SPECCTRA®

Version 7.0

User's Reference

COOPER & CHYAN TECHNOLOGY, INC.

Version

SPECCTRA 7.0

Publication Date

December 1996

Table of Contents

About This Manual

This manual describes how to use SPECCTRA® software products from Cooper & Chyan Technology, Inc. (CCT) to place and route printed circuit board (PCB) designs.

Audience

This manual is written for SPECCTRA users who are familiar with the current methods and practices used to design a PCB.

Using This Manual

The *SPECCTRA User's Reference* contains the following chapters and appendixes.

- Chapter 1, “How SPECCTRA Works” provides an overview of SPECCTRA products, including information about using SPECCTRA in the PCB design process and product features.
- Chapter 2, “Starting SPECCTRA” describes how to start SPECCTRA and how to run SPECCTRA using batch scripts, and explains the SPECCTRA startup options.
- Chapter 3, “Evaluating Component Placement” describes techniques to help you detect and correct placement violations, and explains how to use the built-in analysis tools to achieve optimum placement.
- Chapter 4, “Meeting Advanced Technology Design Requirements” describes techniques to help you successfully autoroute printed circuit boards that use SMD components.
- Appendix A, “Troubleshooting the Design File” explains how to find and correct problems in your design file.
- Appendix B, “Crosstalk and Coupled Noise Concepts” explains how SPECCTRA supports parallel segment and tandem segment crosstalk rules.

- Appendix C, “SPECCTRA Colors and Fonts” describes the default colors and fonts used by SPECCTRA and how you can change them.

Product Features

The following table describes the icons used in this manual to identify features that are available in various product options.

Icon	Feature
ADV	Advanced Rules (ADV) option
DFM	Design for Manufacturing (DFM) option
FST	Fast Circuit (FST) option
HYB	Hybrid (HYB) option
Place	AutoPlace product

Conventions Used in This Manual

The following fonts, characters, and styles have specific meanings throughout this manual.

- **Boldface** type identifies text that you type exactly as shown, such as SPECCTRA command names and keywords.

  For example, **write** is a command name and **session** is a keyword.

  Use the **write session** command to create the session file.

  Command examples that appear on a separate line are not bold. For example

  ```
  write session design.ses
  ```

Boldface also identifies commands that you choose in menus. The ⇨ symbol indicates a menu command. For example, the phrase "click **Rules** ⇨ **Layer** ⇨ **Clearance**" tells you to choose Layer in the Rules menu and Clearance in the Layer cascade submenu.

- *Italic* type identifies titles of books and emphasizes portions of text.

  See the *SPECCTRA Design Language Reference* for information about SPECCTRA design file syntax.

  Italicized words enclosed in angle brackets (<>) are placeholders for keywords, values, filenames, or other information that you provide. For example

  write session <*filename*>

  You replace <*filename*> with the actual path and filename for the session file you want SPECCTRA to write.

- `Courier` type identifies prompts, messages, and other output text that appear on your screen. For example, if you misspell the command name *define* as *defin*, an error message appears.

  ```
  ‘Syntax error in command: token 1 = defin’
  ```

  `Courier` type also identifies operating system commands and switches. For example, `specctra` is a command name and `-do` is a switch.

  ```
  specctra -do cmds.do design.dsn
  ```

  `Courier` type enclosed in brackets ([ ]) identifies keys on your keyboard and mouse buttons.

  For example, `[Shift]` means the shift key. The carriage return key is labeled "Enter" on some keyboards and "Return" on others. This manual uses `[Enter]`.

  Mouse buttons are identified by two uppercase letters enclosed in brackets.

`[LB]`	left button
`[MB]`	middle button
`[RB]`	right button

  If you have a 2-button mouse, press `[Alt] and [RB]` simultaneously when you see `[MB]`.

Special Terms

The following special terms are used in this manual.

- The word *enter* used with commands means type the command and press `[Enter]`.

  “Enter the command **grid wire 1**” means

  1. Type **grid wire 1**.
  2. Press `[Enter]`.

- *Click* means press and release the left mouse button.
- *Click-middle* means press and release the middle mouse button.
- *Click-right* means press and release the right mouse button.
- *Drag* means press and hold the left mouse button while you move the pointer.
- *Drag* `[MB]`means press and hold the middle mouse button while you move the pointer.
- *Double-click* means press and release the left mouse button twice in rapid succession.
- *Click twice* means click twice on the same place in the SPECCTRA workspace.
- *Select* in SPECCTRA means a mechanism you use to identify individual objects, such as wires, nets, or components, for exclusive processing by routing or placement commands. When you *select* wires, nets, components, or other objects before using a command, SPECCTRA operates only on the objects that you have selected.
- *Switch* refers to one or more characters that you can use with an operating system command, such as the command you use to start SPECCTRA. A hyphen (-) precedes each command line switch.

Where to Find Additional Information

For additional information about using SPECCTRA, choose an online help topic in the Help menu.

If you need information about SPECCTRA design file syntax, use your Acrobat™ Reader from Adobe System, Incorporated to read the *SPECCTRA Design Language Reference* manual

Your Comments About This Manual

We are interested in your comments and opinions about this manual. Please consider the following questions when you form your comments.

- Is the information organized logically? If no, how could we better organize the information?
- Did you find any inaccuracies or omissions? If yes, what are they?
- What suggestions do you have for improving this manual?

Please send CCT your comments by fax at (408) 252-9565 or, if you are on the Internet, by email to **pubs@cctech.com**. Remember to include the manual's title with your comments.

How to Contact Technical Support

If you have questions about using SPECCTRA, please call the CCT Technical Support group at 1-800-366-7902.

How SPECCTRA Works

Chapter content

This chapter explains the PCB design process using SPECCTRA, and provides information on how the SPECCTRA automatic and interactive tools work. It also includes an overview of the following SPECCTRA products.

- AutoPlace, which places components interactively and automatically
- EditRoute, which edits wires and vias interactively
- AutoRoute, which routes component connections automatically

Using SPECCTRA in the PCB Design Process

SPECCTRA is a family of printed circuit board (PCB) design automation tools that run on UNIX, Windows, and Windows NT platforms. You can use SPECCTRA tools to automatically and interactively place and route dense surface-mounted device (SMD) and through-pin (PTH) designs.

All SPECCTRA products include a graphical user interface (GUI) and the same adaptive, Shape-Based technology. SMD pads, through-pins, wires, and other copper elements are modeled as basic geometric shapes. Each shape can have rules associated with it that determine design constraints such as component spacing and orientations, wire widths and clearances, and timing, noise, and crosstalk controls. The following table explains the benefits of Shape-Based technology.

The Benefits of Shape-Based Technology

Shape-Based benefits	What it means
Accurately models pads, pins, wires, and vias in a shape database instead of a memory-consuming grid map	Minimizes memory requirements
Uses the exact dimensions of shapes	Maximizes the use of available space and accommodates mixed pin pitches and mixed size components
Supports complex hierarchical design rules	Improves manufacturability
Routes gridless or with submil wire and via grids	Maximizes the use of the PCB routing area, which can reduce signal layers

Because SPECCTRA associates design rules with geometric shapes, you do not have to manage and apply rules through traditional grid-mapping techniques. SPECCTRA places and routes without grids or with sub-mil grids, avoiding the massive memory requirements of traditional grid-mapped tools.

Many grid-based autorouters try to complete all connections in each routing pass until the PCB is routed completely. They prohibit crossover and clearance conflicts.

The SPECCTRA autorouter uses a different approach called “adaptive routing.” the autorouter tries to connect all wires in the first routing pass by allowing crossover and clearance conflicts. During each additional pass, the autorouter reduces conflicts by using its intelligent rip-up-and-retry and push-and-shove algorithms.

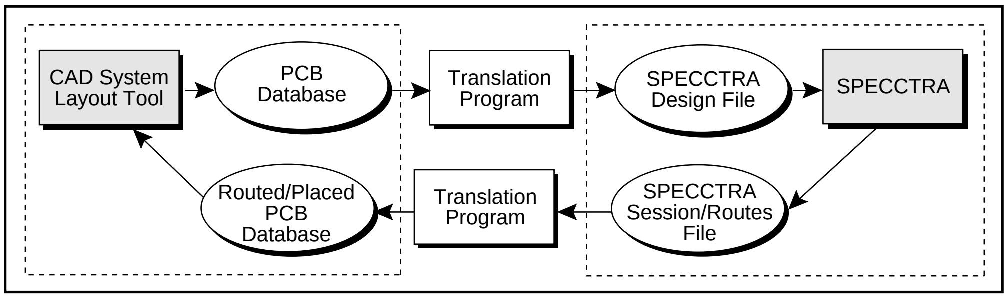
With each pass, the autorouter gathers information and “learns” about the problem areas where conflicts exist, and uses this information to reach its goal of eliminating all conflicts and completely routing the PCB.

Even though it sometimes uses a large number of routing passes, the autorouter achieves a high completion rate because it uses the conflict information from each pass to guide it toward an overall solution.

Understanding the SPECCTRA Interface to Your PCB Design

SPECCTRA works with your PCB layout tools to produce a placed and routed PCB design. After creating a PCB database in your layout system, you translate the database to a SPECCTRA design file. This file contains all the physical information SPECCTRA needs to place and route the PCB. Any design rules you set in the layout system are also translated into the design file.

After placing and routing in SPECCTRA, you translate the routes file and session file. The translator merges these files with your original layout system files, creating placed and routed files that you can use to update the PCB database in your layout system. The process from layout system to SPECCTRA and back again is illustrated in the following diagram.



The PCB Design Process Using SPECCTRA

The SPECCTRA design file is an ASCII text file that contains the netlist, boundary outlines, keepout areas, and component libraries you need to place and route your PCB. It also contains any rules you set in the layout system to establish design specifications for placement and routing.

Note: If you don't plan to place components in SPECCTRA, you must create a PCB database with placed components in your layout system before translating the database to a SPECCTRA design file.

