between these two points to maintain the topology.

A topology is applied to a net either through a Topology Rule, or by defining fixed From-Tos. For more information on net topology and fixed From-Tos refer to the *Working With a Netlist* topic and the Routing Topology design rule in the chapter, *Specifying the PCB Design Requirements*.

- ◆ If your design includes a large net with many nodes, or there are a large number of primitives (tracks, vias, etc) on the net, you may find that the automatic connectivity monitoring will take a long time to analyze this net. To disable the connectivity monitoring for a particular net you can “hide” the net. When you select **View » Connections » Hide Net** to hide the net connection lines, it is also hidden from the connectivity checker.

Preparing to Route

Preparing the design for routing is an important part of the design process. Use these tips to improve the routing process.

Setting the Grids

Traditionally PCB's were designed on a standard grid. The grid was calculated to allow objects to be placed quickly and accurately, without the possibility of violating the design requirements. For example; a design that used through-hole components with pins spaced in multiples of 100 mils could be routed on a 25 mil grid. This allowed for 12 mil tracks, 13 mil clearances and one track to pass between the pins of an IC.

Changes in packaging technologies, where both imperial and metric pin spacing are used, make it difficult for today's designer to specify a standard grid that fits all the component and design requirements. This has become a major shortcoming of the traditional grid-based PCB design environment.

The PCB Editor includes a number of features to aid the designer in overcoming this limitation. These include: an electrical grid which allows one electrical object to snap to the hot spot of another electrical object, even if it is off grid; look-ahead track placement, allowing you to predict where you want the next track segment to go and accurately terminate the current segment; and violation-free object placement with automatic clipping. With these features the PCB Editor behaves as a *shape-based manual router*, allowing you to route quickly and accurately to any object, at any point in the workspace.

While these features may initially sound complex, they are quite easy to work with. If your design is not suitable for one of the traditional routing setups, or the density requires you to pack the tracks and vias more tightly than a grid based routing model allows, then set the snap grid to a small value, such as 5 mils or 1 mil. Continue reading this chapter for an explanation of how to route in this mode.

For more information about setting the snap grid, component grid and the electrical grid refer to the *Grids* topic in the chapter, *Setting Up the PCB Workspace*.

Move Components onto the Grid

To maximize the number of routing channels available, as many component pads as possible should be on the snap grid. Check if the components are on grid by selecting the **Edit » Select » Off Grid Pads** menu item (shortcut: S, G). To move all the components onto the snap grid, select the **Tools » Interactive Placement » Move To Grid** menu item (shortcut: I, G). The Component Move dialog will pop up allowing you to specify the grid.

Check the Routing Density

To help in the process of determining how the board should be routed, that is, the track and grid sizes, the via size, the number of layers, and so on, the PCB Editor includes a Density Map feature. Select the **Tools » Density Map** menu item. After a few moments the board will be “painted” with a colored map. The green color represents “cool”, or less dense regions, the red color represents “hot”, or most dense regions. If there are large areas of red you may wish to analyze the current component placement and try to remove these “hot” zones. If this is not possible your design may require more routing layers.

Enable the Routing Layers

The PCB Editor has 32 signal layers (top, bottom, and 30 mid layers), as well as 16 internal power plane layers. If your design requires the use of internal signal layers and you intend to use blind and buried vias, then the signal layers are normally used as *layer pairs*. Read the chapter *Defining the Board*, for information about defining the Layer Stack, and the *Vias* topic in the *PCB Design Objects* chapter for more information on using blind and buried vias.

Set up the Design Rules

The PCB Editor is a rules-driven environment – as you place routing objects in the workspace the PCB Editor checks the rule system to see which rules apply to that object.

When you create a new PCB, there will be a number of default rules created, including a track width constraint (set to 10mils), a clearance constraint (set to 10mils), routing layers (top-horizontal, bottom-vertical) and a routing via style (50mils).

Before you begin routing you should set these to suit your design. If you have not already done so, review the chapter, *Specifying the PCB Design Requirements*, and set up the appropriate rules.

Manually Routing the PCB

- ◆ Before you commence routing, review the *Tracks Placement Modes* topic in the *PCB Design Objects* chapter. A good understanding of all the track placement features is essential to utilize the full routing potential of the PCB Editor.
- ◆ If your design includes large nets with many nodes, you may find that the automatic connectivity monitoring takes a long time. To disable the connectivity monitoring for a particular net you need to “hide” the net. When you select **View » Connections » Hide Net** to hide the net connection lines, it is also hidden from the on-line connectivity checker.

Because the PCB Editor monitors the net connectivity for you, routing is very straightforward. You place tracks, vias, fills and arcs to create the physical connectivity, the PCB Editor monitors the connectivity and updates the connection lines accordingly.

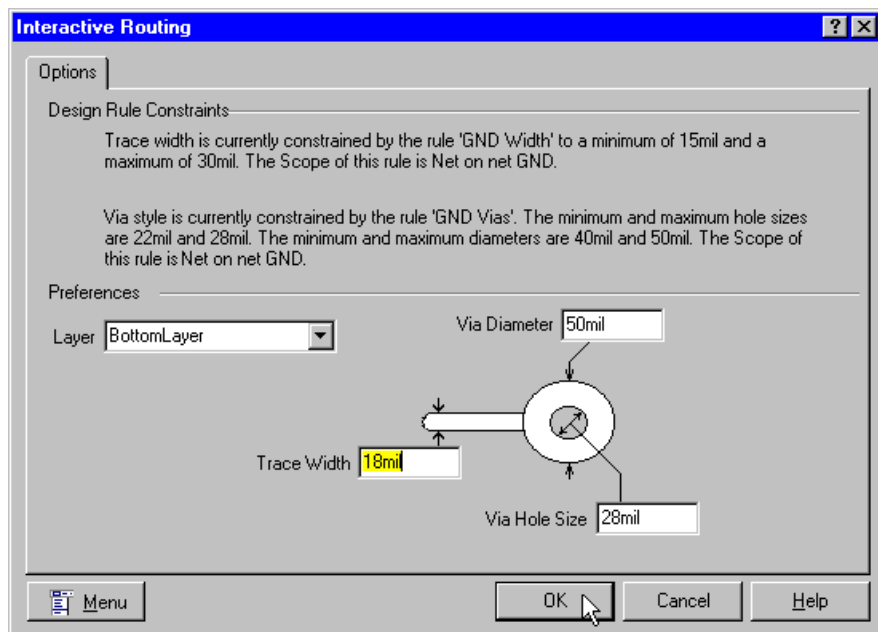
If you select **Place » Interactive Routing** and then click on a design object that has a net name, the track you are placing will adopt the net name, becoming part of that net. If you click on a connection line, the PCB Editor will jump to the nearest pad, and then keep the connection line attached to the end of the track you are placing.

When you quit from placing tracks, the PCB Editor examines the net and updates the connection lines. The connection line may or may not stay on the end of the track. The PCB Editor adds the connection lines between all the parts of the net that are not physically connected, based on the net topology. The default topology is shortest, so the connection lines will be added between the closest points on the sub-nets. For more information on net topology refer to the chapter *Understanding Connectivity and Topology*, and the Routing Topology design rule in the chapter, *Specifying the PCB Design Requirements*.

The track width that is used when you click on a pad to start routing is defined by the Width Constraint design rule. When you create a new PCB document it will have a number of default rules, including a 10 mil Width Constraint applying to the whole board. If you would like a particular net to have a different track size, apply a Width Constraint to that net. Refer to the chapter, *Specifying the PCB Design Requirements*, for information on applying design rules.

Changing the Routing Parameters During Routing

During manual routing the track width and routing via parameters can be changed on-the-fly by pressing the TAB key. This pops up the Interactive Routing dialog where you can change the track width, the via diameter and the via hole size. Changes made to these settings updates the Preferred attributes of the applicable Width Constraint and Routing Via Style design rules. If the values are changed to be outside the current maximum and minimum settings of the applicable rule, they are automatically clipped.



Press the Tab while routing to change the track width and via parameters

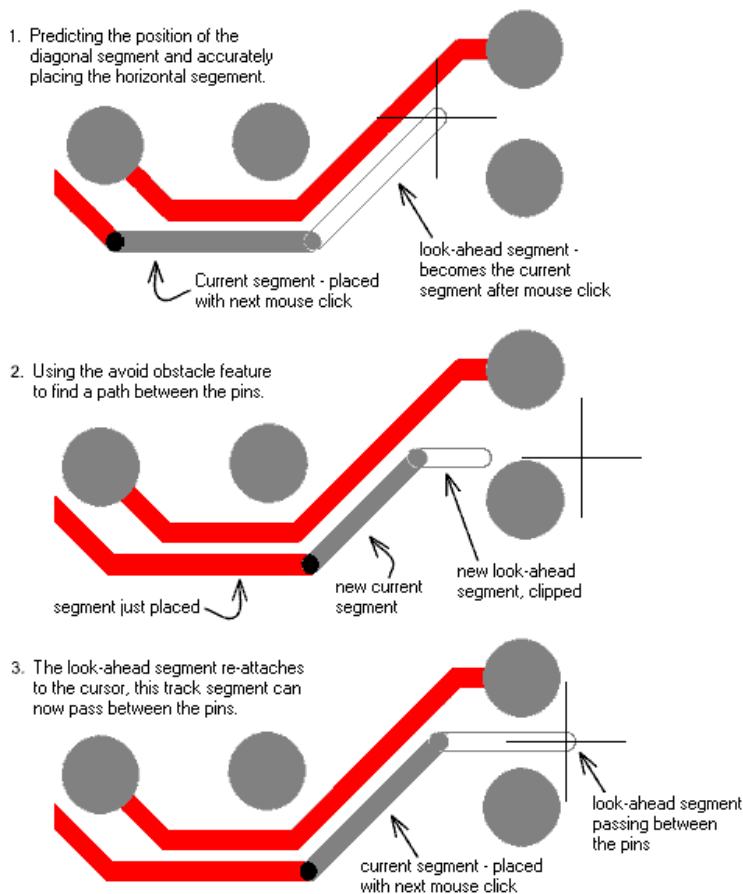
Routing Shortcuts

Use the following shortcuts to speed the routing process:

- Press the BACKSPACE key to remove the last “corner” during routing.
- Press the * key to toggle through the routing layers while routing and insert a via.
- Press the / shortcut key to place a via connecting to a power plane.
- Use the SPACEBAR to change between the Start and End placement modes. Press SHIFT+SPACEBAR to change track placement modes.
- Press the CTRL+SPACEBAR to cycle through each connection line that connects to the pad you just started routing from.
- Press the END shortcut key to refresh the display while routing.
- Hold the CTRL key to temporarily suspend the electrical grid
- Press the SHIFT+E shortcut keys to toggle the electrical grid on and off.
- Hold the ALT key to temporarily switch from Avoid Obstacle mode to Ignore Obstacle mode.
- Press the SHIFT+R shortcut keys to cycle through the 3 Interactive Routing modes.

Placing Tracks and Looking-Ahead

The PCB Editor incorporates a sophisticated “look-ahead” feature that operates as you place tracks. The track segment that is connected to the cursor is a look-ahead segment (shown in outline/draft mode). The segment between this look-ahead segment and the last-placed segment is the current track that you are placing (shown in final mode). This is shown in the following diagram.



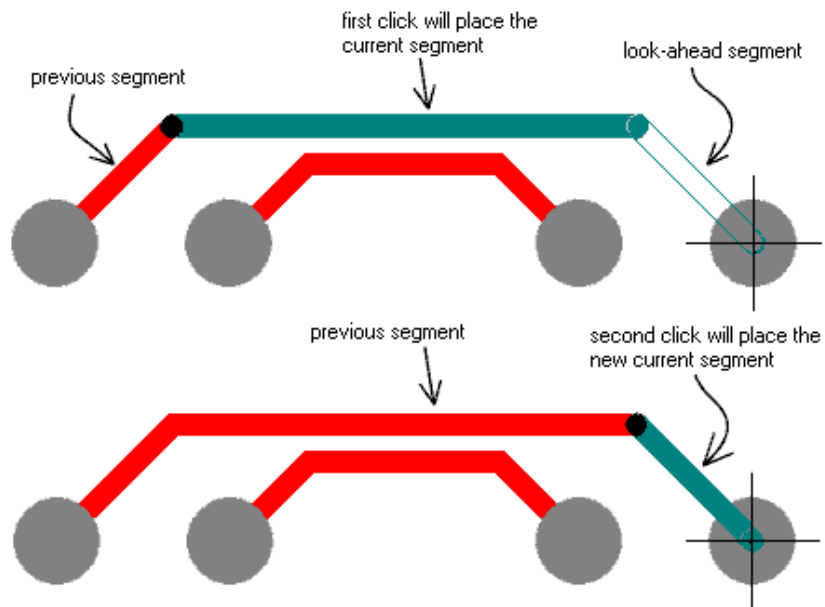
Use the look-ahead feature to predict the next segment, and accurately position the current one

Use the look-ahead segment to work out where you intend to place the next segment and to determine where you wish to terminate the current segment. When you click to place the current segment, its end point will be positioned exactly where you need to commence the next segment. This feature allows you to quickly and accurately place tracks around existing objects and plan where the next track segment can be placed.

As you use the look-ahead segment to guide your routing, you will notice that the track end does not always remain attached to the cursor. It “avoids” electrical objects that belong to another net. This feature allows you to only place primitives where they do not violate any clearance design rules, as shown in step 2 of the previous diagram. In step 2 the cursor has been moved to the right of the pads, but the look-ahead segment is clipped back to the point where no violations would exist. As soon as the cursor is moved up to a point where the look-ahead segment can pass between the pads without causing a violation, it extends across to the cursor. Routing Modes are discussed in more detail in the next topic.

Consider another example of manual routing shown in the following diagram. You wish to route horizontally across, and then diagonally down to a pad. Previously this was a process of trial and error, judging exactly where to terminate the horizontal segment and commence the diagonal one. The look-ahead segment allows you to bring the cursor down onto the target pad, clicking once to terminate the horizontal segment, then clicking a second time to terminate the new diagonal segment.

◆ Remember, the segment displayed as an outline is the look-ahead segment, not the segment you are currently placing. If you are trying to place a segment and nothing happens when you click, you are probably trying to place the look-ahead segment. This can happen when you have the placement mode set to End when it should be Start, or it is set to Start when it should be set to End. Press the SPACEBAR to toggle between the Start and End placement modes.



Using the look-ahead feature to predictively place tracks

Interactive Routing Modes, including Push and Shove

The PCB Editor has three different interactive routing modes; Ignore Obstacle, Avoid Obstacle, and Push Obstacle. These modes define the behavior of the track you are placing, in relationship to the existing objects on the board. All three modes work in conjunction with the Loop Removal feature.

Ignore Obstacle

In this mode you can route tracks anywhere in the workspace. If the track you are placing violates one of the design rules, then the new track and the existing object will both be highlighted as an error, assuming on-line DRC is on (**Tools » Design Rule Check** and **Tools » Preferences**). This mode is useful when you are rerouting and you need to temporarily create a violation, which will be removed by subsequent routing.

Avoid Obstacle

In this mode you can not route a track to create a violation. When you move the cursor to a location where a violation would be created, the track segment automatically clips back to the point where it complies with the design rules. Working in this mode makes the task of routing very easy – you do not need to carefully place tracks, always watching to avoid creating errors – the PCB Editor only allows tracks to be placed where they do not create violations. This is the suggested mode for manual routing.

◆ Hold the ALT shortcut key to temporarily switch from the Avoid Obstacle mode to the Ignore Obstacle mode.

Plowing Through Polygons

If the Plow Through Polygon option is enabled in the **Preferences** dialog you can route over the top of a polygon when the Interactive Routing Mode option is set to Avoid Obstacle.

When you finish routing the polygon automatically repours, depending on the settings of the Polygon Repour options. The Repour option defines when a repour should occur. If Threshold is selected, then polygons with more than the Threshold number of primitives will prompt to confirm before performing the repour.

Push Obstacle

In this mode, the track you are placing pushes existing tracks out of the way as you move the cursor. It can not push immovable objects, such as vias, pads and locked tracks. If the track(s) that you are pushing can no longer move (because they have been pushed up against an immovable obstacle) then the routing mode reverts to Ignore Obstacle. This mode is ideal for rerouting. There may be a large number of changes to the contents of the screen as you work in this mode – press the END key on the keyboard to refresh the display as you route.

◆ Use the SHIFT+R shortcut keys to cycle through the modes as you are routing.

Re-routing

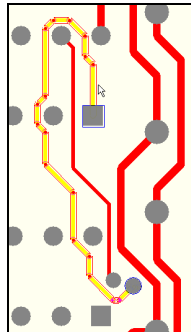
Re-routing is a normal part of the design process. Perhaps a new component has been added, a footprint changed, or you are tidying up after autorouting. The PCB Editor includes powerful features to assist in the process of re-routing tracks – the Push Obstacle routing mode, and automatic loop removal. Both of these options are configured in the Options Tab of the Preferences dialog (**Tools » Preferences**).

To re-route an existing track:

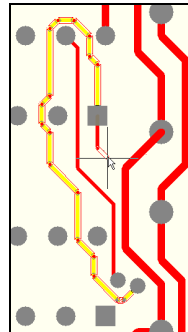
1. Select the **Place » Interactive Routing** menu item.
2. Click on a track or pad to commence re-routing.
The net will highlight, and a track segment will appear attached to the cursor.
3. Re-route the track along its new path, pushing existing tracks as required, bringing the new routing back to meet the old track. This can be anywhere along its length, or at a pad.
4. When you have finished the re-route, click RIGHT MOUSE or press ESC.

◆ Loop removal and the Push Obstacle mode are enabled in the Options Tab of the Preferences dialog.

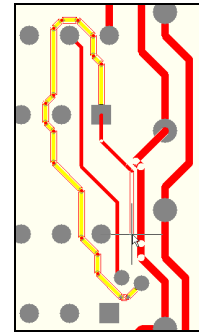
This new track will create a loop. The PCB Editor will now remove the old part of the loop. The re-route can include new vias, and the automatic removal of existing vias.



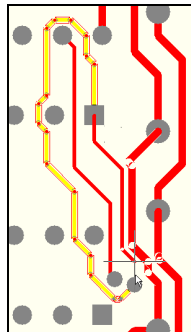
Note the path of this connection



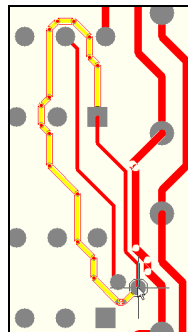
Starting to reroute



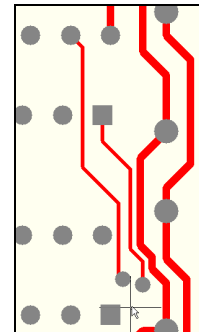
The wide track is pushed away



The route approaches the pad, pushing the wide track further



The reroute is complete, press ESC to exit



The old routing is automatically removed

Using Internal Power Planes

Power planes are special “solid” copper internal layers. The PCB Editor supports up to 16 internal power planes. If your design is netlist-based, you can assign a net to each of these layers. It is also possible to “share” a power plane between a number of nets by splitting it into two or more isolated areas.

There are two styles of connecting each pin to a power plane: either a direct connection or a *thermal relief* connection. Thermal relief connections are used to thermally isolate the connected pin from the solid copper plane when the board is soldered. The PCB Editor allows you to define the thermal relief shape of each or all pads connecting to the power plane.

Special support is also provided for connecting SMD power pins to power plane layers. SMD pads on a net that is connected to a power plane are automatically “tagged” as connected to the appropriate plane. The autorouter completes the physical connection for these pads by placing an SMD “stringer” – a short track and via which is relief or direct connected to the plane layer.

Assigning a Net to a Power Plane

To assign a net to a power plane:

1. Select the **Design » Layer Stack Manager** from the menus to pop up the Layer Stack Manager dialog.
2. Add a new plane to the layer stack (if one has not already been added) – to do this click in the picture on the layer name that you want the plane to be underneath, then click the Add Plane button. Note that PCBs are fabricated from an even number of copper layers (for example, you can not normally have a 3 layer board), so you may need to add another signal or plane layer to return to an even number of layers.
3. Double-click on the new plane layer to assign the net, you can also change the layer name if you want to.
4. Close the Layer Stack Manager dialog.

All the From-Tos for that net will disappear. A small cross will appear at each pad on the net, on the appropriate power plane layer. The cross will look like a “+” for a relief connection, or an “x” for a direct connection.

◆ The properties of the pad-to-plane and via-to-plane connections are controlled by the Power Plane Connect Style design rule. Refer to the chapter, *Specifying the PCB Design Requirements*, for information on using this rule.

Pins that Do Not Connect to a Power Plane

Pads not connecting to the plane are isolated from the plane by a region of no-copper. This region of no-copper is specified as a radial expansion around the pad hole by the Power Plane Clearance design rule. Refer to the chapter, *Specifying the PCB Design Requirements*, for more information on using this rule.

Connecting Vias to Power Planes

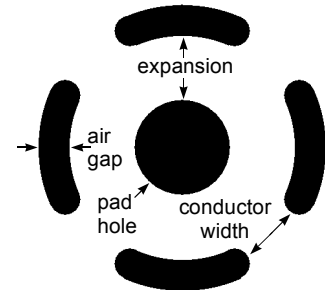
Like pads, vias automatically connect to an internal power plane layer that is assigned the same net name. The via will connect in accordance with the applicable Power Plane Connect Style design rule.

If you do not want vias to connect to power planes, add a Power Plane Connect Style design rule with a connection style of No Connect. Refer to the topic *Examples of Using Design Rules*, in the chapter *Specifying the PCB Design Requirements*, for an example of how to set this rule up.

Viewing a Power Plane

Power planes are created in the negative. Objects placed on the power plane layer become voids in the copper, while the unfilled regions on your screen will become solid copper. To view how the pins connect to a power plane:

1. Enable only the plane layer, pad holes layer and the multi layer.
2. Click on the power plane layer Tab at the bottom of the workspace.
3. If necessary, redraw the screen by pressing the END key.



Relief connected pads are displayed as shown in the adjacent figure. As direct connected pads have solid copper to the pin, they simply show black up to the pad hole. Note that in this book, the white page represents copper and the black represents no-copper.