Understanding the SPECCTRA Design File

The SPECCTRA design file contains data that includes the PCB boundary, component footprints, reference designators, padstack definitions, and the netlist. The design file comprises four main elements:

- structure—Includes layer definitions, PCB boundary, via padstack ids, rules, and grid definitions.
- placement—Includes component locations and reference designators.
- library—Includes image (component footprint) definitions and padstack definitions.
- network—Includes the netlist.

The design file can include other data in addition to the main elements. For more information about the structure of the SPECCTRA design file, use your Acrobat™ Reader from Adobe System, Incorporated to read the *SPECCTRA Design Language Reference* manual.

Using Commands to Control SPECCTRA

SPECCTRA is controlled by commands that set rules and determine how a design is placed and routed. SPECCTRA provides several ways for you to enter commands. You can

- Use the mouse to choose commands from menus and edit dialog boxes
- Type commands on the keyboard
- Run a do file that contains a sequence of commands
- Run SPECCTRA using a batch startup script that specifies one or more do files.

When you are learning SPECCTRA and how to construct commands, it is useful to use the menus. Then, you can examine the did file or output window where the commands are recorded and learn the correct command syntax. A did file contains a record of all commands generated by menu selections during a session. Users often enter commands through the menu, then cut and paste these commands from the did file or output window into a do file. This do file is what they use to control the autorouter.

The preferred method of running the autorouter is to use a do file. The do file not only allows you to run the autorouter unattended in batch mode, but it also serves as a record of rules applied to your design. Chapter 3 provides a basic do file you can use to autoroute many of your PCBs. You can modify and add to this do file to meet your design requirements.

You should understand the basic do file before you create your own do file or use the **smart_route** command. The **smart_route** command evaluates and automatically routes your PCB design.

You can use the * and ? characters as wildcards in commands where you want to specify multiple objects. The * wildcard substitutes for any number of characters; the ? wildcard substitutes for a single character.

For example, **protect net sig?** protects the net named sig and all nets with four-character net names that begin with sig. The command **protect net sig*** protects the net sig and all net names that begin with sig.

Placing Components

The AutoPlace product provides automatic and interactive tools for placing and adjusting components on the PCB. It includes the following capabilities.

Automatic capabilities

- Setting placement rules for manufacturability
- Grouping components by connectivity or net connections
- Defining a component floor plan
- Performing initial placement
- Placing decoupling capacitors in patterns
- Interchanging components to improve placement
- Rotating components
- Swapping gate and pin connections to reduce Manhattan lengths
- Generating histograms
- Analyzing component density
- Generating reports

Interactive capabilities

- Drawing rooms for a component floor plan
- Preplacing critical components
- Locking components in place

- Placing components by connectivity
- Moving, rotating, and flipping components
- Trading components
- Aligning components
- Shoving components
- Swapping gate and pin connections
- Undo and Redo

Editing and Routing Interactively

The EditRoute product interactively routes and edits PCB wiring. The EditRoute product includes the following capabilities:

- Routing and rerouting with push and shove, including blind and buried via support
- Pushing and shoving wires and vias
- Copying single and multiple layer wires
- Cutting wire segments
- Deleting nets, wires, and wire segments
- Reducing extra wire bends and notches
- Undo and Redo

The EditRoute product supports a fast-circuit rules options:

- EditFST
  - Minimum and maximum length rule checking
  - Minimum and maximum length rule status display

Autorouting

The AutoRoute product is used to automatically route a PCB design. You apply rules and control routing by using menu commands or a command file for batch operation. The AutoRoute product includes the following capabilities:

- Rip-up and retry with push and shove routing for maximum completion rates
- Routing with or without grids
- Definable via grid
- Definable wire grid
- Automatic route improvement for manufacturability
- 45-degree recornering
- Hierarchical rules-based routing

The AutoRoute product supports four options:

- ADV (advanced rules)
  - Layer assignment of signals
  - Via assignment by net and net class
  - Width and clearance by layer
  - Net and net class rules by layer
  - Time length factors by layers
- DFM (design for manufacturability)
  - Automatic testpoint generation
  - Mitering
  - Spreading
- HYB (hybrid design)
  - Blind and buried via support
  - Vias under SMDs
  - Wirebond support
- FST (fast circuit rules)
  - Crosstalk control by parallelism and accumulated noise rules
  - Timing controls by length or delay
  - Differential pair routing
  - Rules by area
  - Shielding
  - Rounded corners
  - Virtual pin topology control

Starting SPECCTRA

Chapter content

This chapter explains how to begin a SPECCTRA session on both UNIX platforms and Windows and Windows NT platforms. You can start SPECCTRA by using the Startup dialog box, by using the **specctra** command with optional startup switches, or by running a batch script. This chapter explains all three methods.

The Startup dialog box lets you use the mouse to set startup options or search for files. Command line switches are useful when you know the syntax for the options you want to use and the paths and filenames of the files you want to specify. Batch scripts let you run several SPECCTRA sessions unattended.

Before you start SPECCTRA, you must translate your PCB design from your layout system to a SPECCTRA design file. See your SPECCTRA translator manual for information about translating PCB data from your layout system.

Note: On a Windows or Windows NT system, the security key must be attached to your parallel port, and the password file must be installed in your SPECCTRA installation directory or the common subdirectory, before you run SPECCTRA.

Using the Startup dialog box

You can start SPECCTRA by using the Startup dialog box or by using command line switches. This section explains how to use the Startup dialog box to start a new session or to restart a previous session.

Use the Startup dialog box when you want to use the mouse to set startup options or search for files.

To start a new session, you specify the design file. To restart a session, you specify the session file. You can specify a do file containing SPECCTRA commands that automate part or all of the session. When restarting a session, you can also specify a wires file or routes file and placement file containing routing and placement information from the previous session.

To specify a file, you can either enter the filename in the filename data entry box or click the Browse button and locate the file in the Select File dialog box (on UNIX systems) or the Open dialog box (on Windows or Windows NT systems).

When you enter a filename, you must include a path name if the file is not located in the current directory.

The Startup dialog box that appears when you first start SPECCTRA displays only the basic startup options. To see additional options, click the More Options button. The dialog box expands to display additional options. See "Understanding Startup Options" later in this chapter for explanations of all the SPECCTRA startup options.

➢ To start SPECCTRA using the Startup dialog box

1. Open the Startup dialog box by using one of the following:

   On a UNIX system, type **specctra** in a shell window, and press `[Enter]`.

   On a Windows or Windows NT system, double click the SPECCTRA icon in the SPECCTRA group.

2. Specify a design file or a session file in the Design / Session data entry box.

3. Set the startup options you want to use.

   You can specify a do file and a routes or wires file. Click the More Options button if you want to use additional startup options.

4. Click the Start SPECCTRA button.

When you click a browse button, a file browser dialog box appears. You can use the mouse to select a file or change directories, or use the keyboard to edit or enter any part of a path or filename.

➢ To choose a file on a UNIX system

1. If the file you want is not in the current directory, use one of the following to change directories:

   Double-click a directory name in the Directories list, and repeat until you are in the directory you want.

   Enter a directory path in the File Selection data entry box.

   Enter the directory path in the Filter data entry box, and press `[Enter]`. SPECCTRA updates the names shown in the Directories and Files lists.
2. Double-click the filename in the Files list, or type the filename in the File Selection data entry box.
3. Click OK.

On Windows and Windows NT systems, SPECCTRA uses the standard Open dialog box for file browsing.

➢ To choose a file on a Windows or Windows NT system

1. If the file you want is not on the current drive, choose the drive you want in the Drives pulldown menu.
2. If the file you want is not in the current directory, double-click a directory name in the Directories list, and repeat until you are in the directory you want.
3. Double-click the filename in the File Name list, or type the filename in the File Name data entry box.
4. Click OK.

Using Command Line Switches

You can start SPECCTRA by using the Startup dialog box or by using command line switches. This section explains how to use command line switches to start a new session or to restart a previous session.

If you know the paths and filenames for the files you want to specify, and understand the startup switches, you can bypass the Startup dialog box and start SPECCTRA from the command line. The **specctra** command line syntax is

- **specctra** {[<*switch*>]} [<*design_or_session_file*>]
  The <*design_or_session_file*> is the name of the design file, if you are starting a new session, or the name of the session file, if you are restarting a previous session.
- A <*switch*> is one of the optional SPECCTRA command line switches explained in "Understanding Startup Options" later in this chapter.

If a file is not located in the current directory, you must specify a path with the filename.

See "Understanding Startup Options" later in this chapter. for explanations of all the SPECCTRA startup options.

➢ To start SPECCTRA on a UNIX system

❑ Enter the **specctra** command in a shell window, including the switches you want to use and the design or session filename, and press [Enter].

➢ To start SPECCTRA on a Windows or Windows NT system

1. Double click the File Manager icon.
2. Double click the SPECCTRA installation directory name.
3. Click **File** ⇨ **Run** to open the Run dialog box.
4. Enter the **specctra** command in the Command Line data entry box, including the switches you want to use and the design or session filename.
5. Click OK.

On a Windows NT system, you can also start SPECCTRA by in a command prompt window.

1. Double click the Command Prompt icon to open a command prompt window.
2. Change into the bin directory under the SPECCTRA installation directory (if this directory is not included in your path variable).
3. Enter the **specctra** command with the switches you want to use and the design or session filename.

Using SPECCTRA Batch Scripts

You can use a batch script to run several SPECCTRA sessions in sequence, unattended. At the start of each session, SPECCTRA reads a design file and a do file. To run SPECCTRA batch sessions, prepare the following files:

- Design file for each session
- Do file for each session
- Batch script

The batch script controls each session with a **specctra** command that includes a design file, a do file, and any switches you want to use.

To avoid license failures during a batch session on a UNIX system, you can use the `-lr` switch to specify the licenses you need for the session. See "Understanding Startup Options" later in this chapter for explanations of the SPECCTRA startup options.

Creating a batch script

A batch script is a text file that contains a series of **specctra** commands. Each command controls a session using the do file and any switches you include. Use a text editor to create the batch script.

Each command line in a batch script starts a new session. The `-do` switch specifies a do file containing commands that control the session. The `-quit` switch ends the session after the last command in the do file. You must use -quit at the end of each command line in the script.

If a file is not located in the current directory, you must specify a path with the filename.

The following example shows a batch script for UNIX and Windows NT systems.

```
specctra design1.dsn -do des1.do -quit
specctra design2.dsn -do des2.do -quit
specctra design3.dsn -do des3.do -quit
specctra sample.dsn -do sample.do -quit
```

Because of limitations in Windows 3.1, you must use DOS, and disable Windows entry password protection, to start multiple SPECCTRA sessions on a Windows system. The following example shows a batch script for Windows 3.1 systems.

```
win specctra design1.dsn -do des1.do -quit
win specctra design2.dsn -do des2.do -quit
win specctra design3.dsn -do des3.do -quit
win specctra sample.dsn -do sample.do -quit
```

In these examples, the specified design files and do files are located in the current directory, and the SPECCTRA installation directory is included in the path variable.

In the following examples, the design files and do files are not in the current directory. You can use the $/ symbols on UNIX systems (\$/. in a C shell), or the $\ symbols on Windows and Windows NT systems, if the do file is in the same directory as the design file.

```
specctra /samples/sample.dsn -do \$/sample.do -quit
specctra c:\samples\sample.dsn -do $\sample.do -quit
win specctra \samples\sample.dsn -do $\sample.do -quit
```

If the SPECCTRA installation directory is not included in your path variable, you must specify the full path to the **specctra** executable file. In the following example, the installation directory is cct_install.

```
/usr/local/cct_install/bin/specctra sample.dsn -do sample.do -quit
c:\cct_install\bin\specctra sample.dsn -do sample.do -quit
win c:\cct_install\bin\specctra sample.dsn -do sample.do -quit
```

Before running a batch script on a UNIX system, use the shell **chmod** command to make the script an executable file

```
chmod +x <filename>
```

Creating a SPECCTRA .ini file

On a Windows 3.1 system, you must exit Windows after each batch session before you can start the next batch session. If your batch script runs multiple SPECCTRA sessions, you must create a text file called SPECCTRA .ini in the \WINDOWS directory before running the script.

You can also change SPECCTRA's default fonts in this file. See Appendix C for details.

To exit Windows at the end of a SPECCTRA session, the SPECCTRA .ini file must contain the following lines:

```
[QA]
ExitWindows = 1
```

The SPECCTRA .ini file causes you to exit Windows when you quit SPECCTRA. If you want to use SPECCTRA interactively without running a batch script, rename the SPECCTRA .ini file or set `ExitWindows = 0`.

- `ExitWindows = 1` forces Windows to exit to DOS at the end of the SPECCTRA session.
- `ExitWindows = 0` does not force Windows to exit at the end of the SPECCTRA session.

The base name of the SPECCTRA .ini file must be the same as the name of your SPECCTRA executable. For example, if your executable is named spxx.exe, the default fonts file must be named spxx.ini.

➢ To create a SPECCTRA .ini file in your Windows directory

1. Open a text editor such as Window's Notepad.
2. Enter the lines you want to include in the file.

   You can also change SPECCTRA's default fonts in this file. See Appendix C for details.
3. Save the file in the Windows directory.
4. Exit the text editor.

Running SPECCTRA Batch Sessions

The procedure for starting a batch script depends on the platform you are using.

➢ To start a script on a UNIX system

- ❑ Enter the script filename in a shell window, and press `[Enter]`.

➢ To start a script on a Windows NT system

1. Double-click the Command Prompt icon in the Main window.
2. Change into the SPECCTRA installation directory, and make sure your batch script file is in this directory.
3. Enter the script filename, and press `[Enter]`.

➢ To start a script on a Windows 3.1 system

1. Exit Windows. You should be at the DOS prompt.
2. Change into the SPECCTRA installation directory, and make sure your batch script file is in this directory.
3. Enter the script filename, and press `[Enter]`.

Understanding SPECCTRA Startup Options

The following table describes the SPECCTRA startup options you can set using the Startup dialog box or the **specctra** command.

Startup dialog box field	**Switch**	**Use this option to...**
Design or Session File		Runs SPECCTRA using this design file or session file.
Wires or Routes File	-w <*file*>	Restarts SPECCTRA using this wires file or routes file.
Placement File	-place <*file*>	Starts or restarts SPECCTRA using this placement file. This file data overrides the placement descriptor data of the design file.
Do File	-do <*file*>	Begins session by running commands from this do file.
No Graphics	-nog	Runs SPECCTRA without the GUI (on UNIX systems only).
Quit After Do File	-quit	Exits SPECCTRA after running the last command in the do file.
No Preroutes	-nowire	Ignores prerouted wires in design file.

Startup dialog box field	**Switch**	**Use this option to...**
Show Usage	-help	Displays help information on command line switches in the output window.
Don't Strip Orphan Shapes	-noclean	Keeps orphan shapes (copper without net assignments). Without this option, SPECCTRA removes all orphan shapes.
Simplify Polygons	-sim	Replaces small polygons with rectangles. Useful for component pads with complex shapes. A design with many polygon-shaped pads can slow performance.
Password File		Specifies a password or license file. Use this option only to specify a password or license file that is not in the usual location or in your search path. You can
		• Click *Default* to use SPECCTRA's default search path.
	-p <*file*>	• Click *Node Locked* and enter the path and filename for a node-locked password file.
	-lf <*file*>	• Click *Network* and enter the path and filename for a network license file.
Did File	-did <*file*>	Specifies the file where SPECCTRA commands are logged during an SPECCTRA session. The default filename is the month, day, and time with a .did extension.
Message Output File	-o <*file*>	Redirect the output messages to the file you specify.
Status File	-s <*file*>	Specifies the autorouter status file, which contains routing status information. The default status file is monitor.sts.
Color Mapping File	-c <*file*>	Specifies the color map file that defines the display colors and patterns for design objects and graphical features in the work area. The default filename is color.std in the current directory. If you do not provide a color map file, SPECCTRA uses colors and patterns defined in the design file, or uses internal defaults.

The following table describes additional startup options you can set using the **specctra** command on a UNIX system.

Switch	Description
-design *<file>*	Specifies the design file.
-display *<host:display>*	Directs the SPECCTRA GUI from the host system where you are running SPECCTRA to a display terminal where you want to display SPECCTRA graphics.
-docmd "*<command>*"	Runs a SPECCTRA <*command*>. You can include multiple commands by separating them with semi-colons (;). If this switch precedes a -do switch, SPECCTRA runs these commands before the do file commands. If a -do switch precedes this switch, SPECCTRA runs the do file commands first.
-exact	Restricts the SPECCTRA session to only those feature licenses specified with the `-lr` switch. If rules embedded in your design file require other feature licenses, a warning message is displayed when you start SPECCTRA.
`-ignore_net`	Causes SPECCTRA to ignore the input net names of wires read from a wires file. The wires are identified with the net names currently associated with the connected pins.
-lr <*feature*>	Causes SPECCTRA to immediately obtain this license. If no license is available, SPECCTRA asks if you want to wait, cancel startup, or continue without the license. The licenses you can specify with this switch are

Feature	Description
ViewBase	Basic SPECCTRA license
RouteBase	Basic AutoRoute license
RouteADV	AutoRoute ADV license
RouteDFM	AutoRoute DFM license
RouteHYB	AutoRoute HYB license
RouteFST	AutoRoute FST license
EditBase	Basic EditRoute license

Switch	Description	
	EditFST	EditRoute FST license
	IPlaceBase	Interactive AutoPlace license
	PlaceBase	Automatic AutoPlace license
-geometry [*W***x***H*] [[**+**|**-**]*x* [**+**|**-**]*y*]]	Sets the size and position of the SPECCTRA window. You can specify the width (*W*) and height (*H*) of the window, and the number of pixels horizontally (*x*) and vertically (*y*) between the top left corner of the window and the top left (**+**) or bottom right (**-**) corner of the screen.	

Evaluating Component Placement

Chapter content

This chapter explains how to detect and correct placement violations and how to achieve the best placement you can by using built-in analysis tools. SPECCTRA provides reports and graphical analysis tools to help you evaluate your placement results. You can also use the SPECCTRA autorouter to evaluate component placements.

Using the Placement Status Report

The placement status report includes the session date and time, the report date and time, a history of all automatic placement operations, a list of Manhattan length improvements, and the CPU time for each placement operation.

Use the status report to monitor the component placement process and observe improvements that result from each automatic operation. For example, the column labeled Manhattan lists the changes in Manhattan distances following each placement command. Manhattan distances are calculated for all placed components.

Correcting Placement Violations

Placement violations are caused by placing components

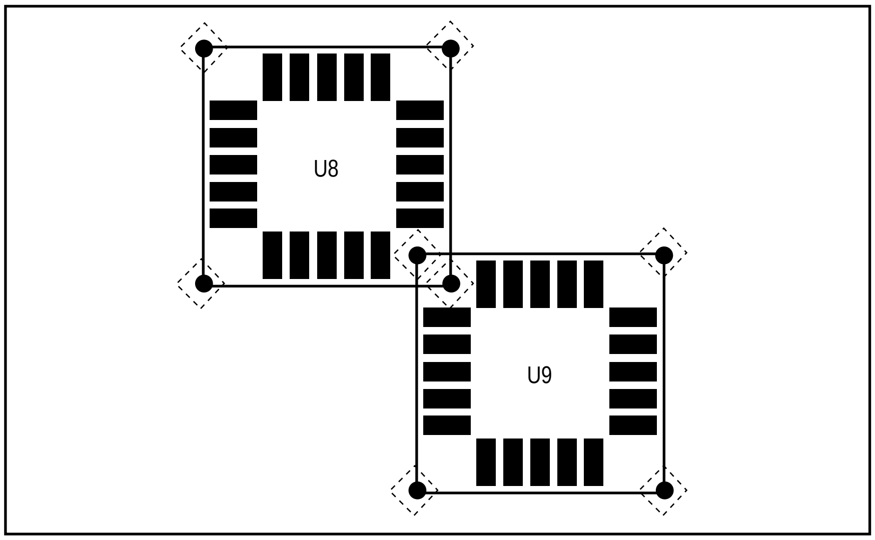
- Outside the PCB boundary.
- With less intercomponent spacing than a rule permits.
- Overlapping keepout areas.
- In the wrong orientation.
- On the wrong side of the PCB.
- In an area where a power dissipation or height rule is exceeded.
- In an area where the components are excluded.
- Outside the area where the components should be included.