Creating a Split Power Plane

When you wish to share an internal power plane between two or more nets, you divide, or “split” the plane into regions. Typically the net with the greatest number of pins is first assigned to the internal plane, then regions are defined (split off) for the other nets that you wish to connect via this plane. Each region is defined by placing special boundary tracks to encompass all the pins on that net. Any pins which cannot be encompassed in the split plane region continue to display a connection line, indicating that these pins must be connected on a signal layer.

Power planes are constructed in the negative, so the special boundary tracks that you place become a strip of no-copper, creating the separation between this net and the adjacent net(s) on the plane.

Before attempting to define a split plane read the topic *Tips for Defining a Split Plane* later in this chapter. To create a split plane:

1. First assign the net with the greatest number of pins to the plane, refer to the topic *Connecting to an Internal Plane* on the previous page for details on this.

2. Make the internal plane the current layer by clicking on its layer Tab at the bottom of the workspace.
3. Select the **Design » Split Planes** menu item to pop up the Split Planes dialog.
4. To add a Split Plane press the Add button. The Split Plane dialog will pop up. Set the Track Width, Layer and Connect to Net as required.
5. Click OK when the Split Plane dialog is set up. The dialog will disappear and a cross hair will appear on the cursor.
6. Click to define each point on the boundary, coming back to the start to create a closed boundary.

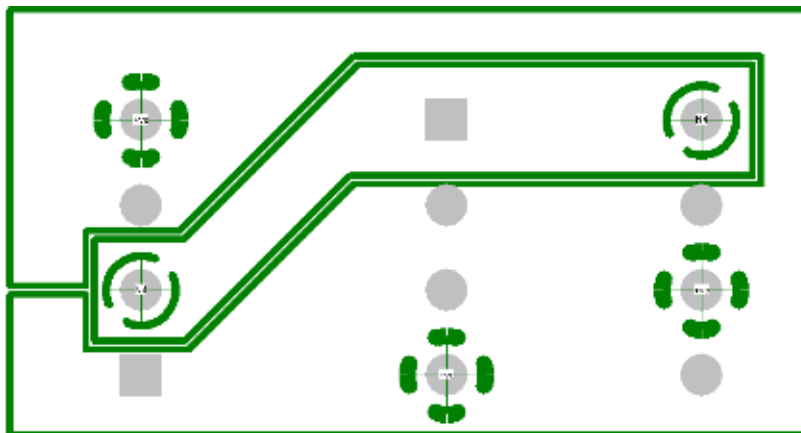
Once the boundary is closed the Internal Planes dialog will reappear. The new split plane will appear in the Split Planes list, click on it to display the region.

◆ Use the SPACEBAR to change between the different split plane boundary placement modes. The boundary can also include arcs.

Splitting a Plane to Support More than Two Nets

If the plane is to connect to *more than two nets*, continue to add split planes to create the other regions on the plane. Adjacent boundary tracks can be placed on top of each other if desired. Typically they would be placed with the tracks overlapping by a small amount to create a continuous no-copper region.

To define a split region within another split region, the outer split region must “wrap around” the inner split region, as shown in the following figure.



A split region within a larger split region – note how the larger region wraps around the inner region. In this example there are now three nets sharing this power plane, the original net associated with the plane, plus the two nets using these split regions. The boundary tracks have only been made thinner to clearly show the two split regions, normally they would overlap slightly.

Tips for Defining a Split Plane

To make the pads that you wish to encompass within each split plane easily visible, the following is suggested:

1. Display only a minimum of layers – the Keep Out layer, the multi layer, any mechanical layers needed and the power plane which is being used.
2. Set the display mode of pads to draft mode in the Show/Hide Tab of the Preferences dialog.
3. Enable the Highlight In Full option and the Use Net Color For Highlight option in the Options Tab of the Preferences dialog.
4. Set the color attribute of each net on the split plane to a different color. To do this, set the Browse mode in the Panel to Nets, select the net in the list and press the Edit button to pop up the Change Net dialog.
5. Select the **Edit » Select » Net** menu item and click on one of the pads on the net. Repeat this for the other net connecting to the plane. Extend Selection must be enabled to select more than one net at a time (Options Tab of the Preferences dialog). These two sets of pads will appear in different colors and should now be easy to identify.
6. You are now ready to define the split region(s). Assign the most common net to the internal plane, then define a split region for each of the other nets sharing the plane.

◆ A split region must not overlap into another region, always wrap one around the other.

Modifying a Split Plane

The boundary of a split plane can be modified after it has been defined. The following modifications are supported:

1. The boundary track width, the layer that the split plane is on, and the net that is connected can all be changed. To change one of these attributes double-click inside the boundary of the split plane to pop up the split plane dialog.
2. The location of the boundary tracks can be changed. Select the **Edit » Move » Split Plane Vertices** menu item. You will be prompted to Choose a Polygon (a split plane is just an empty polygon), click inside the split plane to be modified. The boundary track editing handles will be displayed. Click on an editing handle to move the handle.

◆ For more information on modifying a polygon (a split plane is an empty polygon), and an activity which takes you through the steps of modifying a polygon boundary, refer to the *Polygons* topic in the *PCB Design Objects* chapter.

Creating a Copper Plane on a Signal Layer

A common part of a PCB design is a copper plane on a signal layer. This may be a hatched ground plane on an analog design, it may be a solid power supply plane for carrying heavy currents, or it may be a solid ground plane for EMC shielding.

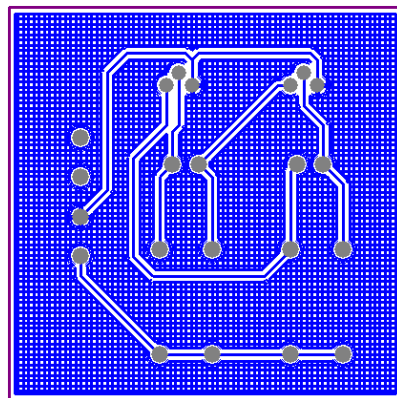
You can create a copper plane on any signal layer, by placing a polygon plane. The copper plane is created by interactively defining the boundary, which the PCB Editor then fills in, avoiding existing objects in accordance with the design rules.

There are a number of options that define how a polygon is filled in. These include 90 degree hatching, 45 degree hatching, horizontal tracks, vertical tracks, and completely filled in.

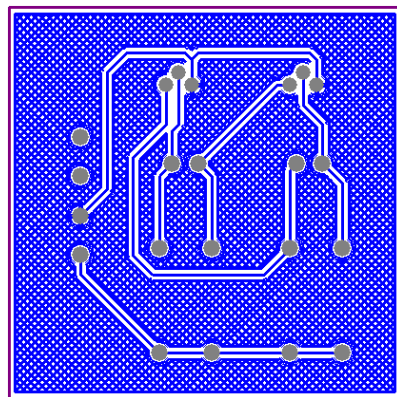
Polygons are created by selecting **Place » Polygon Plane** from the menus. The polygon can be attached to a net – if it is, it will connect to each pad on this net in accordance with the Polygon Connect Style design rule. Select **Design » Rules** from the menus and click on the Manufacturing Tab to change the connect style.

The default polygon settings will give a hatched polygon, because the grid size is larger than the track size. To completely fill the polygon make the track size smaller than the hatch size.

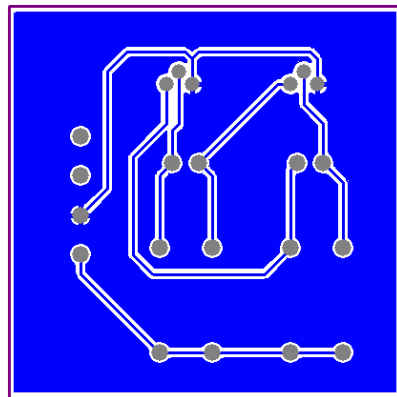
♦ Refer to the *Polygons* topic in the *PCB Design Objects* chapter for complete details of the Polygon Plane dialog, and activities on placing and modifying a polygon plane.



Polygon with 90 degree hatch



Polygon with 45 degree hatch



Solid polygon

Autorouting the PCB

Protel 99 SE includes an easy-to-use, powerful, high quality, shape-based autorouting server, which is tightly integrated with the PCB Editor. Although it plugs in as a separate server in the Design Explorer, the Autorouter is set up and run in the PCB Editor, and routes directly in the PCB window.

Using the Autorouter is easy. Simply define the appropriate PCB design rules, and select **Autoroute » All**. While the Autorouter is routing your board, you might like to browse this chapter to learn more about the power and capabilities of Protel 99 SE's shape-based Autorouter.

Setting Up to Autoroute

Protel 99 SE's Autorouter is tightly integrated with the PCB Editor. Although it is a separate Design Explorer server, it routes your board in the PCB window. The Autorouter will route the board in accordance with the current design rules.

Setting Up the Design Rules

Select **Design » Rules** from the menus to set up the routing rules prior to running the autorouter. The autorouter obeys most of the routing design rules, there are some rules that are not obeyed with certain scope settings. Rule compliance is reported at the bottom of the Design Rules dialog – to check if a rule is followed by the router click on the rule to select it, a message will appear stating if that rule will be followed by the router.

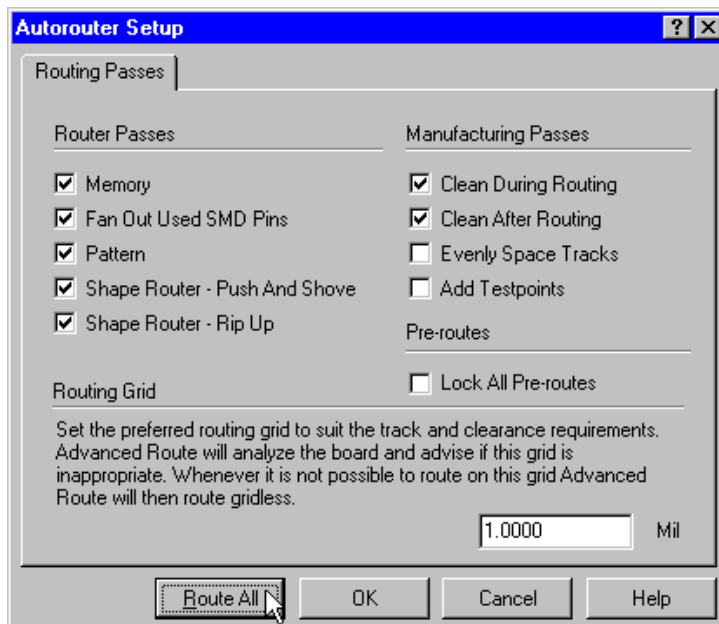
Protecting Pre-Routes

Often you will want to manually pre-route certain nets, then autoroute the remainder of the board. You can protect the pre-routes from being ripped up and rerouted by the Autorouter by enabling the Lock All Pre-routes option in the Autorouter Setup dialog.

◆ The board should be free of design rule violations before autorouting. This means that all pre-routed tracks should be covered by appropriate design rules. For example, if you have pre-routed the ground net using a mixture of 20mil and 50mil tracks, your Width Constraint for the ground net must have a Minimum Width of 20mils, and a Maximum Width of 50mils.

Routing Passes

To set up the routing passes select **Tools » Autoroute » Setup Autorouter** to pop up the Autorouter Setup dialog. Enable the required routing passes.



Enable the routing passes in the Autorouter Setup dialog

Unless you have good reason for so doing, we recommend that you always use the default router pass setup, which runs all the algorithms. The description of each pass includes notes about when it is better to not run all passes.

Memory

This pass selects all Memory or Memory-like nets on the board. This algorithm is both Heuristic and Search. Always run it, even if you do not have memory components on the board.

- ◆ If you have a true memory bank on your board and are concerned with where it should be placed, how the components should be oriented, and so on, select Memory only to evaluate the memory pattern.

Fan Out

This pass is used to fan out or “disperse” vias from surface mount component pads. A short segment of track is routed out from each surface mount component pad and a via is placed at the end. The Fan Out pass is both Heuristic and Search and should always be used if there are single-sided components on either top, bottom or both layers. Fan Out fails are shown as a small yellow circle with an X in the center of the circle.

♦ Very dense boards with SMD parts on both the top and bottom layers may have difficulty during Fan Out. We suggest you make a trial run with only the Fan Out pass active before you commit to autorouting the entire board. If you find that about 10% of the total pins to be fanned out fail, then it is highly likely that 100% completion will not be achieved. If this happens you should consider repositioning the components in the area of the board where the Fan Out fails occurred.

Pattern

On virtually every board you can find connection patterns. The success in routing these patterns depends upon the sequence that the connections are selected while routing the pattern. The Pattern router has a collection of different algorithms, each addressing a particular type of pattern. The Pattern router is a Search router. It should always be utilized during autorouting.

Push and Shove

The Push and Shove router is the main routing pass within the Autorouter. It is considerably advanced over the conventionally available Push and Shove routers, in that it pushes and shoves on a diagonal, has no limits on how far it can push other tracks, can hop over vias and pads, and in general, represents a new level of sophistication and power in Push and Shove algorithms.

Rip Up

The Autorouter is a “contention cleanup” router. Upon completion of the Push and Shove Pass there may be spacing violation contentions remaining. Contentions are shown on the monitor as a small yellow circle. Generally, successive passes of the various routing algorithms will remove these contentions. In very difficult boards there may be contentions remaining after all the algorithms have been completed. The Rip Up router is then used to rip up the routed tracks associated with the contentions and reroute them so as to eliminate the contention.

Manufacturing Passes

These passes evenly space tracks, miter track corners and add testpoints to each nets. They can be included in the initial router setup, in which case they will be run upon completion of the primary routing passes. Alternatively, you can run only the primary routing passes, examine the board and then activate the Manufacturing Improvement Passes and run them.

Cleanup Passes

There are 2 cleanup passes – the first runs during the routing sequence, at the end of each routing pass, the second runs after all the routing passes are complete. The cleanup passes are designed to be used in conjunction with the main routing passes, they focus on straightening the routing connections and cleaning the pad entries.

Evenly Space Tracks

This pass is used to evenly space tracks between pads in the following situation. When the routing parameters allow for two tracks between IC pads but only one track is routed between two particular pads, it may be placed on a 20 mil channel near one or the other of the IC pads. Running Evenly Space Tracks will shift this track to the center of the space between the IC pads.

Add Testpoints

Enable this pass to instruct the autorouter to attempt to add a testpoint to each net. Before adding any new testpoint pads, the board is scanned for existing pads/vias that could be defined as testpoints. The autorouter will then attempt to place a test point pad for any net that does not have a testpoint, in accordance with the Testpoint design rules. Refer to the following chapter, *Including Testpoints on the PCB* for more information on testpoints.

Autorouting Options

As well as being able to route the entire board, there are a number of other selective routing options.

Autoroute All

To route the entire board with the current routing setup select **Autoroute » All**.

Autoroute Connection

Autoroute Connection allows you to select the order in which you want the Autorouter to route, on a connection-by-connection basis. Not all of the full Autorouter algorithms are used by **Autoroute » Connection**, so you may not want to attempt to route the entire board with this option.

Autoroute Net

Select **Autoroute » Net**, place the cursor over any connection in the net and click. All connections in the net will be routed, using all the routing passes.

Autoroute Component

Select **Autoroute » Component**. Click on a component pin and all connections Starting/Ending on that component will be routed. Note that within a net, only the connections starting and ending on the selected component will be routed.

Autoroute Area

Select **Autoroute » Area**. Using the cursor, draw the area to be routed. All connections starting and ending in the designated area will be routed.

Inside Protel 99 SE's Autorouter

Why is Protel 99 SE's Autorouter a Better Autorouter?

Protel 99 SE's Autorouter is the most recent generation of board autorouters in a long history of autorouter development that spans nearly three decades - but with a difference. Prior to the Protel 99 SE Autorouter, autorouters have had two prime objectives: (1) to route boards to 100% completion, and (2) to route boards in the minimum amount of time. The Protel 99 SE Autorouter also has "completion and fast route time" as two of its objectives and is one of, if not the most powerful and fastest autorouter available in desktop EDA today. However, Protel 99 SE's Autorouter has a third objective. This characteristic is "quality of routing".

Historically, most professional board designers have held the opinion that an autorouted board does not provide the classical "quality" associated with a designer-interactively routed board. Complaints are many and include: too many vias; tracks wander around the board; vertical layers contain too many horizontal segments and vice versa; tracks not evenly spread and autorouted boards being too difficult to edit. The most often expressed comments are about X-Y Orthogonal routing versus Diagonal routing. Experienced designers instinctively route using diagonals, whereas autorouters usually route in an X-Y orthogonal fashion. Routing on a diagonal not only presents a more pleasing, aesthetic appearance, but in a given area, diagonal routing can include more track segments than routing in an X-Y fashion. Hence the probability of finishing a more difficult board may be increased by routing on a diagonal.

To summarize, the Protel 99 SE Autorouter has three objectives: 100% completion, routing speed and quality of routing (like a professional board designer). To meet these objectives, the Autorouter uses a combination of new technologies as well as time-proven developments.

Angled Direction By Layer - What Is It?

Routing in a specific non-orthogonal direction per layer of a multi-layer board is a concept practiced by experienced professional board designers. Before this autorouter it has not been available in a board autorouter.

Simply stated, routing with "direction per layer" means ordering a connection to a layer based upon the user-assigned layer direction and the connection's "slant" or tangent, and then routing the track along the slant or approximate hypotenuse of a triangle as opposed to routing orthogonally along the X and Y legs of the hypotenuse.

The advantages of routing a "direction per layer" include: (1) the approximate hypotenuse track length is shorter than the X-Y leg length, and (2) there are fewer vias.

This shorter track length provides considerably more flexibility in the placement of parts, and together with fewer numbers of vias per connection, are important considerations for High Speed Logic autorouting.

Additionally, an advantage of “angled direction per layer” for non-High Speed Logic boards (as well as for High Speed Logic) is the increased probability of routing to 100% completion.

The choice of which direction to use per layer is entirely that of the designer and in fact, may be limited to conventional horizontal/vertical, if desired. During the autorouting sequencing, the sequencer selects those connections that most nearly match each layer direction and routes the selected track in the selected direction on that layer.

The basic movement utilizes long 45° diagonal segments with incremental orthogonal steps that bring the primary direction in alignment with the selected direction.

Contention Routing

The term “contention” is a new word in autorouting vocabulary and its significance is as follows. All the routing algorithms included in Protel 99 SE’s Autorouter are contention routers. That is to say, it is permissible for the algorithm to place a track or via in such a position that it will create a spacing violation with other tracks, vias and pads. This spacing violation is a “contention” between an existing track, via or pad and the location where the autorouter would like to place the track being routed.

A contention can be of the following form. A track segment crossing a track segment of a different net, a track segment being placed on top of a track segment of a different net, a track segment that violates space with an existing via or pad, or a via that violates space with another via, pad or track segment.

When a contention is created, a small hollow circle is created at the point of contention and maintained until the contention is cleared. Subsequent router passes will attempt to remove the contention by either pushing and shoving or rerouting the track that caused the contention while being routed, or else pushing, shoving or rerouting the existing track where the contention was created. In those rare instances when there is simply not enough room to push and shove or reroute a track to eliminate the contention, the contention will remain at the completion of routing.

The number of contentions is shown on the Status line. At certain times the number of green circles (contentions) on the screen may differ from the number shown on the Status line. This is because the screen is updated instantly as a connection is routed, while the Status line is updated after a major routing pass.

Including Testpoints and Teardrops on the PCB

Printed circuit board testpoints are locations on the board that can be used to test the correct function of the board. Testpoints are used to validate the board before the components are added – ensuring that the connectivity and isolation requirements of the routing is correct, and after the board is assembled – ensuring that the circuit functions within the design specifications.

Each testpoint is a location where a test probe can be connected to the PCB. Test probes can be connected by automated test equipment (typically by a bed-of-nails type test jig), or they can be connected by a person during manual testing.

The automated test equipment applies voltages and currents to each testpoint in turn, and simultaneously takes measurements at the other testpoints. On an unloaded board the PCB can be tested for correct continuity between every node in each net, for shorts between different nets, and the isolation resistance between nets can be measured. On an assembled board the testpoints are used to measure the performance of the circuit.

The steps you go through to add testpoints to your design include:

- Defining what is a valid testpoint
- Defining which nets require testpoints
- Finding existing pads and vias that can be used as testpoints
- Adding testpoints during the autorouting process
- Reporting the location of each testpoint
- Reporting any nets that failed to receive a testpoint

What is a Testpoint?

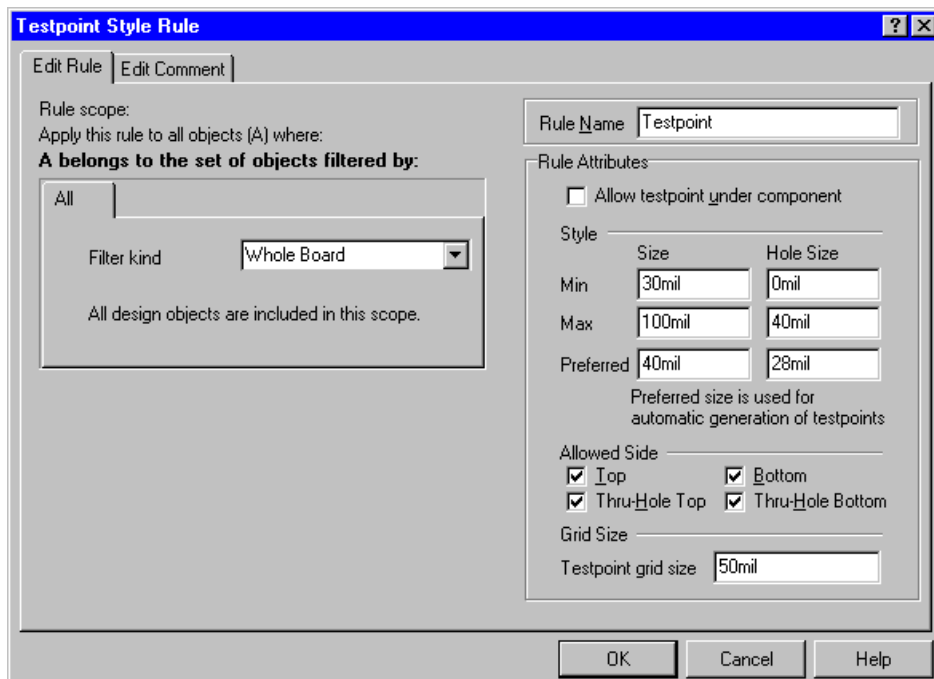
Both pads and vias can be used as testpoints, by enabling one or both of their testpoint attributes. These attributes allow any pad or via to be nominated as a top layer testpoint, a bottom layer testpoint, or as a top layer and bottom layer testpoint.