In addition to violations, SPECCTRA displays warning messages components are placed off grid (if you defined a placement grid), outside a soft-bound room, or over the wrong power plane.

Understanding violation markers

When a component is involved in a violation, SPECCTRA identifies the component by attaching diamond shapes at the corners of its outline and changing the outline from a thin line to a thick line. All violations except off-board violations are indicated in this manner.

If you manipulate a component interactively, only that component is marked when a violation occurs. The following figure shows an example of two components manipulated and marked for violations due to an overlap.



Diamond Shapes Mark Components Involved in Violations

You can review the conflicts report to determine the type of placement violation for a particular component.

➢ **To determine the type of placement violation**

1. Click the Select Component button on the tool bar.
2. Click on the component in question.
3. Click **Report** ⇨ **Conflicts**.
4. Read the violations listed in the report window.
5. Click Close when you are finished with the report window.

If you click **Report** ⇨ **Conflicts** without selecting one or more components, SPECCTRA generates a report of all components involved in violations.

Eliminating violations

The solutions for eliminating certain types of violations are obvious. If a component is marked as a violation because it lies outside the PCB boundary, you must move it within the boundary to remove the violation. Other types of violations can require more analysis to determine a solution.

A simple method for analyzing spacing violations uses the Measure mode. You can zoom-in and use the pointer to measure distances between components that are involved in spacing violations. If the spacing between components is only slightly less than the spacing rule, you can move the component to a different location or consider relaxing the rule to eliminate the violations.

The PCB spacing, permitted side, and permitted orientation rules are set through the **Rules ⇨ PCB** menu. You can review the global (PCB) rules settings by viewing the rule report.

You can also use the report facilities to determine which rules are being violated. When you generate a component report, both the properties assigned and the applicable rules are included in the report.

➢ To generate a component report

1. Click **Report ⇨ Component**.
2. Choose a component from the Component list in the Report Component dialog box.
3. Read the component information in the report window.
4. Click Close when you finish viewing the report.

Following an automatic placement operation, some components can remain outside the PCB boundary due to rules that prevent their placement. For example, if a component is excluded from one area of the PCB, and height constraints and power supply restrictions prevent placement in other areas of the PCB, there could be no alternative but to leave that component outside the PCB boundary.

When placement restrictions must be applied to certain areas of the PCB, they are applied by creating rooms and assigning rules to the rooms. You can confirm how rooms are defined and the rules that are assigned by viewing the room report.

➢ To view a room report

1. Click **Report ⇨ Rooms**.
2. Read the room information in the report window.
3. Click Close when you finish viewing the report.

Using the Autorouter to Evaluate Placement

Another method for evaluating component placement is to autoroute the design and analyze the results. Use the Place and Route buttons in the tool bar to switch between place and route environments.

An autorouting trial can be a very effective way to evaluate component placement. Autoroute three or more passes and evaluate the status file results. If the number of conflicts after the first pass versus the number of connections is less than five, the current placement has a good probability of routing to completion.

Sometimes when you route the PCB, a small number of unconnects remain because of poor placement of a few components. You can try to make room for the unrouted connections by shifting those few components and restarting the autorouter. If the design is very dense this strategy will probably fail. In this case, the best strategy is to delete *all* wiring, shift the components, and autoroute the complete design from the beginning.

Analyzing placement results

To evaluate placement problems that are confined to certain areas of the PCB, zoom-in and visually inspect them.

Poor component placement can produce the following symptoms:

- Unroutes due to insufficient routing channels
- SMD components without access to pin escapes
- Conflicts that can't be removed at or near pin exits
- Excessive numbers of vias due to poor component orientation

You can also use the autorouter status report to determine possible placement problems. For example, if unroutes exist after the first five routing passes, look at the pins that are left unrouted to see if the unrouted component is placed over or near a routing keepout.

If conflicts remain at a constant number after 50 routing passes, and additional passes result in no further reduction, inspect the connections involved in the conflicts. Dependent on the number of conflicts, it might be easier to route these connections interactively. This routing symptom is usually the result of a component placement problem.

Evaluating net lengths

If you have high speed rules turned on in Setup, you can use the place lengths report to evaluate timing delays for nets that have length or delay rules assigned in the autorouter. SPECCTRA evaluates both routed and unrouted guides and nets. You can use this report to evaluate how well your length or delay rules apply to the current placement before you route the board.

The place lengths report lists all maximum, minimum, and match length rules. The report lists all nets that have assigned length or delay rules, the evaluated total length or delay of each net, and warning messages for any potential violations.

➢ To generate a place lengths report

1. Click **Report** ⇨ **Place Lengths**.
2. Read the length and delay information in the report window.
3. Click Close when you finish viewing the report.

If your design file does not contain length or delay rules, you must assign timing length rules in the autorouter before you can generate a place lengths report. SPECTRA calculates place lengths or delays for all placed components, and uses dashed ratsnest lines to show any nets, groups, or guides that violate the maximum length rules.

Meeting Advanced Technology Design Requirements

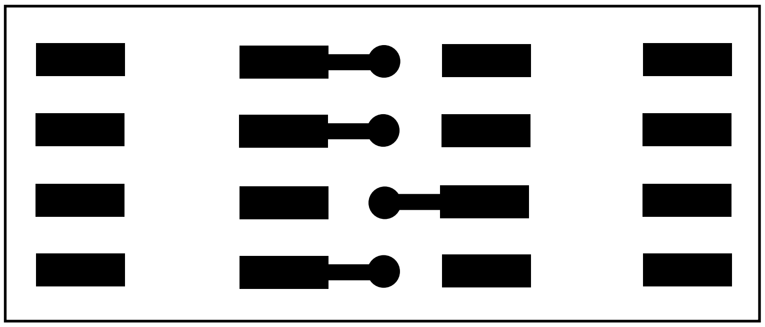
Chapter content

The techniques described in this chapter can help you autoroute printed circuit boards that are designed with SMD components. Many of the techniques are from expert users who frequently apply the autorouter to difficult benchmark PCBs.

Layouts designed with SMD components are often more difficult to autoroute than those designed with through-hole components. One difficulty is that SMD pads can be accessed on only one layer. Accessing an SMD pad from another layer of the PCB requires a via. For this reason, most techniques that address SMD autorouting involve improving via access to SMD pads.

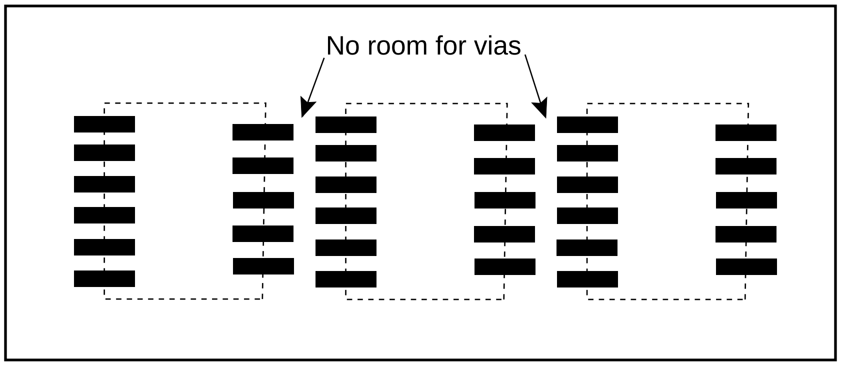
Placing Components Effectively

One of the most important aspects of successful SMD design is choosing a component placement that effectively manages the via space needed to escape SMD pads. If adequate via space between components is not provided, you can expect high conflict counts and unroutes. The following figure shows how the space between the pads of adjacent SMD components should allow at least one via site. For SMD components having a large pin count, more space is required to escape into the PCB. A minimum of three to four via sites should be allowed between the pads of adjacent components that have large pin counts.



Allow for at Least One Via Between Components During Placement

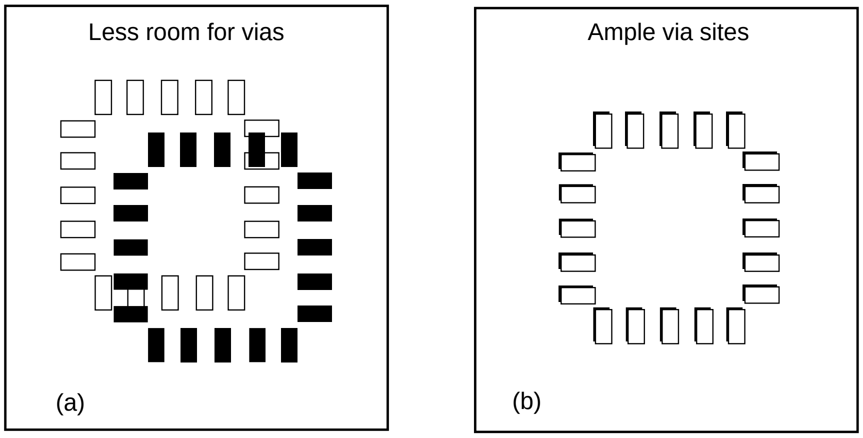
Don’t *brickwall* SMD components by placing components too close together. The following figure shows an example of brickwalling, where the autorouter has no chance to access the SMD pads from another layer by using vias between components.