Keep the testing method in mind when you choose the side of the board that testpoints will be allowed on – will the board be probed from the bottom side only, or the top side only, do some points need to be on the bottom for automated testing and others on the top for infield testing, and so on.

Defining the properties of pads and vias that can be used as Testpoints is done by configuring the Testpoint Style design rule.

Setting up the Testpoint Design Rules

Testpoint Style Design Rule



The testpoint style design rule defines the properties of the pads/vias that can be used as testpoints

The Testpoint Style design rule specifies the allowable physical parameters of pads and vias that are used as testpoints. The attributes of the rule include:

Allow testpoint under component – If this option is enabled testpoints are allowed under a component, on the same side of the board that the component is mounted on. Testpoints that are under a component, but on the opposite side of the board, are always permitted. Normally the only time you would allow a testpoint under a component would be when that testpoint is used for testing the board prior to assembly.

Style – The Min and Max settings define the minimum and maximum pad/via diameter and hole diameter of a valid testpoint. These settings are used by the Testpoint Find feature, and by the on-line and batch DRC. The Preferred settings define the size of testpoint pads placed by the autorouter. Set the Style settings to suit your testpoint requirements.

Allowed Side – These options define the side of the board the testpoint must be on, and whether the testpoint can be pads/vias with a hole, or single-sided pads. These settings are used as preferences by the Testpoint Find feature when it searches for possible

testpoints, and by the Autorouter when it places testpoints. The Testpoint Find feature and the Autorouter use the options in the following pre-defined order:

- Bottom – bottom layer surface mount pad
- Top – top layer surface mount pad
- Bottom thru-hole – bottom side of a via
- Top thru-hole – top side of a via
- Bottom thru-hole – bottom side of a pad with a hole
- Top thru-hole – top side of a pad with a hole

Each setting is only considered if it is enabled. The bottom thru-hole option enables both bottom side thru-hole pads and vias (and likewise for the top thru-hole option) – they are listed separately to show that vias have a higher priority during testpoint searching by the Testpoint Find feature.

Grid Size – The grid is used by the Testpoint Find feature when it attempts to locate possible testpoint sites.

Testpoint Usage Design Rule

This rule is used to specify which nets must have a testpoint, or which nets must not have a testpoint. Set the rule scope to target those nets that must have a testpoint.

Locating Existing Testpoint Sites

The Testpoint Find feature searches for existing pads and vias that can be used as testpoints. Select **Tools » Find and Set Testpoints** from the menus to run the Testpoint Find feature. Obeying the Testpoint Usage and Testpoint Style rules, this feature will analyze the PCB and attempt to find a pad or via that complies with the Testpoint Style rule, for each net covered by the Testpoint Usage design rule, and set the appropriate testpoint attribute for that pad or via. When it has finished, a dialog will appear reporting how many pads and vias were found and set to testpoints (found testpoints are left selected once it has finished). All Testpoint attributes can be cleared at any time by selecting **Tools » Clear all Testpoints** from the menus.

If the pre-defined Allowed Side order is not appropriate for your design when you are using the Testpoint finder, run multiple passes with different allowed side settings.

Adding Testpoints with the Autorouter

Enable the Add Testpoints option in the Autorouter Setup dialog to instruct the autorouter to attempt to add testpoints once it completes all other routing passes, in accordance with the Testpoint Style and Testpoint Usage design rules.

If the Add Testpoints option is enabled the autorouter first scans the board for possible existing testpoint pads/vias, then attempts to place a testpoint pad in each net which still needs one.

Reporting the Location of Each Testpoint

The CAM Manager testpoint report includes complete details about each testpoint on the board. Select **File » CAM Manager** from the menus to create a new CAM document, where you can set up a Testpoint report. Refer to the chapter, *Generating the Manufacturing Files*, for complete details on using the CAM Manager to generate a Testpoint report.

Reporting Nets that Failed to Receive a Testpoint

The DRC report can list each net that failed to receive a testpoint in accordance with the Testpoint design rules. Enable the Testpoint Usage rule on the Report Tab of the Design Rule Check dialog (select **Tools » Design Rule Check**), enable the Create Report File option at the bottom of the dialog, then click the Run DRC button at the bottom of the dialog to generate the report. The report will include a list of nets that do not include a testpoint.

Enable the Testpoint Style rule in the Design Rule Check dialog to ensure that the pads and vias that are flagged as testpoints fit within the requirements of the Testpoint Style design rule.

Clearing all Testpoints from the Board

The Testpoint attribute for all pads and vias can be cleared at any time by selecting **Tools » Clear all Testpoints** from the menus.

Adding Teardrops to Pads and Vias

Teardrops can be added to pads and vias by selecting **Tools » Teardrops** from the menus. This will pop up the Teardrop Options dialog, where you configure the teardrop requirements. The dialog includes the following options.

All Pads – Add teardrops to all pads, in accordance with the Selected Objects Only and Force Teardrops options.

All Vias – Add teardrops to all vias, in accordance with the Selected Objects Only and Force Teardrops options.

Selected Objects Only – Add Teardrops to selected pads and vias only. Use this option when you need to teardrop some pads/vias, but not others. Use the global editing feature to change the selection status of the pads/vias that need to have teardrops.

Force Teardrops – Place teardrops on all pads/vias, including those where the teardrop tracks/arcs will create a DRC violation.

Create Report – Create a report file which lists the number of pads and vias that teardropping was attempted on, as well as each pad and via that was not teardropped, or not completely teardropped.

Verifying the PCB Design

Design verification is the process of ensuring that your design is correct and complete. It is a fundamental and integral part of the board design process. The verification process must ensure that the board logically matches the schematic. It must also ensure that the board will be physically functional – electrical objects such as tracks, pads, vias, etc, must not violate their clearances.

The PCB Editor includes a powerful Design Rule Check feature which verifies that the design meets the requirements specified by the design rules. It tests for routing violations such as clearance errors, unrouted nets, width errors, length errors, and also conditions that effect manufacturing.

To enhance productivity while designing the board, the PCB Editor includes an online DRC feature, which tests for enabled design rules as you route. To clear the violation simply move the object causing the problem and continue routing.

♦ To be able to verify that your design is correct and complete, you must have some reference to test against. The reference that the PCB Editor uses is the set of design rules defined in the Design Rules dialog, and the netlist that was loaded at the start of the design process. Refer to the chapter, *Specifying the PCB Design Requirements*, for further information on design rules.

What is the Design Rule Checker?

Design Rule Checking (DRC) is a powerful automated feature that checks both the logical and physical integrity of your design. This feature should be used on every routed board to confirm that minimum clearance rules have been maintained and that there are no other design violations. Because the PCB Editor allows you to make changes to the layout at any time, it is particularly important that you always perform a design rule check prior to generating final artwork.

Online DRC

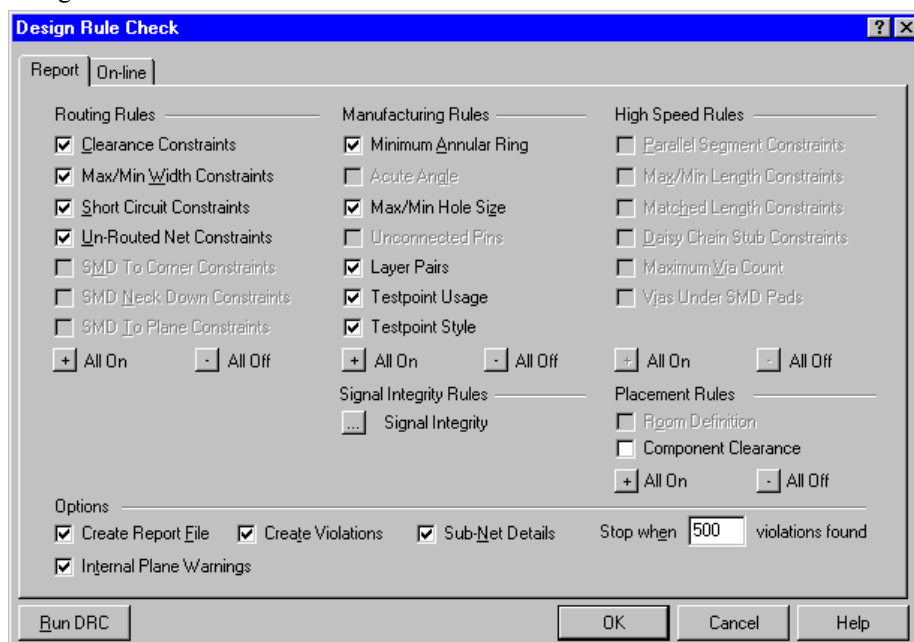
The online Design Rule Check feature is enabled in the Options Tab of the Preferences dialog. Turn this on when manually routing to immediately highlight violations as you work. You can configure the rules that are checked as you work in the On-Line Tab of the Design Rule Check dialog (select **Tools » Design Rule Check**).

♦ If a rule is not available (appears grayed out) it means that you have not created a rule of this type for the PCB Editor to test against. Select **Design » Rules** to configure the design rules.

If you are working on a design with large nets, which are routed using polygons and power planes, you may find that the on-line connectivity checker takes a long time to analyze these nets. To disable connectivity checking for these nets “hide” them from the connectivity checker by selecting **View » Connections » Hide Net**. You will be prompted to click on a point on the net.

Setting Up for a Batch Mode Design Rule Check

Select the **Tools » Design Rule Check** menu item to pop up the Design Rule Check dialog.



Design rule checking validates both physical and logical layout integrity prior to artwork generation

The batch DRC includes the following features:

Rules

Enable the set of rules that you wish to test. Refer to the Design Rules chapter for a description of each rule.

Create Violations

Enable this to highlight primitives with clearance or parallel segment violations in the current DRC Errors color.

◆ You can set the Panel to browse by Violations. Use this feature to quickly locate and correct violations.

Sub-Net Details

This option works with the Unrouted Net Rule. Enable this option if you require sub-net details.

◆ The Unrouted Net Rule should only be enabled when all connections have been routed, as a connection line is effectively an “open circuit”.

Stop When Found XX Violations

The PCB Editor will stop testing the design after finding this many violations.

Create Report File

Enable the Create Report File option to automatically create a DRC report file and open it in the text editor. If no filename is specified the report will be created with the name *filename.DRC*.

Running the Batch DRC

Once you have configured the Design Rule Check dialog press the Run Batch button to start the DRC process. Once the check is complete the report file will be opened in the text editor.

The DRC Report

The DRC report lists each rule that was tested, as specified in the Choose Rule Set to Check dialog (rules that are not present in the design are not tested). Each violation that was located is listed with full details of any reference information; such as the layer, net name, component designator and pad number; as well as the location of the object. Clearance, length and width errors are highlighted by outlining the object in the current DRC color.

Resolving Design Rule Violations

DRC reports can appear quite daunting to the new PCB Designer. The secret to keeping the process manageable is to develop a strategy. One strategy is to limit the number of violations that are reported. Set the Stop When Found feature in the Choose Rule Set to Check dialog to a small number, say 10. Another strategy is to run the DRC in a number of stages. If you find that the design contains a large number of violations, begin by enabling the rules one at a time. With experience you will develop a preferred approach to testing the various design rules.

Cross Probe Between the Report and the PCB

Use Cross Probing to work between the PCB file and the report file. Split the Design Window to show both documents, select the desired reference in the report file, and press the Cross Probe button on the text editor toolbar. The



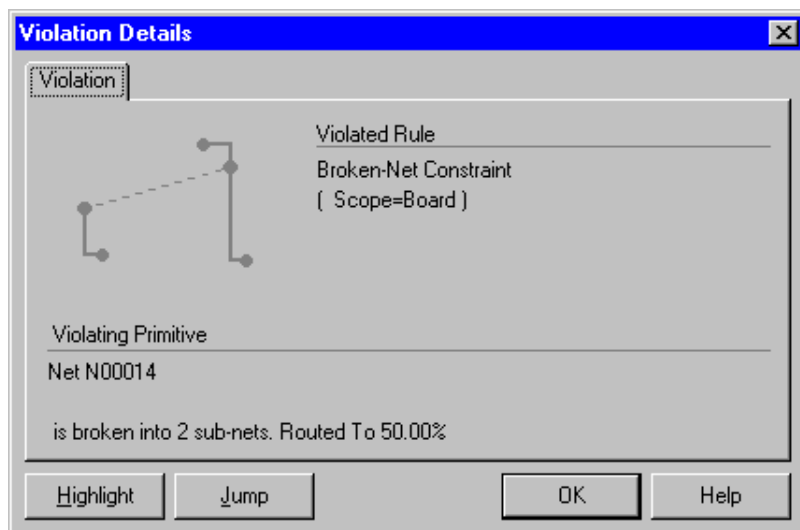
reference can be the component designator, the pin, an X Y location, or a string. The cross probed object will appear centered in the PCB window.

Jump Quickly to Objects in the Workspace

Use the Jump shortcuts to quickly locate components, pads, nets, error markers and specific locations. Press the J shortcut key to pop up the Jump menu.

Use the Browse for Violations Feature

Set the Browse mode in the panel to Violations. All the violations will be listed. Use the Details button to pop up the Violations Details dialog, which includes information about which rule has been violated, and the primitives that are causing the violation.



Tracking Down Broken Nets

A broken net is effectively an “open circuit”. Broken into two sub-nets indicates one break, broken into three sub-nets indicates two breaks, and so on. Use the following steps to help you find the open circuit.

1. Set the Browse method in the PCB Editor Panel to Violations.
2. Press the Jump button to change the zoom level to show the entire net, then use the Highlight button to “flash” the net.
3. To clearly identify the entire net select **Edit » Select » Net** (shortcut: S, N) and click on any point in the net.
4. To identify one sub-net select **Edit » Select » Connected Copper** (shortcut: S, P) and click on an object in the net.

Select » Connected Copper will only highlight the sub-net, whereas **Select » Net** will highlight the entire net. Use this difference to quickly locate the break in the net. You must disable the Extend Selection option in the Preferences dialog for this technique to work.

Resolving Extra Pins on a Net

If your report has one net broken into many sub-nets and a second net reporting extra pins that should be on the first net, there is probably a short between the two nets.

Checking the Signal Integrity

As PCB designs become more sophisticated, with higher clock speeds, higher device switching speeds, and higher density, the need to analyze the signal integrity before the design is manufactured becomes more pressing.

Protel 99 SE includes a sophisticated Signal Integrity Simulator, which can analyze the PCB layout and check that it functions within the design parameters, testing such things as overshoot, undershoot and impedance requirements.

The Signal Integrity Simulator uses the characteristic impedance of the traces, calculated through a transmission line calculator, and I/O buffer macro-model information, as input for the simulations. It is based on a Fast Reflection and Crosstalk Simulator which produces very accurate simulations, using industry-proven algorithms.

If the board fails any of the design requirements (specified as design rules), you can then run a reflection or crosstalk analysis from the PCB, to see exactly how it is behaving.

♦ If the design does not include a power plane the analysis will still be performed, but the results can not be considered accurate.

Setting up the Signal Integrity Design Rules

To set up the Signal Integrity Simulator design rules, Select **Design » Rules** from the menus, then click on the Signal Integrity Tab of the Design Rules dialog. Each rule includes a description of what that rule tests.

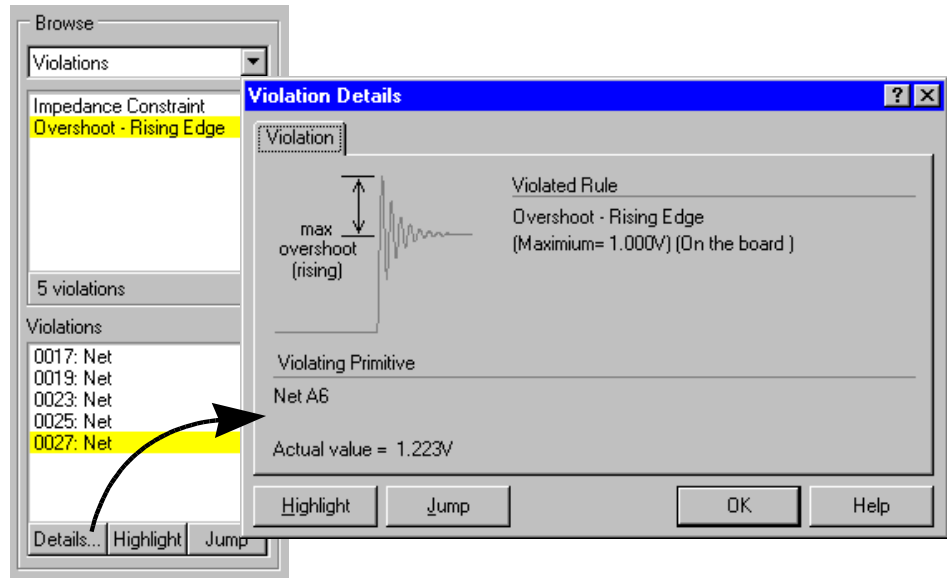
Once you have configured the signal integrity design rules select **Tools » Design Rule Check** from the menus to display the Design Rule Check dialog. Press the Signal Integrity button in the center of the dialog to enable checking of the signal integrity design rules.

♦ You must include a Layer Stack rule to be able to perform a signal integrity analysis.

Running a Signal Integrity Check

If there are signal integrity rules to be tested, when you press the Run DRC button in the Design Rule Check dialog the Signal Integrity Simulator will run. A small status box will appear, showing the progress of the analysis.

Once the analysis is complete set the PCB Editor Panel to browse by Violations. You can then examine each violation – click the details button to display details about the net name, and the actual value (eg, the amount of overshoot).

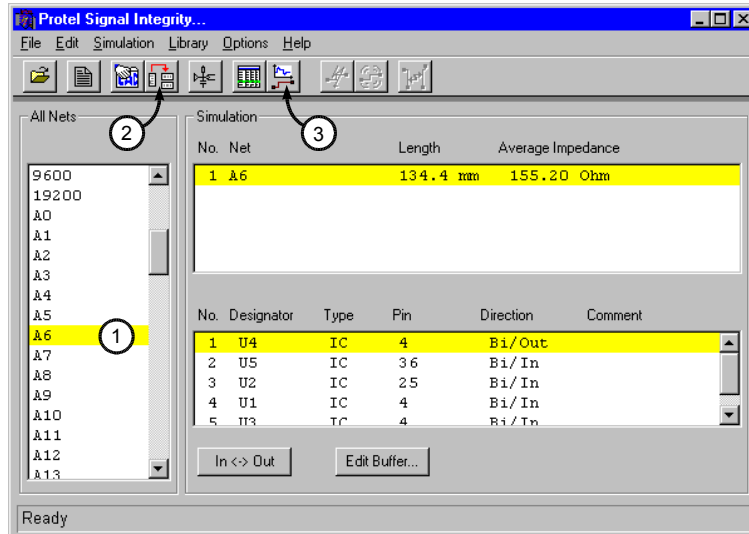


Set the Panel to browse by violations

- ◆ The DRC tests are worst-case – each net is simulated from all possible output pins, and the worst result is used.

Running a Reflection or Crosstalk Analysis

If the design fails one of the signal integrity design rule checks, you may want to do a reflection or crosstalk analysis. To run an analysis select **Tools » Signal Integrity** from the PCB Editor menus. The board will be analyzed, and the Protel Signal Integrity Analyzer will appear.



The Reflection and Crosstalk analyses are performed in the Protel Signal Integrity Analyzer

When the Protel Signal Integrity Analyzer appears it will include all the nets in the design, listed on the left. To analyze a net, or set of nets (use the numbers in the figure above to guide you):

1. Select the net in the list of nets.

It is not advisable to run a reflection analysis on a large number of nets – it will take a long time, and the waveform window will be crowded with waveforms, making it difficult to read any meaningful results.

2. Press the Take Over Selected Nets button on the toolbar.

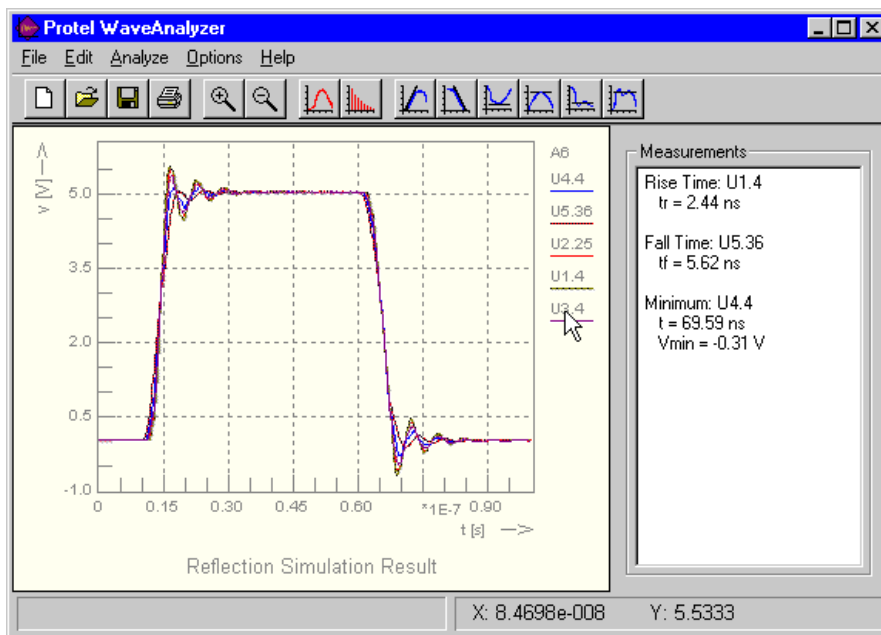
The net is listed at the top of the Simulation region, and each node in the net is listed below. Note that if the net includes a number of potential pins that could be driving the net (bidirectional pins), the first in the list is assumed to be driving. To change a node direction, select the desired node and click the In <-> Out button.

3. Press the Reflection Simulation button to run a reflection analysis on this net.



To run a crosstalk analysis you need to Take Over at least two nets from the list of nets. You then nominate one of these nets to be either the “aggressor”, or the “victim”. The aggressor is the net that the stimulus pulse is injected in, the victim is the net that receives the crosstalk. When this is done press the Crosstalk Simulation button.

Analyzing the Waveforms



Reflection and crosstalk waveforms are displayed in the Protel WaveAnalyzer

When you press the Reflection Simulation button the net is analyzed, and the reflection waveform displayed in the Protel WaveAnalyzer. You can perform a number of measurements directly from the waveforms, simply click on a node in the list to the right of the waveforms, and select an option from the Analyze menu.

- ◆ If you find that the waveform does not seem to match the result given during the design rule check (for example, the DRC gave an overshoot of 1.2 volts, but the waveform has minimal ringing), it is because the Out node that was used for the reflection analysis is not the worst-case node.

Running a Pre-analysis Net Screening

As well as running reflection and crosstalk analyses, you can also perform a net screening for signal integrity effects such as overshoot, delay, impedance, etc. Net screening produces a spreadsheet-like table of results, which can be sorted to quickly identify problem nets. To run a net screening, Take Over as many nets as required, and press the Net Screening button.



- ◆ Refer to the On-line help and the Readme.txt file for further information about the Signal Integrity Simulator.

Generating Reports

The PCB Editor can generate a number of reports that can help as you verify the status of the design. The following reports are available:

Selected Pins

All the selected pins are listed in the Selected Pins dialog. This provides a convenient way to verify the connections within a net. Use the Mask to narrow the list of pins being displayed. Press the OK button to generate a report file.

Board Information

General Tab

This Tab includes – a tally of each primitive type used in the design, the dimensions of the board (based on the most distant primitives in the workspace), the total number of pad and via holes and the number of clearance errors on the board.

Components Tab

Lists all components currently placed on both top and bottom layers. Includes designator and comment, if any.

Nets Tab

Lists all currently loaded nets, by name. Includes a tally of nets loaded.

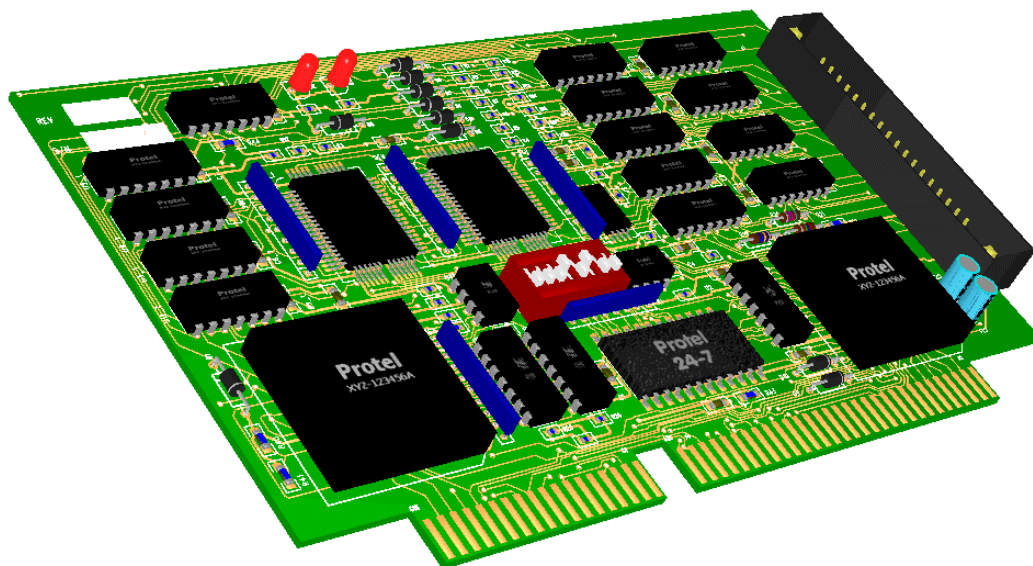
Pwr / Gnd

Press the Pwr/Gnd button to pop up the Internal Plane Information dialog. Each Tab lists the nets connecting to that plane. Select a net to display the pins on that net which are connected to this plane.

Netlist Status

For each net this report lists – the layers used for routing, and the routed net length.

3 Dimensional PCB Visualization



The 3D Viewer is a visualization tool that allows you to preview and print a 3D image of your PCB. The 3D Viewer is built around an OpenGL-based rendering engine, a standard graphics language supported by most graphics cards. It uses a run-time component modeling algorithm that uses the component designator prefix, footprint and outline shape to automatically select model and texture information and construct a suitable component model. Components that can not be recognized are automatically extruded.

Creating a 3D View of the Board

To create a 3 dimensional view of your board select **View » Board in 3D** from the PCB Editor menus. The board is analyzed and a 3D view is created in a new Window.

Changing the View of the Board

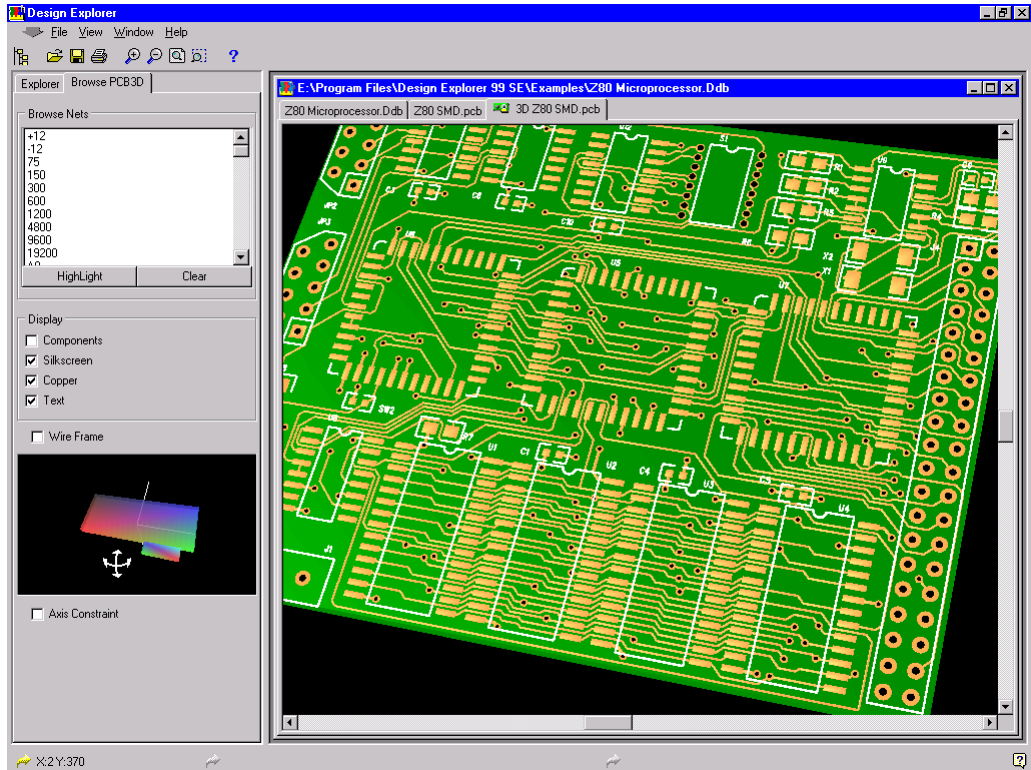
The 3D Viewer supports full rotational and zoom control, making it possible to display the board at any angle. The board can be rotated by clicking-and-dragging in the MiniViewer window in the panel.

The standard PCB Editor display shortcuts are also

◆ The rendering process can be canceled at any time by left-clicking on the 3D image as it is being redrawn.

supported – press the PAGEUP and PAGEDOWN keys to zoom in, the END key to redraw the view, and right-click and drag on the 3D image to display the slider hand and slide the 3D view around.

You can also selectively hide the components, silkscreen outlines, copper, and text strings. These options can be enabled in the panel, or the Preferences dialog (select **View » Preferences**).



Use the display options in the panel to control what is shown on the board

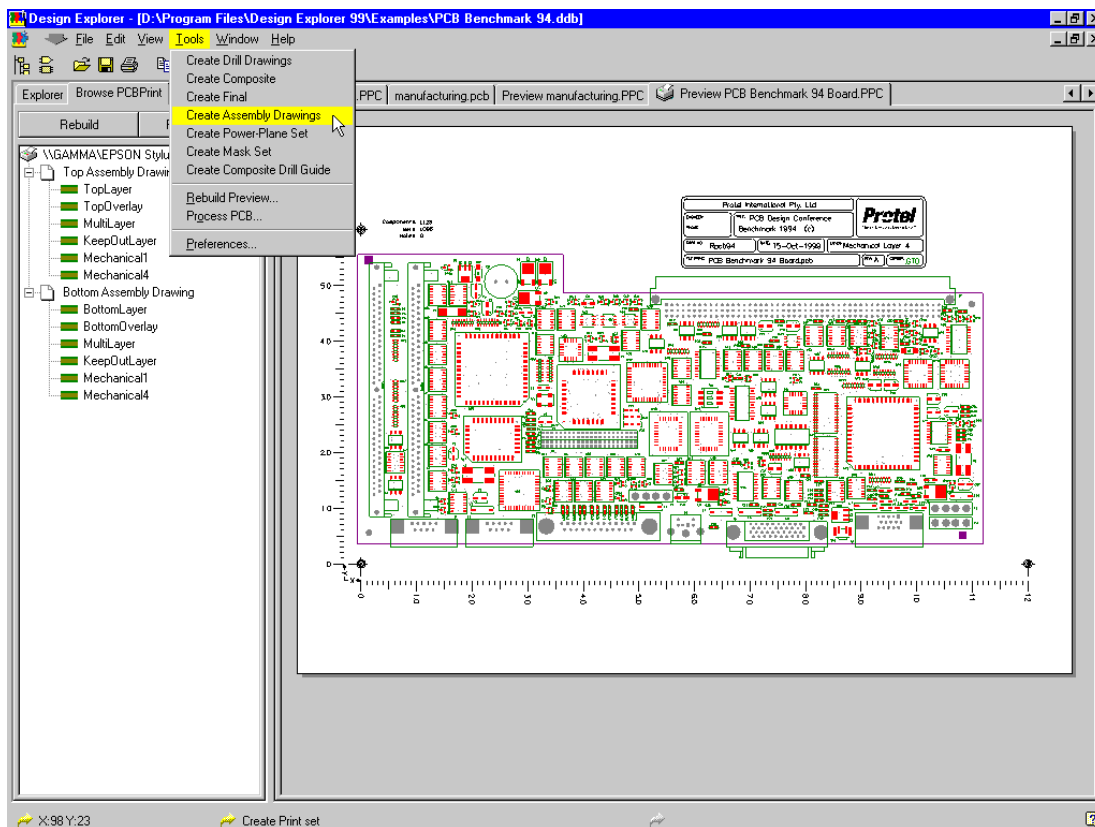
The Browse PCB 3D panel also includes a highlight feature, click on a net name and click the Highlight button to highlight that net on the board. There is also an Animate option, which flashes the net that is being highlighted. The highlight color and the animate feature are set up in the Preferences dialog.

Printing the 3D View

The 3D view can be printed by selecting **File » Print** from the 3D Viewer menus. This will print what is currently displayed in the 3D window.

Three print qualities are supported, Draft, Normal and Proof. The print quality is selected in the Preferences dialog (select **View » Preferences**).

Printing to a Windows Printing Device



Preview and configure the printouts before printing

An essential part of the design process is producing printed documentation about the PCB design. This could include a manufacturing drawing detailing the fabrication information, check plots for verifying the contents of each fabrication layer, and assembly drawings detailing component location information and loading order.

In Protel 99 SE printed output is created by preparing a preview of the required printouts, then printing them with the new Print/Preview feature. Using this approach you can define precisely what mix of PCB layers you want to print, set the scaling and orientation, and see exactly how it will look on the page before you print it.

Protel 99 SE's new print engine also supports printing the current screen area and copying the current preview to the Windows clipboard, making it easy to include PCB information in your documentation.

The Print/Preview feature works by creating a Power Print Configuration document (*.PPC). This PPC document details: which PCB is being previewed, the target printer, the set of printouts, and the PCB layers to include on each printout. When you open a PPC document this setup information is read, the PCB is analyzed, and the previews of the PCB are displayed in a separate Tab in the database window. You can then print the printouts as required.

Because the actual PCB data is not stored in the PPC document it must be extracted from the PCB when you create, modify, or open a PPC document. This analysis happens automatically – if you prefer you can disable automatic rebuilding in the Preferences dialog and then use the Rebuild button (when you change the preview configuration), or Process PCB button (when you modify the PCB) in the Browse PCBPrint panel to update the previews.

Setting up a Print Preview

To perform a preview of a PCB select **File » Print/Preview** from the PCB editor menus. A PPC document will be created and opened, displaying the PCB as it will appear on the printed page(s). The default PPC document name for a PCB called *MyDesign.PCB*, is *Preview MyDesign.PPC*. Note that you can rename this document in the same way you rename any document in the Design Explorer.

♦ If a Power Print document with the default name of *Preview MyDesign.PPC* already exists when you select **File » Print/Preview** it is automatically opened. To create a second PPC document rename the default PPC document.

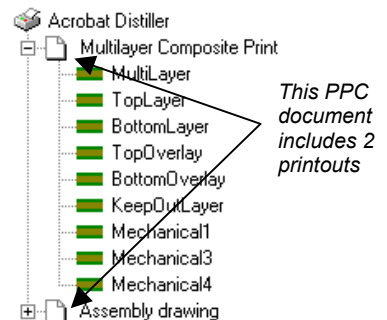
Once the preview appears, click on the **Browse PCBPrint** Tab in the Design Manager to display the current configuration of printouts. The default PPC is configured for a single composite printout, targeting your default Windows printer.

What is a Printout?

A Printout is a set of one or more PCB layers that will be printed as a single print job. Depending on the scaling, this may be on a single piece of paper, or tiled (spread) over a number of pieces of paper.

Each printout is represented as a page icon in the **Browse PCBPrint** panel.

Any combination of PCB layers can be included in a printout, and you can define as many printouts as you like in each PPC document.



Changing the set of Printouts

When you select **File » Print** from the PCB editor menus and the preview pages appear, the default configuration is a *composite* printout. A composite printout is a drawing with all the main layers overlaid on one another, simulating the real PCB.

There are a number of predefined printout sets, click on the **Tools** menu to display the list. When you select a different printout set the current configuration is replaced with the new configuration. You can easily create new printouts, select **Edit » Insert Printout** to add a new printout to the current Print/Preview document. By default a new printout includes the top layer.

Specifying the Layers in a Printout

To display the set of layers in a printout click once on the small + symbol next to the printout icon. The order the layers are displayed in the panel is the order they are rendered on the actual printout.

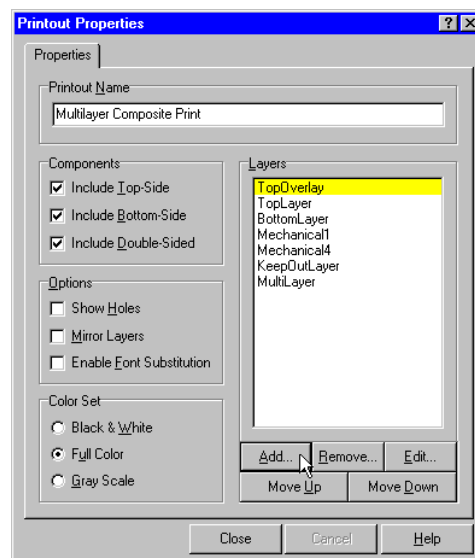
To add and remove layers from a printout right-click on the printout icon and select **Properties** from the small floating menu. The Printout Properties dialog will pop up. The Layers region of the dialog displays the current set of layers that will be included in this printout. Use the buttons at the bottom of the Layers region to modify the layer set.

To add a new layer:

1. Click the Add button to display the Layer Properties dialog.
2. Select the layer you would like to add in the drop down Print Layer Type list.
3. Set the display mode for the primitives as required. Refer to the topic, *Configuring the Layer Properties*, for more information on the primitive display mode.
4. Click OK to close the Layer Properties dialog.

The new layer is added at the bottom of the layer list. This means that this layer will be drawn first in the printer's memory when the image is rendered. Each layer above is then rendered on top in turn. Use the Move Up and Move Down buttons to change its position in the render order.

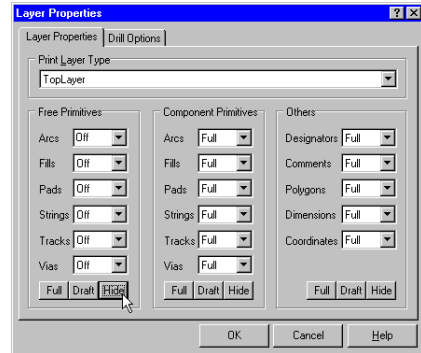
◆ As well as adding mechanical layers individually to a printout, you can also automatically include them in all printouts. Enable the required mechanical layers in the Mechanical Layers Tab of the Properties dialog.



Add layers to a printout in the Printout Properties dialog

Configuring the Layer Properties

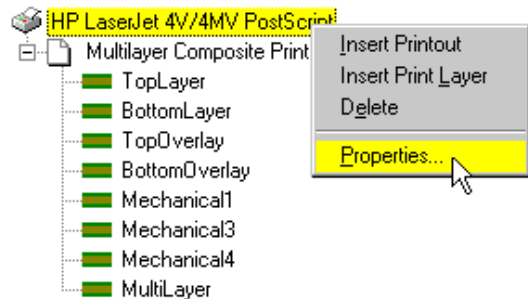
As well as being able to include any PCB layer on a printout, you can also control the way the primitives on that layer are displayed on the printout. This gives you complete control over what appears on the printed page. For example, consider an assembly drawing which includes the component overlay, the top layer (for the surface mount pads) and the multi-layer (for the thru-hole pads). The top layer would be configured to display the component primitives, so the surface mount pads were visible, and hide the free primitives, so the routing is not visible. The multi-layer would also be configured to display the component primitives, so the thru-hole component pads were visible, and hide the free primitives so that the vias were not visible.



Control the display of primitives in the Layer Properties dialog

Changing the Target Printer

When you create a new PPC document it is automatically configured to target the default Windows printer. To change the target printer, select **File » Setup Printer** from the Print/Preview menus. Alternatively, right-click on the printer icon in the panel and select **Properties** from the popup menu that appears.



Right-click on the printer icon to select a different printer

Setting the Paper Orientation, Scaling and other Printer Options

Printing options such as paper orientation, scaling and margins are configured in the PCB Print Options dialog (right-click on the printer icon and select **Properties**).

There are 3 default scaling modes, click the Print What arrow to display these.

Standard Print – print the board at the scaling defined by the current Print Scale amount.

Whole Board on Page – automatically scale the board to fill the page.

PCB Screen Region – fit the current screen region onto the page. Note that any spare space on the page will be filled as well.

Printing

Once the printouts are correctly configured in the preview window you are ready to print. There are a variety of printing options available, click on the **File** menu to display these different options. The options include:

Print All – select this option to print all printouts in the current PPC document. Each printout is sent to the printer as a separate print job, with the same name as the printout.

Print Job – select this option to print all printouts in the current PPC document, with all printouts sent in the same print job. The print job has the same name as the PPC document.

Print Page – print the current page. If the printout is tiled over a number of pages a dialog will appear prompting you to type in the page number, or the page range. As a reference, each page of a multi-sheet tile printout includes a small red preview page number at the top left of the page. The numbers are not included on the printout, they can also be turned off by disabling the Preview Page Numbers option in the Preferences dialog.

Print Current – print all pages in the current printout.

Tip: Printouts can be reordered by clicking and dragging the printout in the PrintPCB Panel.

Copying from the Preview Window to other Applications

Select **Edit » Copy** from the Print/Preview menus to copy the current printout to the Windows clipboard. You can then paste the image into other applications that support the Windows clipboard, such as Microsoft Word.

Note: The image is copied from the preview to the clipboard as an enhanced metafile. You can also export the printouts as either standard or enhanced WMF files.

Exporting the Printouts as WMF files

To export the current set of printouts as WMF files select **File » Export** from the Print/Preview menus. The Export Options dialog will pop up. The WMF files are saved in a folder in the design. There are 2 output folder options: overwrite folder, where the files are written into the same folder each time you export; or create time-stamped folder where the files are written into a new folder each time you export.

You can also save a copy of the PCB into the folder each time you export by enabling the Save a Copy of the PCB File option. This will save a copy of the last saved version of the PCB.

To export the files directly onto the hard disk enable the Export Copy To option. Use the enhanced metafile option to specify if the metafiles are to be written as standard metafiles (16 bit), or enhanced metafiles (32 bit).

Color, Font and other Preferences

To set the layer color, font substitutions and other Print/Preview options select **Tools » Preferences** from the menus to display the Preferences dialog.

Color and Gray Scale Assignments

Each layer in a printout can be assigned a different shade of gray (or color). To change the gray shade for a layer, click once on the gray box next to the layer name, then select the new gray shade from the dropdown palette. To change the color for a layer, click once on the color box next to the layer name to display the **Choose Color** dialog.

Note: Use the Color Set options in the Printout Properties dialog to select Gray Scale or Color for each printout.

Font Options

Each of the 3 PCB Editor fonts can be substituted for a different Windows font on each printout. Use these options to specify the font that you want substituted for each of the PCB Editor's 3 fonts (Default, Serif, and San Serif).

Once the substitute fonts are specified and substitution is enabled, open the Printout Properties dialog to enable font substitution for that printout.

Overlap for Tiled Print

This option defines the amount of printed information that is duplicated on adjacent pages when a printout is tiled (printed over a number of pages).