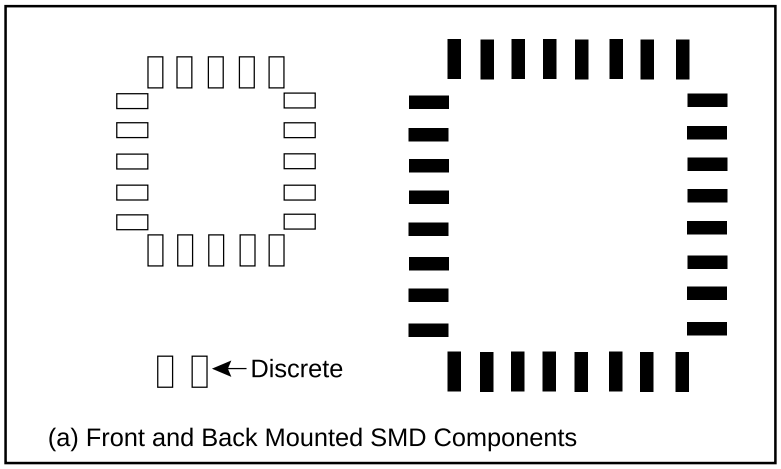
Brickwalling SMDs Leaves No Room For Vias Between Components

Mounting SMDs back-to-back

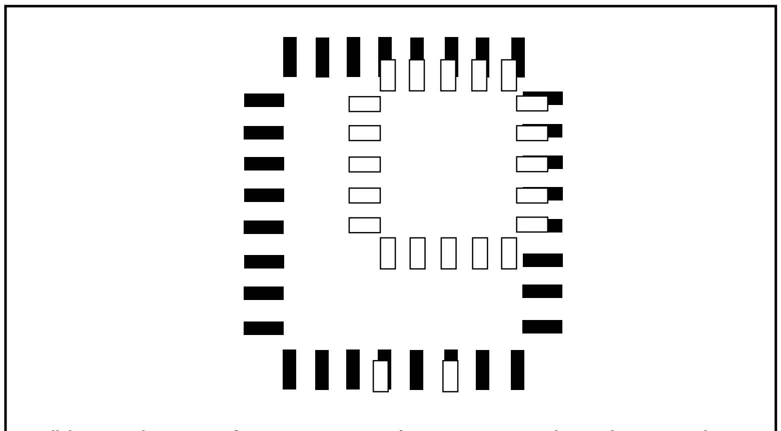
When SMD components are mounted on both sides of the PCB, the pads should be opposite each other as shown in the following figures. The goal is to create as many potential via sites as possible.



Identical Back-to-Back SMDs



(a) Front and Back Mounted SMD Components



(b) Good SMD Placement Overlays Front and Back Footprints

Different Sized SMD Components Should Mount Back-to-Back

Controlling SMD-to-Via Escape

The methods for creating SMD-to-via escapes that you can consider for your designs are described below.

Prerouting wire-to-via escapes

You can create wire-to-via escapes by using EditRoute, then protect the wiring before you route. This method is used when you want to develop and maintain a predictable fanout pattern that meets specific design requirements.

Using the fanout command

You can generate SMD-to-via escapes by using the **fanout** commands Using this method no prerouting is required and the autorouter has more flexibility to generate a solution. This method is effective for designs with components on both sides of the PCB. The **fanout** command is explained in more detail later in this section.

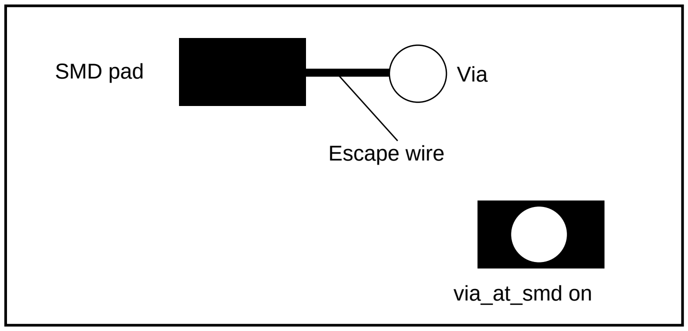
Note: You also can use the **smart_route** command, which automatically generates SMD-to-via escapes and routes your design.

Having no predefined escape wiring

The autorouter automatically routes SMD-to-via escapes, as needed, to complete connections. This method works best for two-signal-layer designs where the autorouter needs maximum flexibility to complete connections.

A connection is made from an SMD pad to a via with an escape wire, or to a via attached directly under the SMD pad as a via_at_smd.

The following figure illustrates the two methods for escaping SMD pads.



Two Methods for Escaping SMD Pads

Using a combination of methods for escaping

You can choose to escape some components with a prerouted pattern, use the **fanout** command on other components, and then allow the autorouter to find escapes for all remaining connections.

Using the fanout command

The **fanout** command should be used only when the PCB uses four or more signal layers. For these multilayer designs, the **fanout** command speeds up the autorouter and helps assure completion. When you use fanout with designs that have fewer than four layers, the potential number of via sites is reduced. This can cause poor completion on very dense designs.

Rip-up and retry fanout

Use fanout with the rip-up and retry option on dense designs where via space is limited. Use the command:

```
fanout 5
```

If more than 5% of the SMD pins don’t have a via after fanout, there could be a problem. A fanout strategy requiring at least two wires between vias improves your rate of routing completion. On very dense designs, you can interactively correct any fanout failures by using EditRoute and then use AutoRoute to route the PCB. To view SMDs that have a fanout failure use the command:

```
highlight no_fanout
```

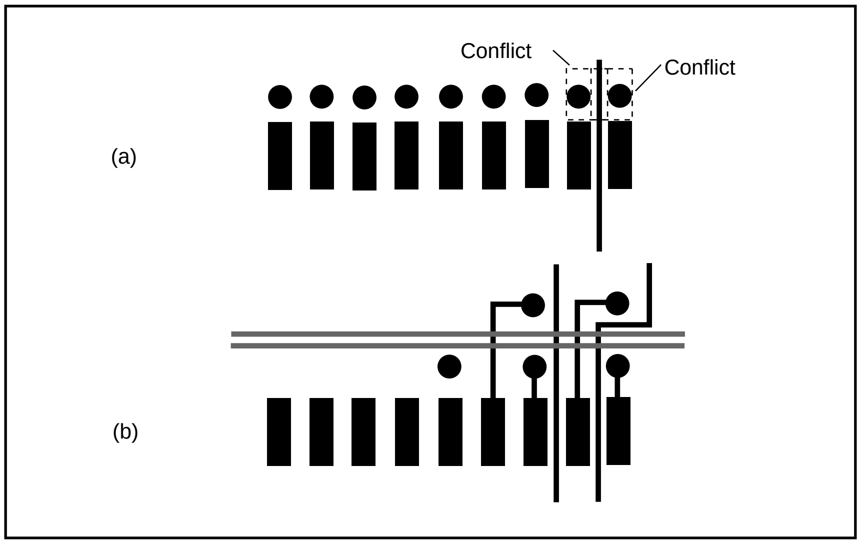
Based on your inspection of the highlighted pins, you could consider several possible improvements. Changes in the placement, via grid, via size, and clearance rules might help. If changes are made, the **fanout 5** command can be used again, followed by the **highlight no_fanout** command, to evaluate results. This process can continue until all, or nearly all, of the SMDs are successfully fanned out. When fanout is satisfactory, autorouting can begin.

Avoid protecting fanout vias

Fanout vias should not be protected. The autorouter needs the flexibility to rip-up and retry via locations to reach a good solution, especially when routing difficult designs.

Managing the via grid

Avoid creating via barriers. A via barrier occurs when vias are placed so that no wires can pass between them. When a via barrier is created, the autorouter has to work harder, routing is slower, and a poor completion rate results. A via grid that is too fine can create a via barrier. Use the **grid smart** command or choose a via grid spacing that allows two wires between vias. In the following figure, the top illustration shows a via barrier and the bottom illustration shows the same component with improved via fanout.



Via Grid can (a) Create a Barrier of Fanout Vias or (b) Allow Routing Between Pads Without Conflicts

➢ To set a via grid spacing for routing

- ❑ Consider the wire grid spacing and the final via grid spacing, and use the **grid smart** command.

  Example:

  ```
  unit mil
  grid smart (wire 1) (via 25)
  fanout 5
  route 25
  ```

When the **grid smart** command is used at the beginning of a do file, it allows two wires between vias, during the initial route passes. If manufacturing rules prohibit setting the via grid to 1 mil, use the smallest grid spacing that is an even multiple of the wire grid and satisfies your manufacturing rules.

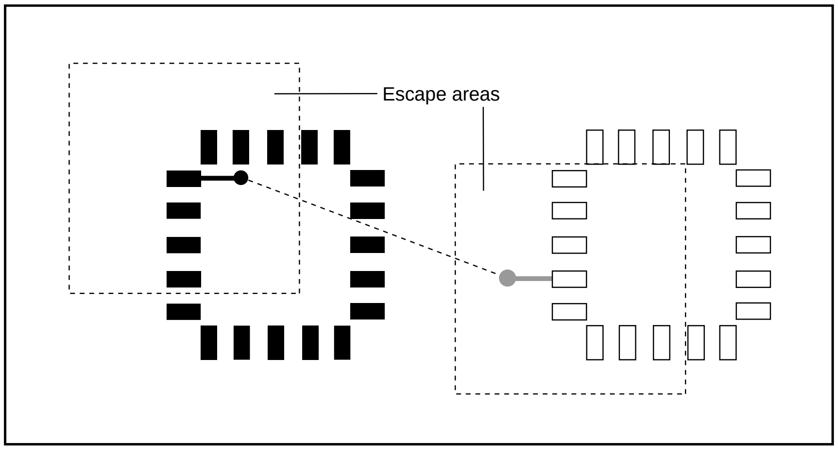
Wiring Forbidden on External Layers

Some multilayer designs prohibit routing on external layers except for SMD-to-via escape wiring. The autorouter automatically meets this requirement when SMD pads are on layers that are unselected.

For example, if you have a PCB with six signal layers (L1 through L6) and layers L1 and L6 are unselected, use the **unselect** command:

```
unselect layer L1 L6
```

When a connection with an SMD pad is routed, an area is created around the pads as shown in the following figure to allow SMD escape.