Automatic Rebuild Option

Each time you change the setup options in one of the Print/Preview dialogs the data is re-analyzed to ensure that the previews are accurate. Disable this option to stop automatic rebuilding.

Select **Tools » Rebuild Preview** (or click on the Rebuild Preview button in the panel) to force a rebuild if you have changed a setup in one of the dialogs. Select **Tools » Process PCB** to force a rebuild if you have modified the PCB (or click on the Process PCB button in the panel).

Mechanical Layers

As well as adding mechanical layers individually to a printout, you can also automatically include them in all printouts.

Generating the Manufacturing Files

Completing the PCB layout is only the first part of the process that culminates in the fabrication and assembly of your PCB. The link between your design and the finished board are the print, Gerber and NC drill fabrication files, as well as the Bill of Materials, testpoint report, and pick and place assembly files. Apart from printed output, all the PCB manufacturing output files are generated by the CAM Manager.

For background information on photoplotting refer to the topic *What is Artwork?* later in this chapter. For information on printing from the PCB editor refer to the topic *Printing to a Windows Printing Device* earlier in this supplement.

Working With the PCB Manufacturer

When you start designing, you should have a clear idea of the output requirements of the PCB technology and production methods you will be using.

If you intend to use the services of a plotting bureau or PCB manufacturer take the time to consult with them before you start generating artwork. Bureaus and manufacturers often have specific requirements that must be reflected in the files or artwork that you submit. For example, you may wish to either “step and repeat” or panelize your Gerber files for efficient quantity fabrication.

To accomplish this, you have to know the film size accepted by the photoplotter, clearance requirements, etc, as well as the manufacturing tolerances involved. Planning for Numeric Control (NC) drilling, requires similar consideration.

In some instances, the bureau or fabrication facility will prefer to work directly with “raw” Gerber files (or even PCB files) rather than panelized Gerbers. Understanding these requirements will help you to plan the entire design process for efficient and trouble-free completion.

Generating Manufacturing Output with the CAM Manager

The CAM Manager is an independent editor that operates in the Design Explorer, alongside the PCB Editor, Schematic Editor, and other editors. The CAM Manager gives you total control over the setup and creation of the manufacturing output files, including Gerber, NC drill, pick and place, Bills of Materials, testpoint reports and DRC reports. The setups are saved in the design in a CAM Output Configuration document (.CAM) and can be modified at any time. CAM documents can also be copied from one design to another, making it easy to transfer your preferred manufacturing output file configurations.

The CAM Manager includes a CAM Output Wizard, which takes you through the steps to create each of the different output setups.

All enabled outputs can be generated at any time with a single command. The output files that are generated are written to a separate CAM Outputs folder, which can be time-stamped if required. There is also an option to automatically export the CAM outputs directly to a disk drive.

The CAM Manager supports the following output types:

- Gerber files
- NC Drill files
- Testpoint reports
- Pick and Place files
- Bill of Materials (BOM)
- Design Rule Check (DRC) reports

Creating a new CAM Document

To create a new CAM document for the current PCB select **File » CAM Manager** from the PCB Editor menus. A blank CAM document will be created, and the Output Wizard will automatically start. The Wizard can be used to create an output setup for each type of supported output. You can also create the various output setups directly by selecting the required type in the **Edit** menu.

How the CAM Documents are Named

When you select **File » CAM Manager** from the menus the new CAM document is automatically named *CAM Outputs for MyDesign.CAM*, for a PCB named *MyDesign.PCB*. If a CAM document with this name already exists then this document is reopened. To create a second CAM document, first rename the original CAM document, then select **File » CAM Manager** from the PCB Editor menus.

Adding Output Setups to a CAM Document

There are 2 ways to add a new output setup to the current CAM document, you can use the CAM Output Wizard, or select the required output type in the CAM Manager **Edit** menu.

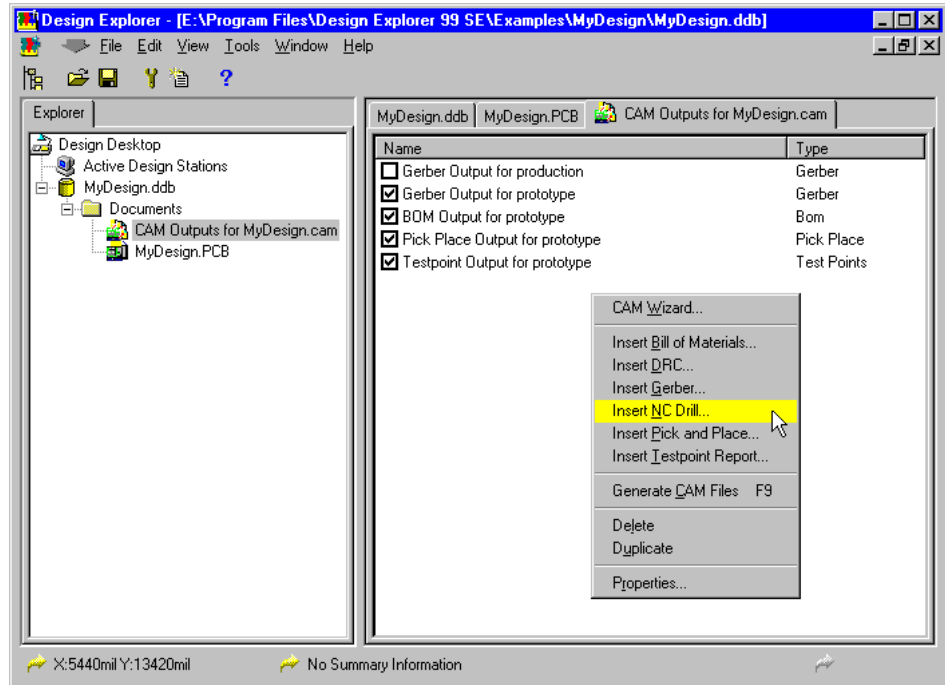
Using the CAM Output Wizard

The Wizard can be run by selecting **Tools » CAM Wizard** from the menus, or by right-clicking in the CAM document and selecting **CAM Wizard** from the floating menu. All of the output types can be set up from the Wizard, select the required type on the second page of the Wizard. If you are unsure of an option as you step through the Wizard leave it on the default setting – any setting can be changed later by double-clicking on the output setup in the CAM document to display the setup dialog. You can then use the What's This help icon (at the top right of each dialog) for information on a specific option in the different setup dialogs.

Directly Adding an Output Setup

Each CAM document can store multiple output setups, including multiples of the same kind of output setup. For example, one CAM document might include the Gerber and NC drill output setups for the prototype run, as well as a second set of output setups for the production run.

To add a new setup select the required type from the CAM Manager's **Edit** Menu, or from the right-click menu. The Setup dialog for that type will appear, configure the setup as required and click OK to close the dialog.



Right-click in the CAM document to add a new output setup

Modifying, Duplicating and Deleting an Output Setup

Double-click to modify an existing output setup. To delete or duplicate a setup right-click and select the appropriate choice from the floating menu.

Configuring the CAM Generation Options

CAM options are configured in the CAM Options dialog, select **Tools » Preferences** to display the dialog. Each option is described below.

CAM Output Folder

CAM outputs are written to a sub-folder that is created in the same folder as the PCB. If the Overwrite folder option is enabled, the folder will be called *CAM for MyDesign* (for a PCB called *MyDesign.PCB*), and the outputs written to this folder each time you select **Generate CAM Files** from the menus.

If the Create Time-Stamped Output Folder option is enabled the date and time is appended to the end of the folder name, and a new folder created each time you select **Generate CAM Files** from the menus.

CAM Output Files Destination

If the One Folder For All Outputs option is enabled all output documents are written to the main CAM folder. If the Separate Folder for Each Output Type option is enabled, sub-folders are created within the main CAM folder in which to store the separate output types.

Archive PCB File Option

If the Save a Copy of the PCB option is enabled the last-saved version of the PCB is copied to the CAM folder when the CAM outputs are generated.

Export CAM Outputs

Enable the Export CAM Outputs option to automatically export the output files to the specified location on the hard disk.

Generating the Output Files

Output can be generated for any of the current output setups at any time. To generate output, first enable the setups you require by clicking on the small square next to the setup name in the CAM document. When the required outputs are enabled select **Tools » Generate CAM Files** from the menus (shortcut: F9). The outputs will be created according to the current setups in the CAM Options dialog.

Gerber File Output Setup

What are Gerber Files?

A PCB is fabricated as a series of layers that the manufacturer assembles into a board through a variety of chemical and mechanical processes. To fabricate each physical layer in the PCB the manufacturer uses an image of that layer – this image is referred to as a phototool. A phototool is a piece of clear film, with black lines, circles and other shapes forming exactly the same patterns as the content of that layer in Protel 99's PCB Editor.

Once the PCB design process is complete and the design has passed all the design rule checks, the Gerber files are generated, one for each layer needed in the fabrication process. The Gerber language is the standard language format used to transfer PCB layout data from the PCB design software to the phototool creation process. These Gerber files are then sent to the manufacturer, who loads them into a photoplotting machine and creates the phototools.

Each phototool is created by exposing the film to build up the image required for that layer. The information needed to form the image includes the shape and size of the objects on that layer, and the coordinates of these objects. The shapes are specified in the Gerber file as apertures, and typically these apertures are created from the board and included in each Gerber file, when they are referred to as embedded apertures. If the apertures are not embedded then they must be stored in a separate aperture file and supplied with the Gerber files.

Setting up the Gerber File Options

To include a Gerber setup in a CAM output configuration document select **Edit » Insert Gerber** from the CAM Manager menus to pop up the Gerber Setup dialog. Click on the What's This help icon at the top right of the dialog for detailed information about each of the options in this dialog.

Note that when you enable the Embedded Apertures option in the Apertures Tab of the Gerber Setup dialog the aperture list is created automatically, each time you generate the Gerber files. The apertures are then embedded in the Gerber files, according to the RS274X standard. This feature means that you do not need to worry if the current aperture list includes all the required apertures – unless your PCB manufacturer does not support embedded apertures it is highly recommended that you use this option. For more information on apertures refer to the topic *What is Artwork?* later in this chapter.

To ensure that the finished PCB meets your design and manufacturing requirements it is important that you contact the fabrication house and discuss their requirements before generating the Gerber files.

Some of the requirements you should discuss include:

Any apertures restrictions – most modern photoplotters are raster plotters which can accept any size aperture. Generally they also accept Gerber files with embedded apertures. In this situation enable the Embedded Apertures option in the Apertures Tab of the Gerber Setup dialog.

Mask expansions – mask expansions are required for the solder and paste mask layers. The solder mask layer defines where the manufacturer must apply a thin layer of solder to the bare copper on the board, typically on component pads. The paste mask layers define where solder paste is applied to the board during the assembly process, and is normally only required for surface mount components. Solder and paste mask expansions are specified in the Manufacturing Tab of the Design Rules dialog in the PCB Editor.

Power plane clearances – if the design includes internal power planes the manufacturer will specify the clearance required for thru-hole component pads and vias that do not connect to the plane. The physical connection parameters used for pads and vias that do connect to a plane must be set to suit the requirements of the design. Power plane clearances and connection styles are also set up in the Manufacturing Tab of the Design Rules dialog in the PCB Editor.

The units and format of the Gerber files – the units can be either inches or millimeters. The format specifies the precision of the coordinate data, this must be selected to suit the placement precision of the objects in the PCB workspace. For example, the 2:3 format has a resolution of 1 mil (1 thousandth of an inch). If your design has objects placed on a sub 1 mil grid then this format will not be adequate. Conversely, the higher precision formats may be more difficult and expensive to photoplot and manufacture.

If the plots should be centered on the film – the Gerber data can be automatically centered on the specified film by enabling the Center Plots On Film option in the Gerber Setup dialog. Note that Gerber coordinates are referenced from the absolute origin if the Center Plots on Film option is turned off.

The drilling requirements – the drilling information is normally supplied in the form of NC drill files, refer to the *NC drill Output Setup* topic for more information.

File Extensions used to Identify each Gerber File

When you generate the Gerber output a series of files are created, each one corresponding to one of the layers enabled in the Gerber setup. These files are then loaded into a Gerber photoplotter, which produces the necessary phototools for PCB manufacture.

Each Gerber file is given the name of the PCB document, with a unique extension that identifies that layer and plot type. For example, the Top Layer Gerber file for a PCB called *MyDesign* will be saved as *MyDeisgn.GTL*, to indicate "Gerber Top Layer". Because each design normally generates numerous Gerber files, these extensions help identify each file.

We recommend that you follow this convention which conforms to general industry practice. The following table shows the extensions that are used:

Top Overlay	.GTO
Bottom Overlay	.GBO
Top Layer	.GTL
Bottom Layer	.GBL
Mid Layer 1, etc	.G1, .G2, etc
Power Plane 1, etc	.GP1, GP2, etc
Mechanical Layer 1, etc	.GM1, .GM2, etc
Top Solder Mask	.GTS
Bottom Solder Mask	.GBS
Top Paste Mask	.GTP

Bottom Paste Mask	.GBP
Drill Drawing	.GDD
Drill Drawing – Top to Mid 1, Mid2 to Mid 3, etc	.GD1, GD2, GD3, etc
Drill Guide	.GDG
Drill Guide – Top to Mid 1, Mid 2 to Mid 3, etc	.GG1, GG2, GG3, etc
Pad Master, Top	.GPT
Pad Master, Bottom	.GPB
Keep Out Layer	.GKO
Gerber Panels	.P01, .P02, etc

NC Drill Output Setup

Numeric Control (NC) drill file requirements are configured in the NC Drill Setup dialog. This dialog appears when you edit an existing NC drill setup in a CAM output configuration document, or when you select **Edit » Insert NC Drill** from the CAM Manager menus to add a new NC drill setup to the current CAM output configuration document.

NC drill files are used to program a drilling machine with the information required to drill the holes in the blank PCB during the PCB fabrication process. Drill files specify the drill sizes, drill tool assignments and hole locations. Drill hole coordinates are referenced from the user-defined relative origin.

Three types of drill files are produced:

MyPCB.DRR - drill report, detailing the tool assignments, the hole sizes, hole count and the tool travel.

MyPCB.TXT - ASCII format drill file. For a multi-layer PCB which uses blind and/or buried vias a separate drill file for each layer pair is created, with a unique file extension.

MyPCB.DRL - binary format drill file. For a multi-layer PCB which uses blind and/or buried vias a separate drill file for each layer pair is created, with a unique file extension.

Setting the NC Drill Options

The NC Drill files should be created with the same format, or precision, as the Gerber files. For example, if the Gerber setup has been configured to use the 2:4 format, then the corresponding NC drill setup should use the same format. Click on the What's This help icon at the top of the NC Drill Setup dialog for information on a specific feature in the dialog.

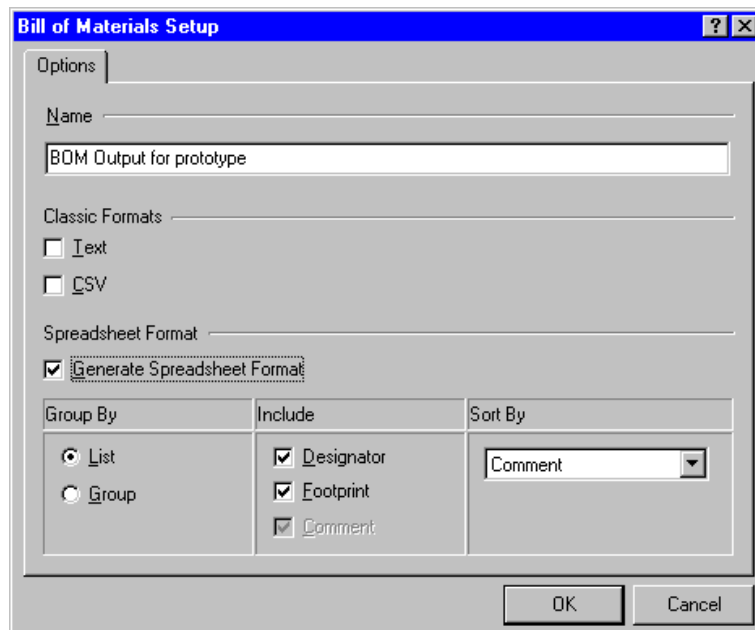
Bill of Materials Output Setup

Bill of Materials (BOM) file requirements are configured in the Bill of Materials Setup dialog. This dialog appears when you edit an existing Bill of Materials setup in a CAM output configuration document, or when you select **Edit » Inset Bill of Materials** from the CAM Manager menus to add a new Bill of Materials setup to the current CAM output configuration document.

Three BOM formats are supported, Text, CSV (Comma Separated Values) and Spreadsheet. The Text and CSV formats group the BOM listing by component comment and value. The Spreadsheet option allows you to specify the format of the BOM.

Setting the Bill of Materials Options

Enable the output format that you want the BOM created in. More than one format can be enabled, each of the 3 BOM formats is given a different file extension. Click on the What's This help icon at the top of the Bill of Materials Setup dialog for information on a specific feature in the dialog.



Set up the Bill of Materials output requirements in the Bill of Materials Setup dialog

Pick and Place Output Setup

Pick and place file requirements are configured in the Pick and Place Setup dialog. This dialog appears when you edit an existing pick and place setup in a CAM output

configuration document, or when you select **Edit » Insert Pick and Place** from the CAM Manager menus to add a new pick and place setup to the current CAM output configuration document.

Pick and place files are used to program machines that automatically load components onto the PCB during assembly. They are called pick and place because they *pick* the required component from a feeder tube and *place* it in the correct location on the PCB. Once a PCB has the components loaded it is then passed through another machine that solders all the connections.

The pick and place file includes the following information for each component:

- Designator
- Footprint
- Location - expressed in 3 formats; by geometric center, component reference point, and pad 1 location
- Side of board
- Rotation
- Component comment

Pick and place component location files can be generated in Spreadsheet, CSV and Text formats, and in imperial or metric units. Pick and place coordinates are referenced from the user-defined relative origin.

Setting the Pick and Place Options

Enable the output format that you want the Pick and Place file created in. More than one format can be enabled, each of the 3 formats is given a different file extension. Click on the What's This help icon at the top of the Pick and place Setup dialog for information on a specific feature in the dialog.

Testpoint Report Output Setup

Testpoint report requirements are configured in the Testpoint Setup dialog. This dialog appears when you edit an existing testpoint setup in a CAM output configuration document, or when you select **Edit » Insert Testpoint** from the CAM Manager menus to add a new testpoint setup to the current CAM output configuration document.

Testpoints are identifiable points on the board that are used to analyze the board. There are typically 2 types of analyses carried out on a board: bare-board testing, to test for shorts between the routed tracks; and assembled board testing, which tests for correct performance of the finished board. These tests are carried out by placing the board on a 'bed of nails', where testpoint probes (nails) contact the testpoints on the board.

A PCB testpoint is simply a pad or via that has one of its testpoint attributes enabled. Pads and vias can be specified to be a top layer, bottom layer, or both top and bottom layer testpoint.

Testpoints can be defined manually, searched for by the Testpoint Find feature, or placed by the autorouter. They are placed according to, and tested against, the Testpoint Style and Testpoint Usage design rules. The Testpoint Style rule reports testpoints that do not comply with the required physical parameters (size, etc), and the Testpoint Usage rule reports those nets which have not had a testpoint correctly assigned. Refer to the *Design Rules* chapter for information on setting up these rules.

The testpoint report is then used to find all pads and vias that have one or both of their testpoint attributes enabled. The testpoint report includes:

- Net name
- Testpoint name
- X and Y coordinates – referenced from the user-defined relative origin
- Side of board
- Hole size
- Testpoint type – thru-hole or surface layer

Setting the Testpoint Options

Enable the output format that you want the Testpoint report created in. More than one format can be enabled, each of the 4 formats is given a different file extension. Click on the What's This help icon at the top of the Testpoint Setup dialog for information on a specific feature in the dialog.

Design Rule Check Output Setup

Design Rule Checking requirements are configured in the DRC Setup dialog. This dialog appears when you edit an existing DRC setup in a CAM output configuration document, or when you select **Edit » Insert DRC** from the CAM Manager menus to add a new DRC setup to the current CAM output configuration document.

DRC testing and reporting can be performed directly in the PCB workspace by selecting **Tools » Design Rule Check** from the menus. The CAM Manager DRC report runs the same testing routines, it is included to simplify the process of preparing the necessary output files when the design phase is complete and the design is ready for release to manufacture. For detailed information on design rule checking refer to the chapter, *Verifying the PCB Design*.

Setting the DRC Options

Enable the rules that you want tested in the DRC Setup dialog. Click on the What's This help icon at the top of the DRC Setup dialog for information on a specific feature in the dialog.

Transferring the CAM Document to Another Design

CAM documents can be copied from one design to another like any other design document. Close the CAM document prior to copying, right-click on its icon in the tree or the folder it is stored in to display the floating menu, and select **Copy**. Right-click in the target folder in the second design and select **Paste** from the floating menu. Like the Windows File Explorer you can also use drag and drop to copy from one design to another.

When you copy a CAM document from one design to another you may need to reset the target PCB. This can be done at any time when a CAM document is open by selecting **Tools » Set Target Board** from the CAM Manager menus.

Note: If you do change the target PCB and you are not using the Embedded Apertures option for the Gerber setup, you must ensure that the aperture list is appropriate for the new PCB before generating the Gerber files.

What is Artwork?

Artwork is the name given to the pieces of film that are used by the PCB manufacturer to fabricate the PCB. These pieces of film, one for each fabrication layer, are normally created by a “photoplotter”, from the Gerber files generated from the PCB.

Artwork can also be generated by high-resolution Postscript “imagesetters”, that are used by graphic design and typesetting bureaus. These machines are capable of producing film positives at resolutions at 2540 dpi (dots per inch) or higher.

However, users should be aware that there are some limitations to using this approach for PCB artwork. The resolution of these systems does not necessarily translate into positional accuracy or linearity, particularly when measured over a large area. There are also film size restrictions. Postscript output files can be generated by the PCB Editor’s Power Print feature.

Photoplotted Artwork

Gerber format photoplotting provides the highest resolution output and is generally considered the method of choice for production PCB tooling as it provides the best quality artwork for board production. Photoplots will be required when the design is either large in total area, or of high-density with fine line details. Gerber outputs are generated by the PCB Editor’s CAM Manager.

About Photoplotters

Photoplotters are similar to pen plotters in many ways, the primary difference being that photoplotters use light to plot directly onto photosensitive film. The many advantages of this approach has led to the widespread adoption of photoplotting in the electronics industry.

Because the etching of printed circuit boards is generally based upon photographic techniques, the production of positive and negative photo-tools (or films) is an inherent

part of the process. When the original artwork is a pen plot, a number of intermediate steps have to be performed to produce the final tools. Pen plots are generally plotted at least 2:1 scale to achieve reasonable accuracy and then photographically reduced.

Photoplotters provide sufficient accuracy to generate a precision 1:1 plot in a single operation. Photoplotting bureau services are widely available and all designers should carefully consider its advantages. To make the best use of photoplotting, it's helpful to understand some key concepts.

Vector vs. Raster Plotters

Photoplotters fall into two general categories, vector and raster.

Vector plotters generally use an aperture “wheel” or “slide” to create the combination of “flashes” and “strokes” to “draw” an image. These make images in much the same way as pen plotters. They select a drawing instrument (or aperture) and describe a vector in the drawing space. The result can be seen as a line the width of which is defined by the aperture. Apertures are a collection of defined shapes which allow the plotter to plot varying track widths, pad shapes, etc. Flashes occur when there is no movement of the light source, strokes occur whenever there is movement while the light source is on. Some plotters use separate apertures for strokes and flashes in order to maintain consistent exposure. Others control the light intensity – all apertures serving for both uses.

Raster plotters do not use a system of fixed apertures. They read the Gerber file, storing an “image” of the whole plot, which is then scanned onto the film, line-by-line, not unlike a television image. Raster photoplotters can synthesize a virtually unlimited variety of different apertures, providing a great amount of flexibility to the designer.

Some photoplotters use the Postscript language. Photoplot files for these devices are prepared using an appropriate Postscript driver. For information about Postscript printing, see the *Postscript Printing Tips* topic later in this chapter.

You will want to know something about the “target” photoplotter, in order to make efficient use of its capabilities when you design.

- ◆ Contact your photoplot bureau before generating any photoplots. Matching available plotting options at the edit level can save considerable time and expense when generating Gerber phototools.

Photoplotter Languages

Nearly all photoplotters are controlled by a vector-based plotting language, developed specifically for this task, generically referred to as “Gerber” – a registered trademark of the Gerber Scientific Company. This language has become an industry standard (also known as RS-274). While the language has evolved to accommodate changes in both plotting equipment and design tasks, a number of potential difficulties and limitations must be considered by the designer when planning a job for Gerber output.

A Gerber format file describes a plot as a series of draft codes (or commands) and coordinates. The draft codes control the aperture to be used, turning the light “on” or

“off” and so on. Coordinates define the position of the various flashes and strokes on the plot. This information is stored as an ASCII text file.

The structure of Gerber files can vary due to a number of “optimizations” that have been added to the format over time, to address the changing capabilities of plotting hardware. Your photoplot bureau may need to know details regarding Protel’s use of Gerber format, so we have described it in some detail below.

Protel Gerber files are divided into individual commands, followed by carriage return code then a line feed code. Each record is terminated by the character “*”.

The records may refer to an absolute location or a draft code which changes apertures. Thus a record might be “X800Y775*” which instructs the plotter to move to a particular coordinate or “D16*” which is a draft code or command, such as a new aperture selection.

Some plotters reserve draft codes D01–D09 for uses other than aperture selection, for example:

- D01 Turns the light source on.
- D02 Turns the light source off.
- D03 Flashes the light source.

On some older plotters the special code “G54” needs to be sent before each change of aperture code. The last Gerber record is terminated by the special record M02*, which is followed by another block, containing the character “hex 08,” then 509 “spaces” (hex 20), then a carriage return and a line feed.

You can inspect any Gerber file with a text editor or word processor capable of loading an “un-formatted” text file.

About Apertures

All Gerber format photoplotters use apertures. Apertures describe the available tools used to draw on film. In the case of a vector plotter these apertures correspond to various sizes and shapes of holes in an aperture wheel or slide. Light is projected through these apertures onto the film emulsion.

Raster plotters are not limited to a set of specific aperture sizes and shapes. Raster imaging systems interpret the aperture information in the generated Gerber file and the entire plot image is synthesized and represented by a bitmap and plotted line-by-line, not unlike a television image.

Using Apertures

The apertures that will be used to translate your PCB file into a set of Gerber files are stored in a file with the extension .APT. Apertures can be regarded like plotter pens. Aperture descriptions include a shape, such as a 50mm square, and use – flash, stroke or anything (either flash or stroke).

Before you can generate a Gerber file, you can either load an aperture file that matches the capabilities of the target plotter, or you can let the PCB Editor automatically create

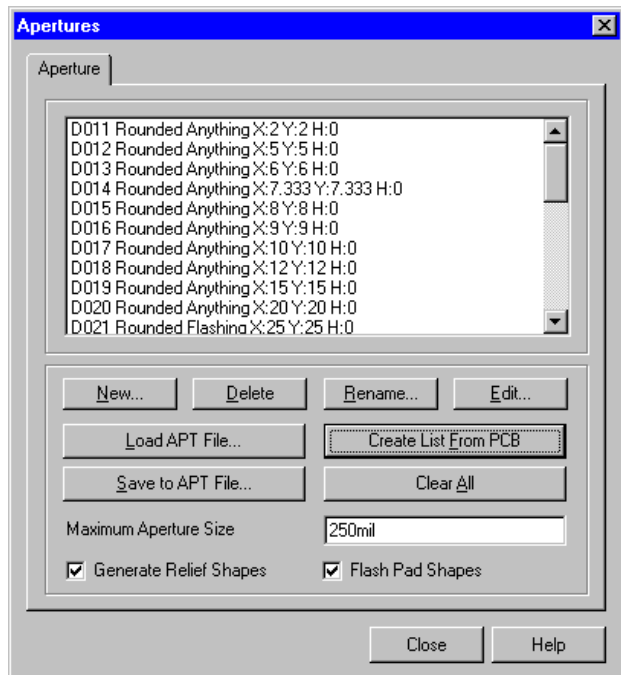
an aperture file, extracted from the primitives (tracks, pads, etc) in the current PCB file. When targeting a vector plotter, the apertures in the .APT file must correspond to the apertures available on the actual aperture wheel or slide to be used. The photoplotting bureau will supply the aperture table to suit their vector plotter. Raster plotters use the aperture file to translate draft codes directly into an image “map”. If the target plotter is a raster device, you can generate the apertures from the PCB and supply the generated aperture table with the Gerber files. Your photoplotting bureau will supply the required file generation details.

When you use an existing aperture file, the PCB Editor scans the primitives (tracks, pads, etc) in the PCB file and matches these with aperture descriptions in the loaded .APT file. If there is no exact match of aperture to primitive, the PCB Editor will automatically “paint” the primitive with a suitable smaller aperture. If there is no aperture suitable to “paint” with, a .MAT match file will be generated listing the missing apertures and Gerber file generation will be aborted.

If targeting a vector plotter, use primitives (track and pad sizes and shapes) for which there is a matching aperture. If the designer is familiar with the aperture set supported by the target photoplotter and tailors the choice of objects placed on a PCB design accordingly, the photoplotter will be able to faithfully reproduce the file in the most efficient manner.

Loading and Editing Apertures

Select the **Design » Aperture Library** menu item to load, create, or edit the apertures used by the photoplot output routines. When you choose this menu item the Apertures dialog will pop up. Any currently loaded apertures will be listed. These apertures would be used if you generated a photoplot at this time. If there are no apertures loaded, the list is empty. Options allow you to work with new or existing aperture files. Changes are applied to the aperture file currently loaded into memory. These changes do not become permanent until you use the Save to APT File button.



All aperture manipulation is done in the Apertures dialog

- ◆ You can define up to a maximum of 1000 different draft codes, in the range D00-D9999, although some of these codes (usually D00-D09) may be “reserved” when targeting some plotters, so use of these codes is not generally recommended.
- ◆ When creating new apertures do not include the “D” in the Draft Code Number dialog.

Postscript Output

High-resolution Postscript “imagesetter” output is now widely available from graphic design and typesetting bureaus. This equipment is capable of producing film positives at resolutions as high as 2540 dpi (dots per inch) and can provide a low-cost alternative to Gerber plots.

However, users should be aware that there are some limitations to using this approach for PCB artwork. The resolution of these systems does not necessarily translate into positional accuracy or linearity, particularly when measured over a large area. There are also film size restrictions. Postscript output files can be generated by the PCB Editor’s Power Print feature, refer to the topic *Printing to a Windows Printing Device* for information on printing from the PCB Editor.

Postscript Printing Tips

Postscript printers and “imagers” generally produce output between 300 and 2540 dpi. Because of the high resolution obtainable from these devices, many users are interested in producing artwork quality Postscript prints as a lower-cost alternative to Gerber plots. However, there are a few limitations which should be considered before printing.

High-resolution laser imagers print directly onto film or sensitized paper. While these devices are quite accurate horizontally, they do not always achieve consistent linearity, particularly on devices where the film or paper moves off a roll, then through the printing mechanism via a series of rollers.

Some typesetting / graphic arts bureaus now use Postscript imagers that use cut, rather than roll film, mounted on a large drum. These imagers suffer much less from linearity problems and may provide a suitable alternative to Gerber plots for non-critical designs.

To test any Postscript device, create a file with vertical and horizontal tracks of known length and carefully measure the output with a rule of known accuracy. This will allow you to apply a correction factor scale setting to either axis, which should minimize the problem. The amount of linearity error may not always be constant, so you should check each final artwork print for accuracy before committing the art to fabrication.

Another problem with 300 or 600 dpi “desktop” laser printers is the “overspray” and “bleed” effects created when the toner is fused to the paper. Small particles adhere to the paper on either side of lines, etc, creating the potential for unwanted effects in your artwork.

When designing for laser print artwork, you should keep the clearances generous, and again, print at a reasonable scale to minimize scale and bleed effects.

The print quality obtainable with a laser printer is largely determined by the paper. A number of special papers are currently available (primarily for the graphics arts trades) which reduce this toner “bleed” into the paper, hence making the outline sharper. Some of these special papers are slightly heavier and treated to resist the waxes and glues used for paste-up, making them easier to handle. Be especially careful to keep these paper laser prints clean.

Postscript compatible photo-typesetting equipment has the advantage of being able to provide output at very high resolutions (up to 2540 dpi). These devices can also print a direct film positive to A3 (or “B”) size.

However, the concern with linearity, described above, applies to these devices as well. The problem of linear accuracy will already be familiar to imagesetting bureaus who provide color separations to the graphics arts industry.

Some Postscript printers will “time out” and discard the current data when they do not receive the end of page marker within a specified time. This can cause problems where you seem to be missing pages from your plots. If you experience this problem using a Postscript printer or any other printing device then you should go to the Control Panel, select the printer icon, select the printer and click the Configure button. Change the Transmission Retry to 500 seconds, or some larger number. This will allow the printer sufficient time to catch up before the Print Manager gives up.

Pen Plotting

You can plot to a pen plotter via a Windows plotter driver. This should not be a problem for newer devices that use raster, rather than vector plot routines, such as the newer large format “ink jet” type plotters. There are also true vector plotter drivers available, contact your plotter supplier for further information.

Passing Design Changes Back to the Schematic

Printed circuit boards are often designed with their designators assigned so that they increment in a logical fashion across the board. This makes it easy to quickly locate a component on the board. The PCB Editor can do this *re-annotation* of designators automatically. Once this is done, you can pass these changes back to the schematic.

Re-assigning the Component Designators

In the PCB Editor, the components on the board can have their designators re-assigned on a positional basis (they can be *re-annotated*). To re-annotate the PCB, select the **Tools » Re-Annotate** menu item. The Positional Re-Annotate dialog will pop up. In this dialog you specify the starting point, and the direction of annotation. The four directional options work as follows:

1. Numbers from bottom to top, starting in the lower left corner.
2. Numbers from top to bottom, starting in the upper left corner.
3. Numbers from left to right, starting in the lower left corner.
4. Numbers from left to right, starting in the upper left corner.

◆ When a board is re-annotated the designators are re-assigned so that there are no gaps in the designation sequence.

The annotation routine uses a “band width” of 100 mils, stepping across in bands as specified by each option. For example, the first option is titled “Ascending X then Ascending Y. This option sets the first band 100 mils wide in the X direction, then scans up the band in the Y direction. It then sets the next band in the X direction and again scans up the band in the Y direction, and so on. The bottom left corner of the component bounding rectangle (the smallest rectangle that encloses all the component primitives) is used as its location reference. Top layer components are annotated first, the bottom layer components.

When you select **Tools » Re-Annotate** from the menus the designator changes are written into a *was/is* file. This is a simple ASCII file which lists what each designator *was*, followed by what it now *is*. Protel 99 SE does not need this file to transfer designator changes back to the schematic, it simply provides a record of the changes that were made.

◆ Select **Design » Update Schematic** from the menus to transfer designator changes back to the schematic.

Updating the Schematic from the PCB

After re-annotation has been performed on the PCB, the designator changes can be transferred back to the schematic. In Protel 99 SE this is easy, simply select **Design » Update Schematic** from the menus. The Synchronizer will open all the schematic sheets,

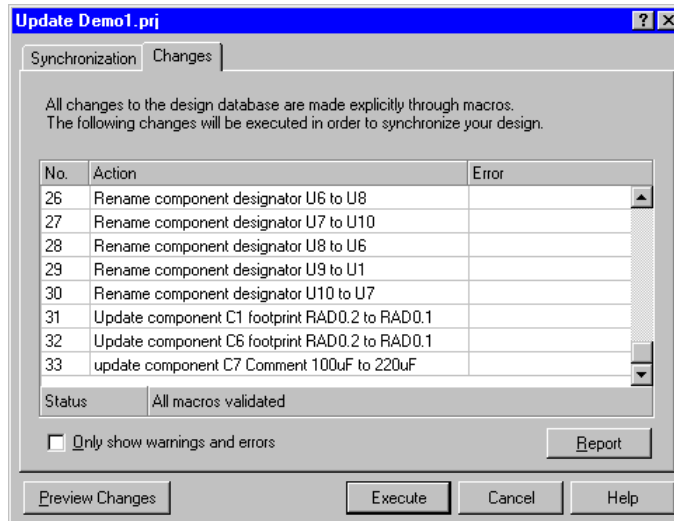
and update the schematic component designators. There are no restrictions on how many times you can re-annotate the PCB before updating the schematic. You can also manually change the PCB designators if you wish, then update the schematic by selecting **Design » Update Schematic**.

As well as transferring designator changes, Protel 99 SE's Synchronizer can also pass other design changes from the PCB, back to the schematic. The following changes are fully supported:

- Component designator changes
- Component comment changes
- Footprint changes
- Component deletions

When you select **Design » Update Schematic** from the menus the Update dialog appears, with the name of the target schematic project in the title. Macros are created for each design change that can be transferred back to the schematic. When you press the Execute button any components on the schematic sheets that have their component attributes changed are automatically updated.

◆ Refer to the chapter, *Transferring the Design Information to the PCB*, in the *Schematic Capture* section of the Handbook, for complete details of the Synchronizer.



The synchronizer passes design changes from the PCB, back to the schematic

◆ The Synchronizer compares the connectivity in the schematic, with the connectivity in the PCB. It does not modifying the schematic wiring, any changes to the net connectivity that you have made in the PCB are logged in a report. Press the Report button at the bottom of the Changes Tab to create the report, then use this as a check list as you update the schematic wiring.

Interfacing to Third-Party Tools

The PCB Editor supports a number of methods of transferring design information into, and out of, the PCB workspace.

Loading a Netlist

◆ If you are doing both the schematic and PCB design in Protel 99 SE, you do not need to use a netlist to transfer the design information. Protel 99 SE includes a powerful design synchronization tool that automates the transfer of design information from schematic-to-PCB, and from PCB-to-schematic. Refer to the chapter, *Transferring the Design Information to the PCB*, in the *Schematic Capture* section, and the chapter *Passing Design Changes Back to the Schematic* in the *PCB Design* section of the Handbook for more details.

Netlists come in many different formats, but are usually generated as ASCII text files which carry at least two types of information:

1. Descriptions of the components in the design.
2. A list of all pin-to-pin connections in the design.

Some netlist formats combine both sets of data in a single description, Others, including Protel, separate the two data into separate sections.

As straightforward text files, netlists are readily translated into other formats using a simple, user-written program. Netlists can also be created (or modified) manually using a simple text editor or word processor.

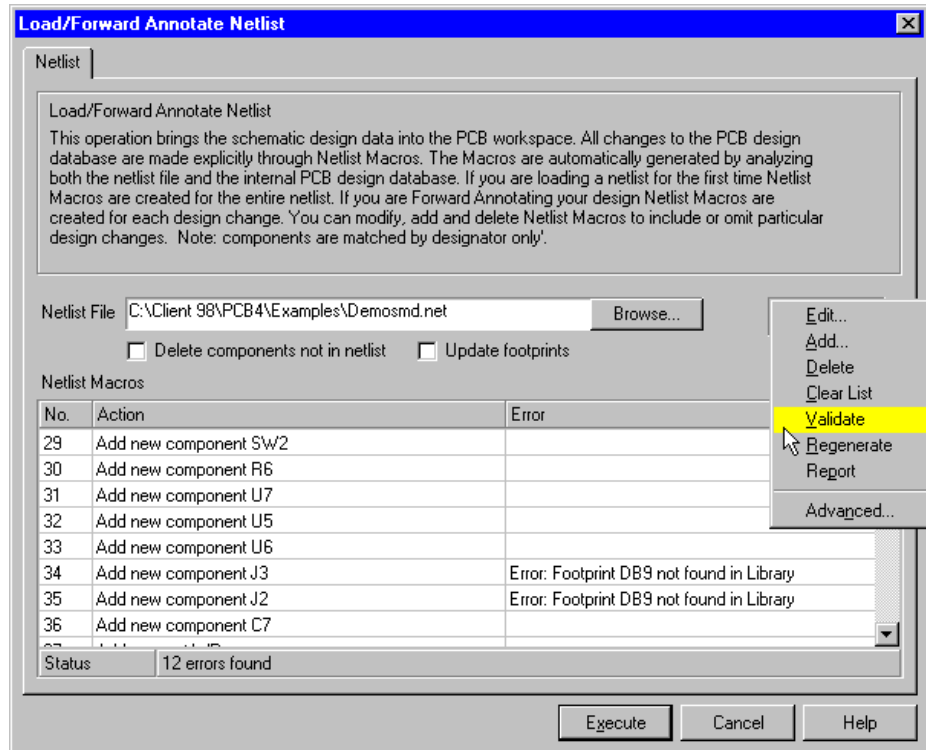
The PCB Editor can load Protel, Protel2 and Tango format netlists.

Locating the Netlist

To load a netlist select **Design » Netlist** from the PCB Editor menus. The Load/Forward Annotate Netlist dialog will pop up. Press the Browse button, locate and select the netlist, and click OK to return to the Load/Forward Annotate Netlist dialog.

The PCB Editor will then analyze both the netlist, and any PCB design data present in the workspace. For each difference detected between the netlist and the existing design data, a *Netlist Macro* is created. This Macro tells the PCB Editor what action must be performed to update the design data to match the netlist.

◆ Ensure the required component libraries have been added to the current library list prior to loading the netlist (**Design » Add/Remove Library**).



If you are loading a netlist for the first time, Netlist Macros will be created for the entire netlist. If you are forward annotating your design, Netlist Macros are created for each design change.

Examine the Macros, and when you are satisfied that they will carry out the actions you require, press the Execute button.

- ◆ It is good practice to resolve any Macro errors prior to Executing the Macros. Refer to the topic *Resolving Netlist Macro Errors* later in this chapter for tips.

Working with Netlist Macros

After the PCB Editor has analyzed the netlist and any PCB design data present in the workspace, it will create the Netlist Macros and list them in the Load/Forward Annotate Netlist dialog, in the order that they will be executed. The Macro Commands include:

- Remove Node
- Remove Net
- Remove Component

- Add Component
- Add Net
- Change Net Name
- Change Component Footprint
- Change Component Designator
- Add Node
- Change Component Comment

As well as allowing you to examine exactly what will happen when you load a netlist, you can also Add, Edit and Delete netlist macros.

Adding, Editing and Deleting Netlist Macros

Use these options when you wish to omit or include certain design changes. You can also use them when you wish to modify the PCB netlist, without returning to the schematic. Points to be aware of when editing Netlist Macros include:

- Be consistent with the case of all text, such as Designators and Nets.
- A netlist node is specified as the *ComponentDesignator-PinNumber*. An example is J3-2. As per the PCB Library Editor requirements, the pin number can have a maximum of four alpha/numeric characters with no spaces.
- The footprint must match the name of a footprint in one of the libraries in the current library list.
- Refresh the screen after executing the Netlist Macros.

Validating Netlist Macros

After editing Macros you should always Validate them. This is particularly important if you have manually created Macros. Pressing this button instructs the PCB Editor to examine each Macro, check if it can be executed, and report any error condition.

Regenerating Netlist Macros

When you press Regenerate the PCB Editor will clear all existing Macros, then re-analyze the netlist and the PCB design data and create new Macros.

Resolving Netlist Macro Errors

Prior to executing the Netlist Macros it is good practice to resolve any errors or warnings. Following is a description of each error/warning. The description includes which Macro can report that error and what has caused the error.

Net not found

A Netlist Macro is attempting to: add or remove a node; remove a net; or change a net name when that net can not be found in the PCB netlist.

Component not found

A Netlist Macro is attempting to: add or remove a node when the component designator is incorrectly specified in the Macro or the component can not be found in

the PCB netlist; remove a component; or change a footprint, designator or comment when the component can not be found in the PCB netlist.

Node not found

A Netlist Macro is attempting to: add or remove a node from a component which does not have that pin; or remove a node which does not exist in the specified net.

Net already exists

A Netlist Macro is attempting to: add a net name when a net with that name already exists in the PCB netlist.