SMD Pad Escape Perimeters Used When Layers Are Unselected

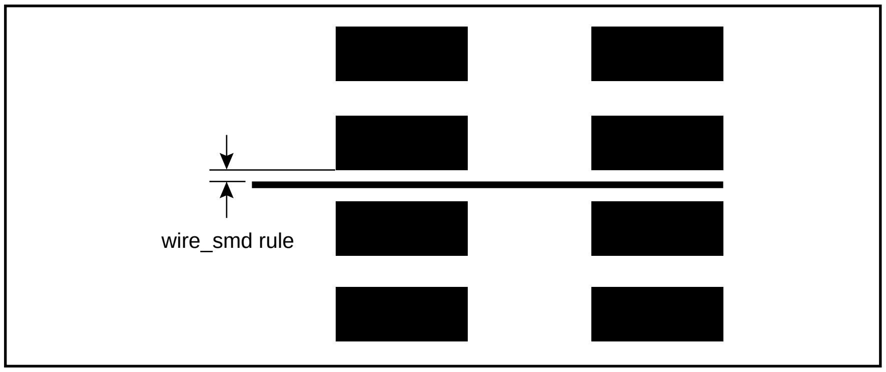
All sides of the escape area are equal in length to twice the smd_escape distance. The default smd_escape distance is 0.25 inches (0.635 cm). You can set the smd_escape distance to a different value by using the **change** command. For example, if you wanted to change the escape distance to 0.125 inches, you would use the command:

```
change smd_escape 0.125
```

Optimizing Design Rules

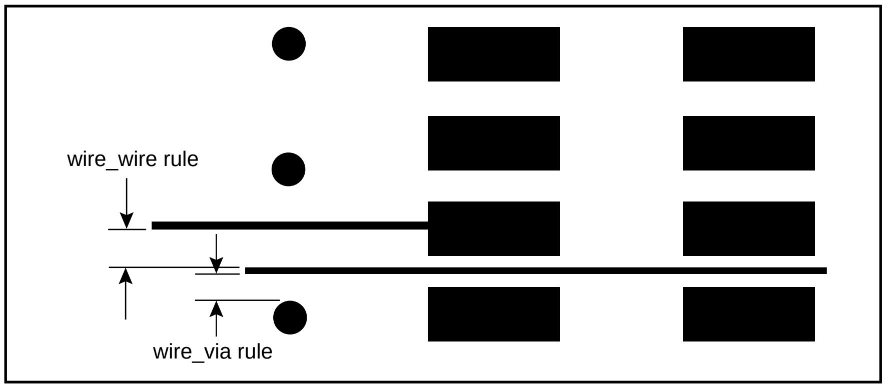
The autorouter offers more flexibility for setting rules by allowing separate control of rules such as **wire_pin** clearance, **wire_smd** clearance, and **wire_via** clearance. You can also control widths and clearances by layer, by net, and by individual connection.

If manufacturing rules permit, set the **wire_smd** clearance rule to allow one wire between SMD pads as shown in the following figure.



Wire-to-SMD Rule Should Allow One Wire Between Pads

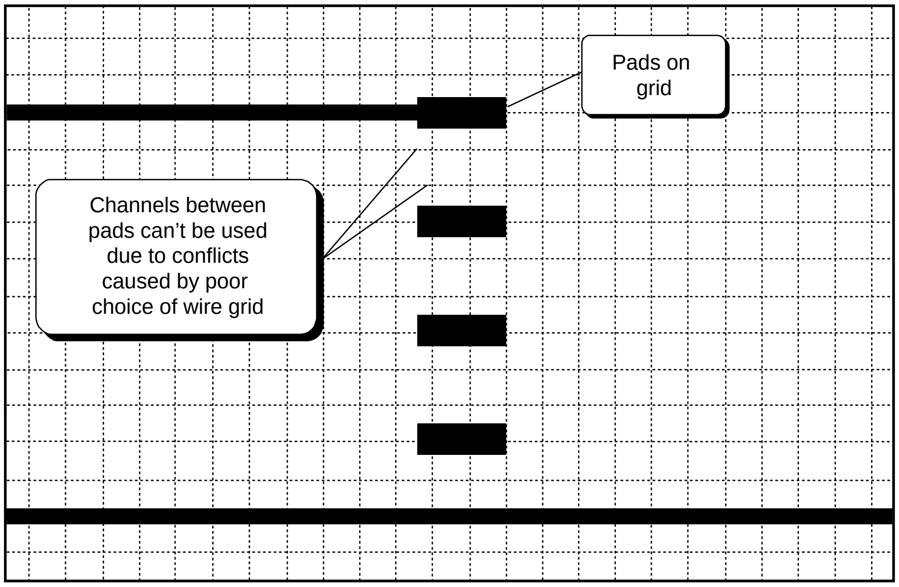
You should set the **wire_pin** and **wire_via** rules to allow two wires between adjacent vias where possible. The following figure illustrates a good wiring pattern that results when the **wire_via** rule is properly set.



Wire-to-Wire and Wire-to-Via Rules Should Allow Two Wires Between Adjacent Vias

Choosing a wire grid

If possible, set the wire grid to zero and autoroute gridless. This provides the best chance for success when autorouting difficult PCBs. If you must use a grid, choose the smallest grid you can. If possible, a grid equal to 0.001 inch (0.0254 mm) should be used. If you can’t use one mil, choose a spacing that allows the best flow of wires through component pins. The following figure illustrates gridding pitfalls you should avoid.


Pads on grid
Channels between pads can’t be used due to conflicts caused by poor choice of wire grid

Grids Between Pads Cannot be Used

Caution: If you route gridless, the layout system must be able to accommodate the gridless wiring, or roundoff problems will occur. A 0.0005 inch wiring grid is sufficient for most designs.

Autorouting Two-Layer Designs

The autorouter's built-in strategy is excellent for two-layer designs. Therefore, improvement of autorouting results for two-layer designs depends on good placement techniques, particularly for layouts with a large number of SMD devices.

The **fanout** command should not be used on two-layer PCBs. You usually have better results when you allow the autorouter to use vias as required. On two-layer designs a large number of routing passes should be used as long as conflicts follow a downward trend. The typical number of passes for a two-layer design is 200 to 300 passes. The autorouter is very fast on two-layer PCBs, so the time to complete each pass is short.

Troubleshooting the Design File

The information in this appendix is intended to be used to update and correct design files that predate SPECCTRA Version 5.0 interfaces. Whenever possible, either correct the data in the PCB layout system or update your interface instead of using the methods described in this appendix.

Always make a backup copy of your design file before you edit it. If you modify your design file and add rules or data that is not supported in your layout system, the results are unpredictable when you return to the layout system.

The SPECCTRA design file is a text file in ASCII format. You can edit the design file using text editor.

Note: Before you edit a design file, familiarize yourself with the SPECCTRA file structure and the design language syntax by referring to the *Design Language Reference*. Use your Acrobat Reader from Adobe Systems, Incorporated to view this manual.

Correcting improper keepout definitions

The five types of keepouts that can be defined in a design file are described in the following table.

Keyword (type)	Description
keepout	Prohibits wires, vias, and components in the keepout region.
via_keepout	Prohibits vias in the keepout region.
wire_keepout	Prohibits wires in the keepout region.
bend_keepout	Prohibits wire corners in the keepout region.
place_keepout	Prohibits components in the keepout region.

The exact location of keepouts within the file is important. For example, when you're reworking a completed design (already placed and routed) that has keepouts defined globally in the structure section of the design file, if you unplace the components, the keepouts remain behind on the PCB. When you automatically place the components, the components' locations are different and the keepouts are in the wrong positions.

To correct this, the keepouts must be moved from the structure section to the images in the library section of the design file to make them part of the image definitions. The process can be a little tedious, since keepout regions defined in the structure section of a design file use absolute coordinates; a keepout region attached to an image must be defined relative to the image's origin.

The following example shows how a 0.250 square via_keepout definition (in bold type) appears in the structure section of a design file.

```
(PCB demo8
 (structure
 (grid via 0.025)
 (boundary (rect pcb .450 .050 1.85 1.6))
 (boundary (rect signal 0.550 0.150 1.75 1.5))
 (Via v25 (spare testpt1 testpt2))
 (rule (width .008) (clearance .008 ))
 (layer s1 (type signal)
 (direction horizontal))
 (layer p1 (type power)
 (use_net +5V GND))
 (layer s2 (type signal)
 (direction vertical))
 (via_keepout (rect s1 0.7 0.3 0.95 0.55))
 (grid wire 0.000 s1)
 (grid wire 0.005 s2)
)
.
.
.
```

If you transpose this keepout to the lcc20 image in the library section of the file, it appears as shown in the following example:

```
(library
 (image lcc20
 (pin p25x75 1 0.0000 0.1500)
 (pin p25x50 (ARRAY 2 3 1 0.0500 0.1500 0.05 0.0))
 (pin p25x50 (rotate 90) (ARRAY 4 8 1 0.1500 0.1000 0.0 -
0.05))
 (pin p25x50 (rotate 180)(ARRAY 9 13 1 0.1000 -0.1500 -
0.05 0.0))
 (pin p25x50 (rotate 270)(ARRAY 14 18 1 -0.1500 -0.1000
0.0 0.05))
 (pin p25x50 (ARRAY 19 20 1 -0.1000 0.1500 0.05 0.0))
 (via_keepout (rect s1 -0.125 -0.125 0.125 0.125))
)
```

The coordinates for the rectangular via_keepout region in the previous example are relative to the image's origin, which in this example is the component center.

Correcting an improper power layer definition

Occasionally a design file is missing a power net definition. This is always the result of either an error in the PCB layout database or a fault in the translator. If a power net is not defined by a **use_net** statement, SPECCTRA assumes it is a signal net. When you start SPECCTRA, it warns you if it finds a signal net with more than 150 pins.

SPECCTRA uses connectivity to prioritize components for automatic placement operations. Because power nets usually contain a large number of pins, SPECCTRA considers them to be the largest nets in the design. If SPECCTRA uses power nets to determine component connectivity, the placement results can be unsatisfactory.