Component already exists

A Netlist Macro is attempting to: add a component when a component with that designator already exists in the PCB netlist.

New footprint not matching old footprint

A Netlist Macro is attempting to: change a component footprint when the *used* pins on the old footprint do not match the *used* pins on the new footprint. This can occur if the new component has fewer pins than the old, or if the pin numbering in the netlist (which comes from the schematic component pin numbers) is different to the pin numbering on the PCB component.

Footprint not found in Library

A Netlist Macro is attempting to: add a new component or change a component footprint when the specified footprint could not be found in any of the libraries in the current library list and no alternative library reference could be found in the Cross Reference file (ADVPCB.XRF).

Alternative footprint used instead (warning)

A Netlist Macro is attempting to: add a new component or change a component footprint when the specified footprint could not be found in any of the libraries in the current library list. An alternative library reference was found in the Cross Reference file (ADVPCB.XRF) and this component will be loaded from one of the libraries in the current library list. Always confirm that the alternative footprint is appropriate before executing a Macro with this warning.

- ◆ When a Netlist Macro attempts to load or change a component footprint which can not be located in any of the libraries, it then uses the component Comment to look-up the Cross Reference file (ADVPCB.XRF). The Cross Reference file lists components by their type against any appropriate footprint(s) for that component. For example, if the component U1 was a 74LS00, but you had forgotten to include the footprint, when the Macro to add this component was validated it would look-up 74LS00 in the XRF file. 74LS00 has DIP14 as a footprint, which would be loaded from one of the libraries in the current library list.

Summary

Most problems with loading a schematic netlist generally fall into two categories.

1. Component footprints – missing components occur when: a footprint is missing from the component information in the netlist; you have forgotten to add the required PCB libraries to the current library list (**Design » Add/Remove Library**); or the footprint in the netlist is not available in any of the libraries in the library list.
2. New footprint not matching old footprint – the cause is usually that the pin numbering on the schematic component differs from the pin numbering on the PCB footprint.

Schematic libraries contain specific components and devices. The PCB libraries contain generic footprints, which can belong to various specific components – each having different pin assignments.

For example, a transistor shape can represent various combinations of “E,” “B” and “C,” – each of which must be assigned to the correct pin number in the PCB Editor. Diodes are a similar case, with pins often named “A” and “K” in the schematic.

You will need to either, modify the PCB footprint pin numbers to match the Schematic pin numbers, or change the schematic component pin numbers to match the PCB footprint.

Supported Netlist Formats

Protel 99 SE's PCB Editor can load Protel, and Protel2 format netlists. Refer to the chapter, *Interfacing to Third-Party Tools*, in the *Schematic Capture* section of this Handbook for details of the syntax of these netlist formats.

Netlists from schematic capture packages other than Protel usually have many similarities to the Protel format. However, the order in which component or net information is displayed may vary, and package names (eg DIP16), component designators and pin identifiers may require editing to match the PCB Editor field restrictions. Often, translation of the netlist is an option in the schematic package. Netlists created using either a Protel or “Tango” output option will usually be fully compatible with Protel 99 SE's PCB Editor.

Netlist Parameter Restrictions

Designators and Package Descriptions (Type) are limited to 12 alphanumeric characters. Part Types can be up to 32 characters long. Net names can be 20 characters. Pin numbers in netlists are limited to 4 alphanumeric characters. No blank spaces may be used within these strings.

Any number of components or nodes can be included in a netlist, limited only by available memory.

Exporting the Netlist

There are two ways of exporting netlist data from the current PCB workspace. You can either export the internal netlist, or generate one from the actual routing. Both can be done in the Netlist Manager (select **Design » Netlist Manager**).

Exporting the Internal Netlist

This exports the internal design netlist. The internal netlist is the connectivity information that was transferred from the schematic. If the design is not routed this connectivity is shown by the connection lines. To export the netlist click on the Menu button in the Netlist Manager dialog and select **Export Netlist from PCB** from the menu. The netlist is created inside the Design Database.

Generating a Netlist from the Routing

This generates a component and connectivity list (netlist) from the components, and the routing (tracks and vias) on the actual board. To generate a netlist click on the Menu button in the Netlist Manager dialog and select **Create Netlist from Connected Copper** from the menu. The netlist is created inside the Design Database.

File Importers and Exporters

The PCB Editor can import from and export to a variety of other file formats. Exporting options include older version Protel file formats and AutoCAD® DWG/DXF. Importing options include older version Protel file formats, Gerber files and Orcad Layout® PCB files.

To import a file select **File » Import** from the PCB Editor menus. Select the required format in the drop down file type list at the bottom of the Import File dialog.

Working with Older Protel Formats

The PCB file format has changed in Protel 99 SE, the file format is now referred to as PCB 4.0. To save a PCB back to the previous format (PCB 3.0) which was used by Advanced PCB V3, Protel 98, Protel 99 and Protel99 SP1, select **File » Save Copy As** from the PCB Editor menus.

Protel 99 SE can import all older version Protel PCB files. Import the file into the design database in the normal way (shortcut: drag and drop the file from the Windows Explorer into the Design Explorer), then double-click on it to open it.

Mechanical CAD Interface

Protel 99 SE includes full support for importing from and exporting to both DWG and DXF format files. Select **File » Import** or **File » Export** from the PCB Editor menus and set the file type to DXF/DWG at the bottom of the dialog.

Summarizing the Mechanical CAD Interface features:

- Full support for DWG/DXF import/export to the PCB editor, all versions from 2.5 to R14.
- User-definable layer mapping on import. The Import from AutoCAD dialog will appear during the import process, where the layer mapping is defined.
- Import from AutoCAD model space or paper space.
- Automatic import scaling if the import data is larger than the PCB workspace.
- Component to block and block to component translation.
- Supports metric or imperial units.

Use the Help button and What's This help in the dialogs for more details on the import/export options.

Orcad Layout® to Protel PCB Interface

Protel 99 SE can import binary format Orcad Layout V9.x design files. Select **File » Import** from the PCB Editor menus and set the file type to MAX at the bottom of the dialog.

Summarizing the Layout Importer features:

- Directly load Orcad Layout (V9.x) PCBs into the PCB editor.
- User-definable layer mapping in the Orcad Layout Importer dialog. This dialog appears during the import process.
- Comprehensive, multi-level import reporting.
- Import Layout (*.LLB) libraries into Protel 99 SE's PCB Library Editor.
- Import a library directly from a Layout PCB file.

Use the Help button and What's This help in the dialogs for more details on the import options.

Index

.	
.IC.....	210
.NS	210
3	
3D PCB view	618
A	
Absolute ABS	358
absolute PCB origin	446
ABSTOL	257
AC analysis	
setting up and running	188
access codes	7
ACCT	259
acute angle constraint.....	544
ADCSTEP.....	260
adding PCB layers.....	522
adding project sheets.....	124
address range	
PLD design.....	343
align	
PCB components.....	569
schematic objects	91
analog simulation	
definition	178
Analog simulation	
running	182
analyses	
setting up the circuit simulator	183
analysis window	
using.....	211
analyzing nets	
switching off.....	581
angled routing	603
annotation	
backward	642
forward	146
reassigning designators in PCB.....	642
reassigning designators in schematic.....	142
ANSI	
sheet border	66
title block.....	67
apertures	
about.....	638
and Gerber files	637
loading and editing	639
arc	
changing in PCB.....	501
PCB	500
placing in PCB	500
array	
placing in PCB	476
placing on a sheet	84
artwork	
about PCB	626
which kind?	636
ASCII	
HL	357
asynchronous	
PLD simulation	369
auto wire placement mode.....	76
auto-junction.....	77
enabling / disabling	64
autopan	
PCB	463
setting up in PCB.....	454
AUTOPARTIAL	259
autoplacement	
in PCB	571
setting up the cluster placer	572
setting up the global placer.....	572
autoroute	
area	603
component	602
net.....	602
options.....	602
setting up	599
avoid obstacle	

Index

setting mode 455
routing 592

B

back-annotation 642
BADMOS3 258
Berkeley
 PLA format 358
Bill of Materials
 schematic 158
bit fields
 PLD design 343
BJT
 simulation 239
blind/buried vias
 using 498
board
 adding layers 522
 creating with the Wizard 524
 defining 521
 drill pairs 524
 layer properties 524
 layer stack 522
 mechanical definition 521
 placement and routing outline 522
 signal/insulation layer order 523
BOM
 setting up in PCB 633
boolean
 excess product terms 356
Boolean expressions 258
BOOLH 258
BOOLL 258
BOOLT 258
borders
 sheet 66
breaking a wire 80
broken net, finding 612
browsing
 PCB workspace 464
 schematic 73
bureau, working with 626
bus
 about 100

connectivity 60
creating connectivity 100
bus entry
 about 100
BYPASS 259

C

CAM Manager
 BOM setup 633
 DRC setup 635
 Gerber setup 629
 NC drill setup 632
 pick and place setup 633
 testpoint report setup 634
 using 626
capacitor
 simulation 237
change
 PCB object 466
 schematic object 78
changing the layer of a PCB component 564
checking the schematic 136
CHGTOL 257
child sheet 130
circuit
 will not simulate 252
circuit design
 about 57
circuit simulator
 features 177
class
 creating from a selection 473
classes
 creating from the schematic 566
 creating PCB classes 537
clearance design rule 539
clipboard
 adding template to 65
 PCB 474
 working with in schematic 83
comment
 PCB component 514
compatibility
 working with an older Protel design 29

compile		
PLD design.....	351	
compiler options		
PLD design.....	354	
compiling SimCode.....	265	
complex hierarchy.....	131	
complex to simple.....	131	
component		
creating for simulation	247	
finding for simulation.....	236	
component		
aligning PCB components.....	569	
changing pads in.....	497	
changing PCB footprint.....	515	
color, changing.....	113	
copying.....	120	
copying between libraries.....	520	
copying from the PCB to a library	516	
creating a PCB footprint.....	519	
creating a schematic component.....	117	
default footprint.....	119	
finding in a PCB library	511	
finding in schematic libraries	109	
flipping and rotating.....	564	
grouping	119	
including routing in the footprint	516	
jump to in PCB.....	464	
listing on board.....	617	
locating.....	491	
modifying on the PCB.....	515	
orientation	113	
part type field	112	
PCB attributes	512	
PCB comment	514	
PCB component union	567	
PCB designator.....	514	
PCB placement.....	563	
PCB project library	520	
positioning on the board.....	564	
return to PCB primitives	516	
schematic attributes.....	111	
selecting a PCB component from the schematic.....	566	
sharing a common graphic	119	
sheet path.....	112	
sheet path field	132	
simulation-ready.....	235	
special PCB strings.....	503	
specifying footprint in schematic	111	
text fields.....	113	
updating the PCB footprint.....	520	
component clearance constraint	550	
component grid, PCB.....	447	
component orientation rule.....	550	
component placement toolbar	569	
component text		
about.....	111	
components		
about.....	58	
accessing in PCB.....	509	
accessing in schematic.....	107	
associating schematic to PCB.....	150	
moving onto grid	586	
PCB	509	
schematic.....	106	
where are the PCB components.....	509	
where are the schematic components	107	
conditional PLD simulation.....	377	
confirm global edit	454	
connection lines		
displaying	452	
hiding.....	581	
connections		
inter-sheet.....	125	
wiring	59	
connectivity		
about.....	59	
between sheets.....	125	
displaying	577	
logical.....	60	
physical	60	
rules for	60	
understanding in PCB.....	576	
CONVABSSTEP	259	
conventions		
document.....	8	
convergence		
troubleshooting.....	253	

Index

- convert special strings 456
- converting
 - an older Protel design 29
- CONVLIMIT 259
- CONVSTEP 259
- coordinate
 - PCB 508
 - PCB coordinate system 446
- copper clearance design rule 539
- copper plane
 - creating 598
- copper trace layers 450
- copy
 - clipboard reference, enabling 65
 - component between libraries 120
 - from PCB to the Windows clipboard 624
 - morphing a schematic object 88
- corners, track placement mode 494
- coupled inductors, simulating 209
- creating
 - schematic component 117
- creating digital simulation models 263
- cross probe
 - from the schematic to the PCB 567
- crystal
 - simulation 243
- CUPL
 - language syntax 344
 - syntax 336
- current controlled current source 230
- current controlled voltage source 230
- current-controlled switch 241
- CURRENTMNS 260
- CURRENTMXS 260
- cursor
 - style 454
- customizing
 - the Design Explorer 36
- D**
- daisy chain stub length design rule 548
- data points
 - displaying 217
- database
 - interfacing to from the schematic 159
- Database, Design 17
- DC analysis
 - setting up and running 191
 - troubleshooting 254
- deactivate unused OR terms 355
- dead copper 505
- debug
 - SimCode 271
- decals
 - component default 119
 - specifying in schematic 111
- DEFAD 257
- DEFAS 257
- default
 - designator 119
 - sheet template 64
- default primitives
 - PCB 459
 - schematic 65
- default resources
 - about 37
- defaults
 - resetting PCB origin 446
 - resetting server resources 38
- DEFL 257
- DEFW 257
- delete
 - PCB Editor 485
- DeMorgan 354
- deselect
 - before global editing 88
 - PCB objects 469
 - schematic objects 81
- design
 - Design Database, what is it? 17
 - editing 21
 - new design 18
 - new design document 20
 - structuring the project 122
- Design
 - Importing a document 22, 23
 - linking a document 22
 - sharing a Design Database 25

working with an older Protel design.....	29	power plane clearance	545
design example		power plane connect style	546
4-Bit Counter.....	433	precedence of the scopes	533
decade counter.....	422	resolving contentions.....	533
seven-segment decoder	430	room definition.....	551
subway turnstile controller.....	420	Routing.....	539
two-bit counter	418	routing corners.....	539
virtual device	434	routing layers.....	539
Design Explorer		routing priority	540
about.....	33	routing topology	540
customizing	36	routing via style.....	542
getting started.....	11	routing width	543
design rules		scope definitions.....	529
about.....	526	setting the scope	528
acute angle constraint	544	short circuit constraint.....	556
adding.....	527	signal base value.....	553
checking which apply.....	536	signal flight time.....	553
component clearance constraint	550	signal integrity.....	552
component orientation.....	550	signal stimulus.....	554
compound scope.....	532	signal top value.....	554
copper clearance.....	539	slope	554
daisy chain stub length	548	SMD neck down.....	542
disabling	535	SMD to corner.....	542
duplicate.....	533	SMD to plane	542
examples of using.....	557	solder mask expansion.....	546
high speed	548	testpoint style.....	546
hole size constraint.....	544	testpoint usage	547
how they are applied	534	unary/binary	531
impedance constraint.....	552	un-connected pin constraint.....	556
import/export.....	535	undershoot	555
layer pairs	544	un-routed nets constraint	556
manufacturing	544	vias under SMT constraint	549
matched net lengths.....	548	design verification	
maximum via count.....	549	PCB	609
minimum annular ring.....	544	schematic.....	136
monitoring.....	535	designator	
net length constraint	548	default.....	119
nets to ignore.....	550	pad.....	496
overshoot.....	552	PCB	514
parallel segment constraint.....	549	schematic, about.....	111
paste mask expansion.....	545	designators	
permitted placement layers.....	551	reassigning in PCB	642
placement	550	reassigning in the schematic.....	142
polygon connect style.....	545	device	

Index

- finding in PLD.....352
- mnemonic.....352
- name in PLD.....352
- secure.....355
- secure PLD device.....355
- selection for PLD design.....351
- virtual device.....353
- diagonal routing.....603
- Digital
 - SimCode simulation language.....262
 - simulation.....262
- digital components
 - simulation.....246
- digital simulation
 - supply and ground.....257
- dimension
 - changing in PCB.....508
 - PCB.....507
 - placing in PCB.....508
- diode
 - simulation.....238
- directives
 - about.....59
- display
 - connection lines.....452
 - draft thresholds.....457
 - DRC errors.....452
 - mode.....458
 - origin marker.....457
 - pad & via holes.....452
 - pad numbers.....457
 - PCB layers.....449
 - visible grids.....452
- displaying waveforms.....211
- DM-PL
 - plots, about.....626
- document
 - copying.....24
 - deleting.....24
 - exporting.....23
 - importing.....22
 - linking to.....22
 - locking.....28
 - moving.....24
- download file
 - PLD design.....357
- draft code
 - about.....637
- draft thresholds.....457
- drag
 - orthogonal in schematic.....64
 - PCB object.....483
 - PCB shortcuts.....483
 - schematic object.....93
 - tracks with component.....455
- draw order of layers.....457
- DRC
 - about.....609
 - enabling online.....453
 - errors, display of.....452
 - jump to error marker.....465
 - online.....609
 - report.....611
 - resolving violations.....611, 612
 - setting up for.....610
 - setting up the DRC in the CAM Manager..635
- drill drawing
 - layer.....452
- drill guide
 - layer.....452
- drill pairs, PCB.....524
- drive capacity
 - simulation.....260
 - simulation override.....260
- DRIVEMNS.....260
- DRIVEMXS.....260
- drivers
 - about, plotter.....641
- DRVMNTYMX.....260
- DXF/DWG
 - import/export to PCB.....649
 - schematic import/export.....169
- E**
 - EDIF.....358
 - editing
 - a document.....21
 - PCB net attributes.....582

- PCB object 466
- PCB object graphically 468
- PCB shortcuts 487
- schematic object 78
- schematic object graphically 79
- Schematic shortcuts 95
- while placing in PCB 466
- while placing in schematic 78
- electrical grid
 - PCB 447
 - Schematic Editor 67
 - using 77
- electrical objects 59
- wiring 59
- Electrical Rule Check
 - setting up 136
- equation-defined source 233
- ERC
 - setting up 136
- error
 - circuit will not simulate 252
 - simulation netlist 252
- error list 358
- error messages 255
- errors
 - synchronizing, resolving 152
- examples
 - PLD design 411
 - using design rules 557
- excess product terms
 - eliminating 356
- expanded macro MX 358
- exponential simulation source 226
- exporting
 - a document from a Design Database 23
- extend selection 453
- F**
- fabrication & assembly layers 451
- Failures
 - Transient Analysis 255
- fault simulation
 - PLD design 385
- fiducials, mask expansion 558
- field
 - PLD keyword 343
- file
 - README 7
- fill
 - about 499
 - changing 499
 - placing 499
- find and replace
 - schematic text 89
- fit on page option, schematic 156
- flat design 128
- flipping a PCB component 564
- focus
 - PCB Editor 466
 - Schematic Editor 79
- folders
 - using 14
- font
 - changing system font 68
 - component text 111
- fonts
 - rotating 155
 - schematic system font 105
 - using in Schematic Editor 105
- footprint
 - component default 119
 - creating 519
 - creating with the PCB Wizard 519
 - specifying in schematic 111
 - updating 520
- format, Gerber 637
- forward-annotation 146
- Fourier analysis
 - setting up and running 202
- frequency and phase, viewing 216
- frequency controlled voltage source 231
- frequency modulation simulation source 228
- From-To
 - about 578
 - creating 579
 - using in a design rule 560
- fuse
 - simulation 243

Index

- statement, PLD design..... 341
- G**
- generating output
 - about..... 626
- Gerber
 - about..... 637
 - apertures..... 638
 - plots..... 626
 - setting up..... 629
- global editing
 - copy PCB attributes..... 479
 - copy schematic attributes..... 87
 - partial string substitutions..... 88
 - PCB attributes to match by..... 479
 - PCB change scope..... 480
 - PCB Editor..... 478
 - PCB examples..... 480
 - schematic attributes to match by..... 86
 - schematic change scope..... 87
 - Schematic Editor..... 85
 - using the {}..... 88
 - using wildcards in schematic..... 87
- global net identifiers..... 59
- GMIN..... 257
- GMINSTEP..... 259
- graphic
 - adding to schematic..... 103
- graphical editing
 - PCB object..... 468
 - schematic object..... 79
- grid
 - about PCB..... 447
 - electrical, Schematic Editor..... 67
 - electrical, using..... 77, 583
 - PCB component grid..... 447
 - PCB electrical grid..... 447
 - PCB snap grid..... 447
 - PCB visible grids..... 448
 - polygon..... 506
 - snap, Schematic Editor..... 67
 - using in PCB..... 586
 - visible, Schematic Editor..... 67
- group
 - component groups..... 119
 - group object
 - PCB..... 504
- H**
- hand
 - PCB slider hand..... 488
- handle
 - in focused PCB object..... 467
 - in focused schematic object..... 79
- Hex download format..... 357
- hidden pin
 - creating connectivity..... 126
- hierarchy
 - about..... 123
 - complex..... 131
 - preferred method..... 130
 - simple..... 130
 - tools for creating..... 135
 - working in a hierarchical project..... 134
- highlight in full..... 456
- HL download file..... 357
- hole size constraint..... 544
- how to
 - global editing in PCB..... 480
- How to
 - break a wire..... 80
 - create a custom template..... 69
 - find and replace schematic text..... 89
 - place a wire..... 75
 - remove a wire corner..... 80
 - select an individual schematic object..... 82
 - select schematic by area..... 82
- HP-GL
 - plots, about..... 626
- I**
- identify a net..... 582
- If ... Then
 - SimCode..... 281
- ignore obstacle
 - setting mode..... 455
- routing..... 592
- IMNTYMX..... 260

impedance constraint, signal integrity.....	552	absolute PCB origin.....	464
impedance plot analysis		current PCB origin.....	464
setting up and running.....	205	new PCB location.....	464
imperial units		PCB.....	464
changing PCB to.....	446	PCB component.....	464
importing		PCB component pin.....	465
into a Design Database.....	22	PCB error marker.....	465
inductor		PCB net.....	465
simulation.....	238	PCB selection.....	465
inductor		schematic.....	74
simulating coupled inductors.....	209	string in PCB.....	465
initial conditions		junction	
enabling.....	187	about.....	100
specifying.....	210	K	
Initial Conditions device.....	210	Keep XOR.....	354
input loading		KEEPOPINFO.....	258
simulation.....	260	keyboard	
simulation override.....	260	PCB shortcuts.....	489
installing Protel 99 SE		schematic shortcuts.....	96
entering access codes.....	7	L	
Installing Protel 99 SE		labels, net.....	59
how to.....	7	language	
integration method		CUPL language elements.....	342
setting.....	256	language syntax	
when to change.....	255	CUPL.....	344
intermediate variable		SimCode.....	272
PLD design.....	338	language, photoplotter.....	637
inter-sheet connections.....	125	languages	
ISO		editing.....	48
sheet border.....	66	layer pairs design rule.....	544
iterations		layer stack-up.....	457
increasing.....	254	layers	
ITL1.....	257	about PCB layers.....	449
ITL2.....	257	current PCB layer.....	450
ITL3.....	257	displaying PCB layers.....	449
ITL4.....	258	draw order.....	457
ITL5.....	258	drill.....	452
J		internal planes.....	451
Jedec		keep out.....	452
producing.....	357	mechanical.....	451
JFET		moving between.....	450
simulation.....	239	multi.....	452
jump			

Index

paste mask	451	simulation override.....	260
PCB active.....	449	LOADMNS	260
power planes.....	451	LOADMXS	260
redrawing automatically in PCB	456	location	
signal	450	jump to on sheet	74
silkscreen overlay	451	marking on the sheet.....	74
single layer mode	457	locking	
solder mask.....	451	locking a design document.....	28
transparent.....	457	logic equation	
layout directives	63	PLD design.....	338
LDMNTYMX	260	logic reduction	
libraries		PLD compiler	356
adding to the PCB library list.....	509	logical connectivity	
adding to the schematic library list.....	107	about.....	60
loading in schematic.....	108	loop removal	
PCB	509	enabling.....	455
schematic.....	106	lossless transmission line.....	245
where are PCB libraries	509	lossy transmission line.....	245
where are schematic libraries	107		
working with older format libraries.....	30	M	
library		macros	50
about components.....	58	mask	
component group.....	119	solder and paste	451
component text fields	119	master sheet	
creating a PCB project library.....	520	about.....	123, 124
creating new schematic library.....	115	matched net lengths design rule	548
editing PCB libraries	518	mathematical analysis and plots	206
editing schematic libraries.....	115	MAXEVTITER.....	259
sheet path field	132	maximize product term sharing.....	356
Library Editor		maximum via count design rule	549
PCB	517	MAXOPALTER.....	259
schematic.....	114	measurement cursors	
library reference field, schematic	111	using.....	218
library text fields	113	measuring distance in PCB.....	486
line		mechanical layer	
about.....	493	about.....	451
linear dependent simulation source	230	menus	
linking		about.....	36
a document to a Design Database.....	22	MESFET	
schematic component to a sub-sheet	112	simulation	241
LIST	259	messages	
loading		errors and warnings	255
netlist.....	644	metric units	
simulation.....	260	changing PCB to.....	446

mils	
toggling PCB units	446
MINBREAK	259
minimization	
individual	355
memory vs speed vs efficiency.....	356
none	356
minimization methods	
PLD compiler	356
minimum annular ring design rule	544
MiniViewer, using.....	461
mm	
toggling PCB units	446
mnemonic	
device	352
mode	
track placement	494
model	
simulation.....	235
modular design	131
Monte Carlo analysis	
setting up and running	194
morphing	88
MOSFET	
simulation.....	240
move	
break track.....	484
PCB component reference point.....	453
PCB object	483
PCB primitive.....	484
polygon vertices	485
schematic object.....	92
moving	
object reference point	65
multipliers	
problems with.....	253
simulation.....	181
multi-sheet design	
managing	123
methods of structuring.....	127
multi-sheet projects	
about.....	122

N

Navigation Panel	
using	13
NC drill, setting up	632
net	
identifying	582
jump to in PCB	465
listing loaded	617
matched lengths design rule	548
power plane assignment	594
routed length constraint.....	548
topology, about.....	578
net identifier	
scope.....	126
Sheet Symbols/Port Connections	130
net identifier scope	
Net Labels and Ports Global.....	129
Ports Only Global.....	128
Net Identifier Scope	
setting for netlist creation	170
net identifiers	
about.....	59
net label	
about.....	100
connectivity	61
creating connectivity	125
negating.....	126
Net Labels and Ports Global.....	129
net topology	
about.....	578
specifying	578
netlist	
about.....	169
changing and updating.....	581
character limits	648
creating	169
exporting.....	649
formats.....	170
generating	170, 649
loading	644
macros	645
Protel 2 format.....	172
Protel format.....	172

Index

Nodeset device	210
noise analysis	
setting up and running	204
non-linear dependent simulation source	233
NOOPALTER	259
NOOPITER	258

O

object	
editing in PCB	466
editing in schematic	78
PCB design object	492
placing in schematic	75
reference point when moving	65
schematic design objects	99
objects	
adding to PCB selection	469
adding to schematic selection	81
selecting in PCB	469
selecting on the schematic	81
OLE Server	
about	34
One-hot-bit state machines	355
online DRC, enabling	453
operating point analysis	
setting up and running	193
Optimize product term usage	354
options	
printer	156
schematic workspace	64
OPTS	259
Orcad	
Capture, importing	167
Layout, importing	650
organization	
adding company details to the sheet	67
origin	
autoplacement	571
jump to in PCB	464
marker	457
PCB absolute	446
relative	446
orthogonal wire placement	76
output	

generating	626
output drive capacity	
simulation	260
Output drive capacity	
simulation override	260
output options	
PLD design	357
output options	
Absolute ABS	358
ASCII HL	357
Berkely PLA	358
EDIF, PLD compiler	358
error list LST	358
expanded macro MX	358
Hex	357
Jedec	357
Palasm PDS	357
PDIF PDF	358
XNF	358
overlay	
layers	451
overshoot	
specifying for signal integrity	552

P

pad	
about	496
changing	497
designator, incrementing	496
display of holes	452
displaying pad info	457
jump to	465
placing	496
rotated, limitations	484
pad numbers	
showing	457
Palasm PDS	357
pan	
PCB slider hand	488
setting up autopanning	454
panel	
creating a PCB panel	475
Panel	
MiniViewer	461

panning		
PCB	463	
schematic	72	
parallel segment constraint	549	
parameter sweep analysis		
setting up and running	198	
parent sheet	130	
part		
default footprint	119	
footprint	111	
in schematic component	112	
orientation	113	
placing on sheet	110	
schematic attributes	111	
sheet path	112	
text fields	113	
part type		
schematic	112	
partial derivatives, simulation	259	
parts		
about component	58	
parts list		
setting up in PCB	633	
paste		
creating a PCB panel	475	
PCB array	476	
schematic array	84	
special in PCB	475	
paste mask		
expansion design rule	545	
layer, about	451	
pattern		
component default	119	
specifying in schematic	111	
PCB		
3D view	618	
adding layers	522	
back-annotation	642	
CAM Manager	626	
components and libraries	509	
copying from PCB to the Windows clipboard		
.....	624	
defining	521	
drill pairs	524	
information on the Schematic	63	
layer properties	524	
layer stack	522	
layout directives, placing	145	
layout directives, transferring	150	
loading older formats	460	
mechanical definition	521	
new	460	
placement and routing outline	522	
printing	620	
reassigning designators	642	
saving to older formats	460	
signal/insulation layer order	523	
updating from the schematic	146	
PCB Editor		
features	439	
working in	461	
PCB layout directive		
about	101	
PDIF PDF	358	
pen plotting, issues	641	
periodic pulse simulation source	222	
permissions		
defining	26	
photoplotting		
about	636	
vector vs raster	637	
physical connectivity		
about	60	
pick and place report	633	
piece-wise-linear simulation source	224	
pin		
about	102	
connectivity	61	
listing PCB component pins	617	
moving schematic pin text	64	
placing on schematic component	118	
PLD pin definitions	342	
showing hidden on sheet	112	
pinnode		
PLD design	337	
pins		
interfacing to in PLD design	334	
PIVREL	258	

Index

PIVTOL	258
placement	
component placement toolbar	569
manual PCB	563
PCB	563
PCB options which affect	563
working between the schematic and PCB ..	566
placement rooms	
working with rooms	568
placing a pad	496
plane	
defining a copper plane	598
PLD compiler	
source file syntax	336
PLD design	
CUPL HDL	336
interfacing to pins	334
multi-sheet schematic	334
schematic libraries	319
schematic symbols	321
state machine	347
PLD design	
compiler options	354
compiler overview	317
compiling	351
CUPL language syntax	344
device selection	351
download file	357
examples	411
features	313
finding the device	352
how to setup and compile	318
output options	357
overview	316
sample design session	389
schematic-based	319
setting up to compile	351
simulating	359
plotting	
about	626
polygon	
changing the shape of	507
connect style design rule	545
connecting to a net	505
connecting to pads	507
defining a copper plane	598
moving vertices	485
PCB	504
placing	504
plowing through	455
repouring	507
polygons	
plowing through	592
port	59
about	101
connectivity	61
creating connectivity	125
cross referencing	128
Ports Only Global	128
Postscript printing	
PCB	636, 640
tips	640
power plane	
clearance design rule	545
connect style design rule	546
connecting a net to	594
connecting vias to	595
defining a split plane	595
listing pins assigned to	617
using	594
viewing	595
power port	
about	100
creating connectivity	126
power print	620
preferences	
layers tab	449
PCB default primitives	459
PCB layers tab	449
setting	453
show/hide tab	458
pre-routes	
protecting	599
previewing prints/plots	156
primitive	
arc, PCB	500
fill	499
line	493

moving in PCB	484
pad	496
PCB	492
PCB string	501
schematic	99
track	493
via	497
primitives	
setting PCB defaults	459
setting schematic defaults	65
printing	
about	626
changing the target PCB printer	623
PCB print/preview	620
schematic	154
setting the PCB printer options	623
setting up a PCB printout	622
sheet borders	66
process	
about	43
launching	43
parameters	44
process container	
about	104
product terms	
number of	358
unlimited	353
Project Hierarchy report	158
project management	
about	122
PROM	
logic minimization	356
propagation delay	
simulation	259
simulation override	260
PROPMNS	259
PROPMXS	259
protecting pre-routes	599
push and shove	
enabling	455
routing	592
push obstacle	
setting mode	455
routing	592

Q

quick copy schematic object	88

R

RAMPTIME	259
random value	
PLD simulation	367
raster photoplotters	637
ratsnest	
display of	452
README file	7
read-only fields, simulation	247
Redo	
PCB	491
schematic	98
redraw	
canceling in PCB	487
canceling in schematic	95
PCB layers	456
re-entrant editing	
PCB	487
schematic	95
registration, software	7
relative PCB origin	446
relay	
simulation	244
RELTOL	258
remove duplicates	454
removing a corner in a wire	80
removing project sheets	124
report	
board information	617
measure distance in PCB	486
netlist status	617
project hierarchy	158
schematic Bill Of Materials	158
schematic component	121
schematic component rule check	121
schematic library	121
selected PCB pins	617
reports	
netlist compare	158
Reports	158

Index

repour polygon	507	SMD to corner design rule	542
re-routing existing tracks.....	593	SMD to plane design rule	542
reset PCB origin	446	topology design rule	540
resistor		updating.....	581
shunt, simulation	260	via style design rule	542
simulation.....	237	width design rule	543
resolution		routing directives.....	145
design database.....	441	RSHUNT	260
resolving schematic errors.....	140	rule scope	
resource		about.....	528
about defaults	37	how to narrow the scope (compound)	532
editing, example	40	strategies for setting.....	531
resources		using wildcards.....	531
about.....	36	S	
room definition		scaling waveforms.....	212
creating from the schematic	566	schematic	
design rule	551	design verification	136
working with rooms	568	forward-annotation	146
rotate		PLD design.....	319
while moving objects	484	PLD libraries	319
rotating a PCB component	564	preparing for PCB layout	142
rotation		reassigning designators.....	142
setting step size	454	resolving errors.....	140
routing		updating component changes	120
changing track and via parameters	588	updating from the PCB.....	642
corners design rule	539	schematic capture	
density map	587	fundamentals	57
diagonal autorouting.....	603	schematic components and libraries.....	106
layers design rule.....	539	Schematic Editor	
maintaining connectivity	584	about.....	53
manually.....	583	features	53
manually routing your design.....	588	working in	72
moving components onto grid.....	586	schematic library editor	114
plowing through polygons.....	592	scope	
predictive track placement	590	net identifier	126
preparing for.....	586	secure device	355
priority design rule	540	select	
protecting pre-routes	599	PCB objects	469
push and shove	592	schematic objects.....	81
re-routing.....	593	selection	
setting the grid.....	586	creating a PCB array.....	476
shape-based routing, manual	586	creating an schematic array	84
shortcuts	589	cumulative	453
SMD neck down design rule	542		

displaying in PCB	470	schematic title block.....	67
from schematic to PCB	472	sheet entry	59
from the PCB panel.....	472	about.....	101
highlight in full.....	456	connectivity	61
highlight with net color.....	456	creating connectivity	125
jump to in PCB.....	465	sheet part filename	119
PCB Editor.....	466	sheet symbol	
PCB Selection Wizard.....	474	about.....	101
Schematic Editor.....	79	using.....	124
selecting a PCB component from the		Sheet Symbols/Port Connections	130
schematic.....	566	sheet template	
strategies in PCB.....	469	creating.....	69
strategies in schematic.....	81	specifying a default	64
working with in schematic	83	sheet templates	68
servers		sheet units.....	68
installing.....	46	sheets	
removing	47	connecting	125
starting and stopping	47	shift step size	455
working with	45	short circuit constraint.....	556
sessions		shortcut	
monitoring open documents	27	dragging a PCB object.....	483
set PCB origin	446	dragging a schematic object	93
routing	455	shortcut keys	
setting up		about.....	36
AC analysis	188	shortcuts	
DC analysis	191	frequently used in schematic	98
DC operating point analysis	193	locating a PCB component	491
Fourier analysis	202	PCB	487
impedance plot analysis	205	PCB list	489
Monte Carlo analysis.....	194	routing	589
noise analysis	204	schematic	95
parameter sweep analysis.....	198	schematic list	96
schematic workspace.....	64	selection.....	82
simulation analyses	183	selection in PCB	470
temperature sweep analysis	201	shunt resistor, simulation.....	260
transfer function analysis	203	signal integrity	
transient analysis	185	checking the rules.....	614
share product terms	354, 355	design rules.....	552
sharing		impedance constraint.....	552
a Design Database	25	overshoot, specifying.....	552
sheet		setting up to check.....	613
adding a sub-sheet.....	124	signal base value, specifying	553
borders.....	66, 67	signal flight time, specifying	553
schematic sheet size	66	signal slope, specifying	554

Index

signal top value, specifying	554	SPICE variables.....	256
stimulus, specifying.....	554	troubleshooting convergence.....	252
undershoot, specifying	555	using measurement cursors.....	218
viewing the waveforms	615	viewing frequency and phase	216
signal layers.....	450	virtual PLD.....	385
silkscreen layers	451	voltage sources	219
SimCode		warnings	255
compiling	265	simulation results window.....	211
language definition.....	269	SIMWARN	260
language syntax.....	272	single layer mode	457
Simlist.TXT file	265	singular matrix error	
simple hierarchy	130	resolving.....	260
simulation		sinusoidal simulation source	220
component, creating	247	size	
components	235	schematic sheet.....	66
creating digital simulation models	263	Smart Technology	
digital circuits.....	262	about.....	32
digital components	246	SmartDoc	
fails to simulate	252	about.....	34
models	235	SmartTeam	
read-only component fields.....	247	about.....	35
simulation		SmartTool	
adding information to the netlist	209	about.....	32
compiling a SimCode model	265	SMD	
current sources	219	PCB components	509
displaying data points.....	217	placing pads.....	496
displaying waveforms	211	SMD neck down design rule	542
errors	255	SMD to corner design rule	542
getting started.....	180	SMD to plane design rule	542
integration method	255	snap grid	
interpreting sweep results.....	218	PCB	447
limits.....	179	Schematic Editor	67
mathematical analysis and plots.....	206	software	
model support.....	178	registering.....	7
number notation	181	solder mask	
PLD absolute (ABS) file	358	expansion design rule	546
PLD design.....	359	layer, about.....	451
PLD fault simulation	385	source	
scale factors	181	equation-defined.....	233
setting up and running	183	exponential	226
SimCode example	266	frequency modulation.....	228
SimCode language definition	269	linear dependent	230
sources.....	219	non-linear dependent	233
specifying the nodes.....	184	periodic pulse	222

piece-wise-linear	224	sub-sheet	
sinusoidal	220	about	123, 124
sources		adding	124
constant	210, 219	supply current	
simulation, using	219	simulation	260
special strings		simulation override	260
converting on screen	65, 456	suppress product terms merging	355
entering their contents, schematic	67	sweep results	
PCB list	503	interpreting	218
schematic list	69	synchronizer	
using in schematic	68	errors, resolving	152
SPICE		using	146
adding information to the netlist	209	warnings	146
errors	255	synchronous	
Gmin stepping failed	253	PLD simulation	369
iteration limit reached	253	syntax	
multipliers	253	highlighting	49
singular matrix	253	system font, schematic	105
source stepping failed	253	system requirements	6
suggested reading	261	T	
troubleshooting convergence	252	team	
variables, setting	256	creating a team member	25
warnings	255	sharing a Design Database	25
SPICE compatibility	178	teardrops	
split power plane		adding	608
creating	595	TEMP	258
SRCSTEP	259	temperature sweep analysis	
stacks, pad	496	setting up and running	201
state machine		template	
PLD design	347	about	68
status bar		adding to clipboard	65
PCB coordinates	446	changing	71
step size	455	creating a schematic template	69
STEP_OFF	271	identifying current	67
STEP_ON	271	removing	71
string		setting preferred	71
changing in PCB	502	updating	71
jump to in PCB	465	testpoint	
PCB	501	about	605
placing in PCB	502	clearing all	608
special, PCB list	503	finding	607
special, schematic list	69	including on the PCB	605
strings		placing automatically	607
converting on screen	65		

Index

reporting location of.....	608	placement mode.....	494
setting up the testpoint report.....	634	placing.....	493
testpoint style design rule.....	546	polygon.....	506
testpoint usage design rule.....	547	routing width design rule.....	543
text		TRANMNS.....	259
about PCB strings.....	501	TRANMXS.....	260
adding company details.....	68	transfer function analysis	
component text fields.....	113, 119	setting up and running.....	203
editing on the sheet.....	78	transformer	
find and replace on sheet.....	89	simulation.....	244
jump to in PCB.....	465	transient analysis	
library component fields.....	113	setting up and running.....	185
moving the pin text.....	64	Transient Analysis	
schematic library fields.....	119	troubleshooting failures.....	255
showing hidden on sheet.....	112	transistor	
text editor		simulation.....	239
languages.....	48	transition time	
syntax highlighting.....	49	simulation.....	259
Text Editor.....	48	simulation override.....	260
document options.....	49	transmission line	
thermal relief		simulation.....	244
to power planes.....	594	transparent layers.....	457
tiling prints/plots.....	156	Troubleshooting	
timestep too small.....	255	SPICE convergence.....	252
options to change.....	258	TRTOL.....	258
timing violations		TRYTOCOMPACT.....	258
messages.....	260	TTMNTYMX.....	260
title block		TURBO bit.....	341
adding company details to the sheet.....	67	tutorial	
TNOM.....	258	breaking a wire.....	80
toolbar		creating a custom template.....	69
creating, example.....	40	finding and replacing schematic text.....	89
toolbars		global editing in PCB.....	480
about.....	36	placing a wire.....	75
topology		removing a wire corner.....	80
design rule.....	540	select schematic by area.....	82
TPMNTYMX.....	260	selecting an individual schematic object.....	82
track			
about.....	493	U	
break track.....	484	UIC option.....	210
changing.....	496	enabling.....	187
corners.....	494	specifying.....	210
dragging with component.....	455	un-connected pin constraint.....	556
look-ahead segment.....	590	undershoot	

- specifying for signal integrity..... 555
- undo
 - setting schematic editor stack size..... 65
- Undo
 - PCB 491
 - schematic..... 98
 - setting PCB editor stack size 454
- union
 - PCB component union 567
- units
 - PCB coordinates..... 446
 - PCB resolution 446
 - PCB workspace size..... 446
 - Schematic Editor 68
 - toggling PCB between imperial and metric 446
- un-routed nets constraint 556
- update
 - schematic from PCB 642
- update PCB
 - from the schematic 146
- update schematic
 - components 120
 - from the PCB 642
- URC transmission line 245
- Use DeMorgan 354
- use net color for highlight 456

V

- vector photoplotters..... 637
- verifying the schematic 136
- vertex
 - adding..... 80
 - removing 80
- via
 - about..... 497
 - changing 499
 - connecting to a plane..... 595
 - display of holes 452
 - maximum via count design rule 549
 - placing..... 498
 - via style design rule..... 542
 - vias under SMT constraint 549
- vias
 - blind/buried 498

- view menu
 - PCB options 461
- violations
 - finding 612
- virtual device 353
- virtual PLD simulation 385
- visible grid
 - Schematic Editor 67
- visible grids
 - display of..... 452
 - PCB 448
- VNTOL 258
- voltage controlled current source 230
- voltage controlled sine source 231
- voltage controlled square wave source..... 232
- voltage controlled triangle wave source 232
- voltage controlled voltage source 230
- voltage-controlled switch 241

W

- warning messages..... 255
 - displaying at run time 260
- waveform analysis window
 - using..... 211
- waveforms
 - single cell, all cells 215
- waveforms
 - displaying 211
 - displaying data points..... 217
 - displaying together 215
 - identifying 217
 - scaling 212
 - using measurement cursors..... 218
- window
 - splitting..... 15
- Windows
 - plotter drivers from..... 641
- wire
 - about..... 100
 - auto wire placement mode..... 76
 - auto-junction..... 77
 - breaking..... 80
 - connectivity 60
 - placement modes 76

Index

removing a corner	80
wiring	
about.....	59
Wizard	
PCB Component.....	519
PCB Maker.....	524
PCB selection.....	474
workspace	
PCB coordinates.....	446

X

XNF.....	358
XOR gates	354
XSpice.....	262
XSpice.....	246

Z

zoom	
PCB	461