If possible, you should always correct this type of design file error in your PCB layout system. A correct power net definition, showing a single power layer that is split between +5V and GND, is shown in the next example.

```
(PCB demo8
 (structure
 (grid via 0.025)
 (boundary (rect pcb .450 .050 1.85 1.6))
 (boundary (rect signal 0.550 0.150 1.75 1.5))
 (Via v25 (spare testpt1 testpt2))
 (rule (width .008) (clearance .008 ))
 (layer s1 (type signal)
 (direction horizontal))
 (layer p1 (type power)
 (use_net +5V GND))
 (layer s2 (type signal)
 (direction vertical))
            (grid wire 0.000 s1)
)
```

Removing a huge component from the design file

Sometimes a graphic object in the PCB layout system gets translated to the design file as a component. The object could be for documentation purposes, part of the PCB silk-screen, or some other graphic object that is not part of the network or component data.

If the object is approximately the same size as the PCB and is defined as a component, automatic placement is unable to place other actual components over the erroneous component. If possible, you should always correct this error in your PCB layout system.

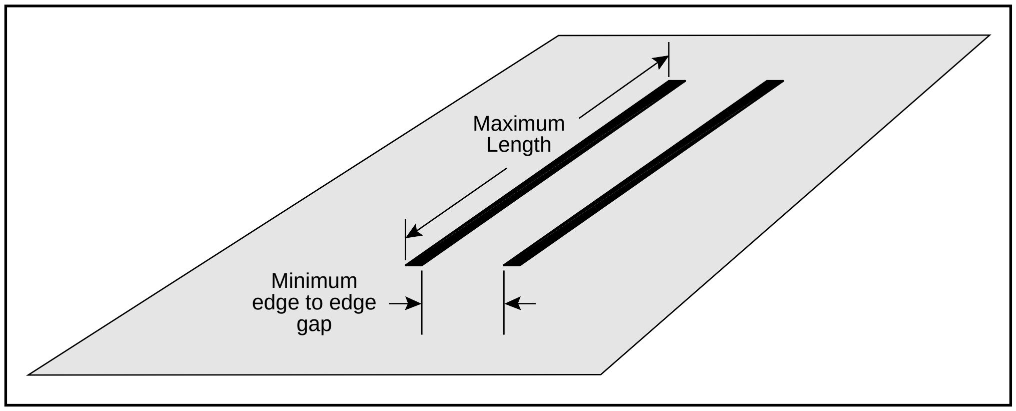
An example of what this type of design file error looks like is shown in the next example.

```
 (placement
 (component NO_COMP
 (place U5 1.0 0.75))
 (component lcc20
 (place U1 0.8000 0.4000 1 0)
 (place U2 1.5000 1.1000 2 0)
 (place U3 1.5000 0.4000 2 0)
 (place U4 0.8000 1.1000 1 0))
)
(library
 (image NO_COMP)
 (image lcc20
 (pin p25x75 1 0.0000 0.1500)
 (pin p25x50 (ARRAY 2 3 1 0.0500 0.1500 0.05 0.0))
 (pin p25x50 (rotate 90) (ARRAY 4 8 1 0.1500 0.1000 0.0 -
0.05))
 (pin p25x50 (rotate 180)(ARRAY 9 13 1 0.1000 -0.1500 -
0.05 0.0))
 (pin p25x50 (rotate 270)(ARRAY 14 18 1 -0.1500 -0.1000
0.0 0.05))
 (pin p25x50 (ARRAY 19 20 1 -0.1000 0.1500 0.05 0.0))
)
```

Crosstalk and Coupled Noise Concepts

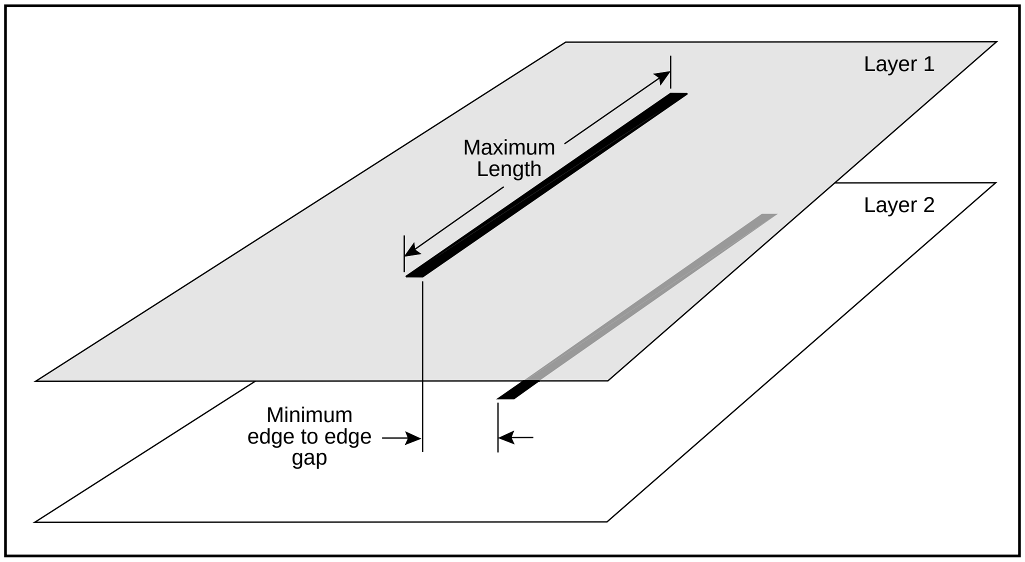
SPECCTRA's fast-circuit (FST) option enforces crosstalk rules. Crosstalk rules are applied at the pcb level, and to individual layers, classes of nets, nets, groups of fromtos, and fromtos. SPECCTRA supports two forms of crosstalk, segment control and noise control, by using parallel segment and tandem segment crosstalk rules.

Parallel_segment crosstalk rules control parallelism between wire segments on the same layer. You supply gap and length parameters as shown in the following figure.



parallel_segment Crosstalk

Tandem_segment crosstalk rules control parallelism between wire segments on adjacent signal layers. There can be no intervening power layers. You supply gap and length parameters as shown in the following figure.

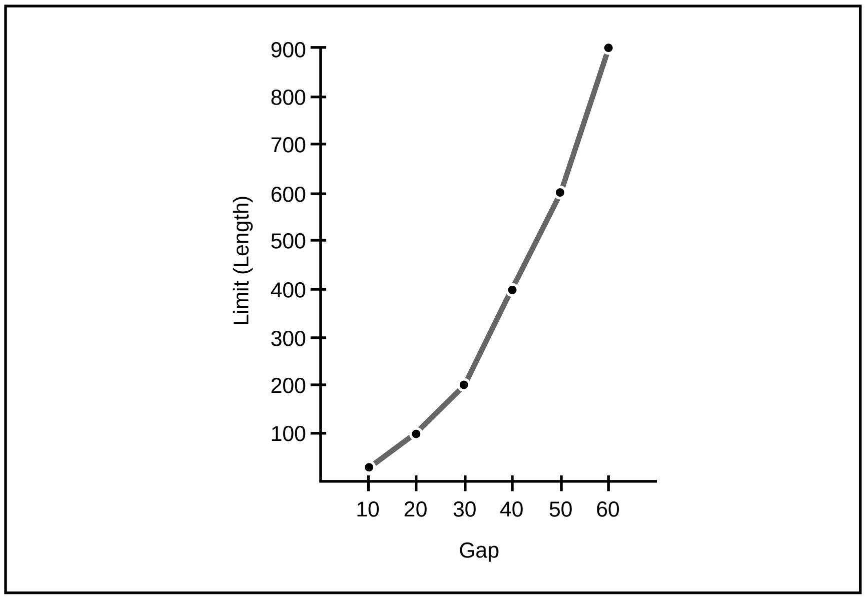


tandem_segment Crosstalk

Using Multiple Crosstalk Rules

Multiple parallel_segment or tandem_segment crosstalk rules can approximate conditions that vary as a function of parallel wire length and gap. The following series of parallel_segment rules approximate a crosstalk characteristic that varies as a function of parallel length and gap. Multiple tandem_segment rules are applied in the same way to control interlayer crosstalk.

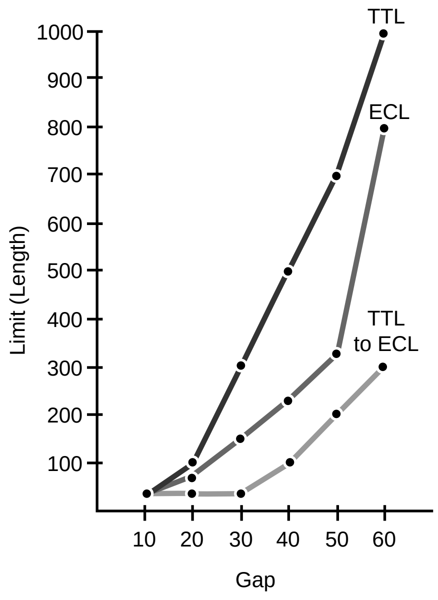
```
rule pcb (parallel_segment (limit 25) (gap 10))
rule pcb (parallel_segment (limit 100) (gap 20))
rule pcb (parallel_segment (limit 200) (gap 30))
rule pcb (parallel_segment (limit 400) (gap 40))
rule pcb (parallel_segment (limit 600) (gap 50))
rule pcb (parallel_segment (limit 900) (gap 60))
```



Parallel Segment and Gap Combinations to the Left of the Curve are Violations

Controlling Class-to-Class Crosstalk

Class-to-class crosstalk rules control crosstalk between different net classes or between nets of the same class. The following figure shows how multiple class-to-class rules can be applied simultaneously to approximate the crosstalk characteristics for several different technologies.



Gap	Length		
	TTL	ECL	TTL to ECL
10	25	25	25
20	100	75	25
30	300	150	25
40	500	225	100
50	700	325	200
60	1000	800	300

Multiple Class-to-Class Crosstalk Rules Are Applied Simultaneously

Controlling Coupled Noise

Noise coupling between wires on a printed circuit board can be the cause of circuit malfunction or failure. To minimize and control noise coupling during autorouting, you must specify the maximum noise that a receiving net can tolerate. To determine whether an excessive noise condition exists, the contributions from all noise sources are accumulated. If the total exceeds the maximum noise specification, a violation exists.

Some nets are noisier than others because of the circuit technology used or due to circuit function, but any net in a design is a potential noise source. Each noise transmitting net must have one or more weights assigned that correspond to the amount of noise transmitted as a function of the distance to a parallel receiving net.

SPECCTRA allows you to specify these factors in global (PCB) rules, by class, by net, and by fromto. The autorouter can reduce or eliminate both inter- and intra-layer coupled noise. The relationship between gap, wire length, layer factor, and total noise coupled onto a net is shown by the following coupled noise expression:

$$\text{NoiseR} = \Sigma\,(L * \text{weight (gap)} * \text{layer\_factor})$$

where:

NoiseR = the sum of all coupled noise components in a receiving net

L = the measured wire length over which coupling occurs

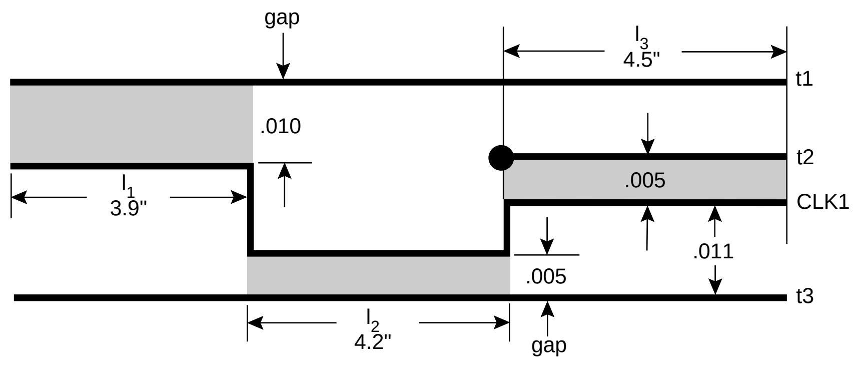
weight(gap) = the factor proportional to noise generated by transmitting net per unit length, at a specified gap

layer_factor = the inter or intra-layer value depending on PCB characteristics; you can specify multiple values in a layer matrix.

The coupled noise expression shows how noise is computed for a receiving net where one or more noise transmitting nets exist. The additional term, not included in this expression but used by SPECCTRA to determine where routing violations occur, is the maximum noise rule (noise budget) for the receiving net.

SPECCTRA computes the noise that intrudes onto a receiving net from neighboring wires. The computation is made for each transmitting wire by multiplying its noise weight by its parallel or tandem length. The weight-times-length product is multiplied by the layer_factor to adjust the noise value as a function of the routing layer. Contributions from all transmitting nets are accumulated for a receiving net and the sum is compared to the maximum noise budget for the receiving net. When coupled noise accumulation on a net exceeds the maximum noise rule, the condition is a violation.

The following figure illustrates a coupled noise violation where the accumulated coupled noise on net CLK1 equals 819.6 millivolts while the max_noise rule for the net is 600 millivolts.



Note:
t1, t2, t3 are noise transmitting nets
CLK1 is the noise receiving net

Commands:
unit inch
rule pcb (parallel_noise (gap .010) (weight 25)) (parallel_noise (gap .005) (weight 83))

Computation:

$\text{Noise}_R = (l_1 * 25) + (l_2 * 83) + l_3 * 83)$

$= (3.9 * 25) + (4.2 * 83) + 4.5 * 83)$

$= 819.6$ millivolts

Coupled Noise Computation

SPECCTRA Colors and Fonts

SPECCTRA uses default fonts and colors to display the menu bar, dialog boxes, tool bar, and other elements of the GUI. On UNIX systems you can change the colors and fonts. On Windows and Windows NT systems, you can change the fonts.

Note: The colors used by the SPECCTRA menu bar, dialog boxes, tool bar, and other components are specified differently from the PCB objects displayed in the work area. The colors of PCB objects are specified by using the SPECCTRA color map file. See the **write** command in Chapter 6 for details.

Changing default colors on Windows and Windows NT systems

You can use the Control Panel to change default colors on a Windows or Windows NT system.

Changing default colors on UNIX systems

You can change the colors used for the SPECCTRA GUI by editing your X resources file. This file is usually named .Xdefaults and located in your home directory. See your system administrator for the location on your system.

The colors of SPECCTRA window can be specified either with color names or with color numbers. The color names you can use are usually listed in /usr/lib/X11/rgb.txt on your X server. Color numbers can be specified with a hexadecimal value that represents the amount of red, green, and blue intensity on a scale from 00 to ff. Precede the hexadecimal value with the pound sign (#).

Example hexadecimal color entries are shown in the following table.

Table B-1 Hexadecimal Color Entries

Value	Color
#ff0000	red
#00ff00	green
#0000ff	blue
#000000	black
#ffffff	white
#888888	medium gray

The fonts that are available on your X server are determined with the **xlsfonts** command. See your X Window System documentation for further details.

You must add the string "=ascii" to the end of the font name. The default values for the resources you can set for SPECCTRA are listed in the following tables.

Table B-2 Default Color Resources

Resources	Color
cct_da*idleColor:	green
cct_da*busyColor:	red
cct_da*interruptibleColor:	orange
cct_da*pausedColor:	yellow
cct_da*modalFormUpColor:	#a020f0
cct_da*stopColor:	red
cct_da*dofileColor:	pink

Table B-3 Default Fonts

SPECCTRA Resource	X Resource	Value
Default font	cct_da*fontList:	*-helvetica-bold-r-normal--*-120-*=ascii
Layer Panel	cct_da*Layers*popup* fontList:	*-helvetica-bold-r-normal--*-100-*=ascii
Status Panel	cct_da*Tools*popup* fontList:	*-helvetica-bold-r-normal--*-100-*=ascii
Scrollable Lists	cct_da*XmList* fontList:	*-courier-medium-r-normal--*-120-*=ascii
Single Line Text	cct_da*XmTextField. fontList:	*-courier-bold-r-normal--*-120-*=ascii
Dialog Box Titles	cct_da*popup*title* fontList:	*-helvetica-bold-r-normal--*-180-*=ascii
Dialog Box Subtitles	cct_da*popup*subtitle* fontList:	*-helvetica-bold-r-normal--*-120-*=ascii
File Selection Boxes	cct_da*filename* textFontList:	*-courier-bold-r-normal--*-120-*=ascii
Report Window Text	cct_da_fileShell*XmText* fontList:	*-courier-medium-r-normal--*-120-*=ascii
Report Window Labels	cct_da_fileShell*XmLabel* fontList:	*-helvetica-bold-r-normal--*-120-*=ascii
Report Window Close Button	cct_da_fileShell*XmPushButton* fontList:	*-helvetica-bold-r-normal--*-120-*=ascii

Changing default fonts on UNIX systems

You can change the fonts used for the SPECCTRA GUI by editing your X resources file. This file is usually named .Xdefaults and located in your home directory. See your system administrator for the location on your system.

To increase the default font size used by SPECCTRA

1. Change into the directory where your .Xdefaults file is located (usually your home directory), start your text editor, and open the .Xdefaults file.
2. Add the following line to the .Xdefaults file:

```
pcb_da*fontList: -helvetica-bold-r-normal--*-140-*=ascii
```

3. Save the modified .Xdefaults file and exit the text editor.
4. Execute the UNIX command

```
xrdb load .Xdefaults
```

When you start SPECCTRA, the menu bar and other window text should be larger. If you want to change the font for other SPECCTRA resources, add additional lines to your .Xdefaults file to include the X resource and font values listed in the previous table.

Changing default fonts on Windows and Windows NT systems

SPECCTRA for Windows and Windows NT uses the internal font default shown in the following table. The face name, height, width, and weight specifications for each font group are set for each of three graphic resolution levels: 640 by 800 by 600, and 1024 by 760 or higher.

Table B-1 Default Fonts Used by SPECCTRA

Font group	Where it's used in SPECCTRA	Font Used (Face Name)	Height	Width	Weight	Display Resolution
MonoFont	Output windows	Courier New	15	0	400	(1024 x 768)
	Report windows	Courier New	13	0	400	(800 x 600)
	List box	Courier New	11	0	400	(640 x 480)
MonoBoldFont	Text field	Courier	15	0	600	(1024 x 768)
	Command entry area	Courier New	13	0	600	(800 x 600)
		Courier	12	0	500	(640 x 480)
DefaultFont	All others	Arial	13	6	600	(1024 x 768)
		Arial	9	5	500	(800 x 600)
		Arial	9	5	400	(640 x 480)
SmallTextFont	Ghost text	Courier New	11	7	400	(1024 x 768)
		Courier New	9	7	400	(800 x 600)
		Courier New	9	7	400	(640 x 480)

Cell height and width are relative values, expressed in logical units, that define a square area around the font. If the width is zero, Windows and Windows NT matches the aspect ratio of the physical device against the digitization aspect ratio of the available fonts to find the closest match, which is determined by the absolute value of the difference.

A font’s weight value determines its thickness. Weight values range from zero to 900. The common values for the font weight are listed in the following table.

Table B-2 Weight Values and Description

Value	Description
0	Don’t care
100	Thin
200	Extra light
300	Light
400	Normal/Regular
500	Medium
600	Semibold
700	Bold
800	Extra bold
900	Black/Heavy

Changing default fonts

You can change any or all of the default font specifications by creating an .ini file in your Windows directory.

The .ini file can define the fonts used for

- Output windows, report windows, list boxes (MonoFont)
- Text fields, text in the Command Entry area (MonoBoldFont)
- Length rule indicators (SmallTextFont)
- All other text (DefaultFont)

The following is an example of a SPECCTRA .ini file. The [GUI] section title is required. Each additional entry can appear in the file only once. SPECCTRA uses internal defaults for any specification not entered in the file. All text to the right of a semicolon (;) is a comment and is ignored by SPECCTRA.

```
[GUI]
; Resets Default Font specifications
DefaultFontFaceName = Arial
DefaultFontHeight = 9
DefaultFontWidth = 5
DefaultFontWeight = 500
; Resets Bold Font face name
MonoFontFaceName = Monaco
; Resets Small Text  Font face name and weight
SmallTextFontFaceName = Monaco

SmallTextFontWeight = 600
```

See "Creating a SPECCTRA .ini file" in Chapter 2 for details about creating a SPECCTRA .ini file.

Index