SPECCTRA® 8.0

User Guide

July 1998

Trademarks

Table of Contents

About This Manual

This manual describes how to use SPECCTRA® software products from Cadence Design Systems, Inc. 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 printed circuit boards.

Using This Manual

The *SPECCTRA User Guide* contains the following chapters and appendixes.

- Chapter 1, “How SPECCTRA Works” is an overview of the SPECCTRA technology, including information about product features and how to use SPECCTRA in the PCB design process.
- Chapter 2, “Starting SPECCTRA” describes how to start SPECCTRA, explains 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 that can help you

autoroute printed circuit boards that use SMD components.

- Appendix A, “Crosstalk and Coupled Noise Concepts,” explains how SPECCTRA supports parallel_segment, tandem_segment, tandem_noise, and tandem_noise crosstalk rules.
- Appendix B, “Setting Colors and Fonts,” describes the default colors and fonts used by SPECCTRA and how you can change them.

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, keywords, and other syntax elements. In the following example, **connect**, **on**, and **off** are keywords.

  (**connect** [**on** | **off**])

  Syntax examples and command examples that are not entered by you from the keyboard are not bold. For example:

  (boundary (rect pcb 0 0 9000 4000))

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

  See the *SPECCTRA Installation and Configuration Guide* for information about installing SPECCTRA.

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

  <*directory_path_name*>

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

  `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 document uses `[Enter]`.

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

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

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

Special Terms

The following special terms are used in this document.

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

  "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.
- *Select* means to identify objects (such as wires, nets, or components) for exclusive processing by routing or placement commands. When you *select* objects before using a command, SPECCTRA works only on the objects that are selected.
- *Switch* refers to one or more characters preceded by a dash (-). You can use one or more switches when you start SPECCTRA.

Where to Find Additional Information

For information about the SPECCTRA documentation library, and an overview of new features and enhancements, see the online help: *What's New in SPECCTRA 8.0*.

For installation information, see the *SPECCTRA Installation and Configuration Guide*.

For descriptions of known problems and limitations in SPECCTRA 8.0, see the *SPECCTRA 8.0 Release Notes*.

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

For information about starting SPECCTRA, command syntax, menu help, and design methodology, open the online help file: specctra.hlp. Use WinHelp for Windows 95 and Windows NT, or Hyperhelp for UNIX. You can access the online help system from the SPECCTRA Help menu.

Your Comments About This Manual

We are interested in your comments and opinions about this manual. If you comment, please consider the following questions:

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

Please send your comments by fax to (408) 342-5647, or via the Internet by email to **specctra_pubs@cadence.com**. Remember to include the document title with your comments.

How to Contact Technical Support

If you have questions about installing or using SPECCTRA, you can visit the Cadence Web site at

http://www.cadence.com

You can reach Cadence Technical Support for SPECCTRA through the Customer Response Center at (800) 223-3622.

How SPECCTRA Works

Chapter content

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

- EditPlace for interactive and automatic component placement
- EditRoute for interactive wire and via editing
- AutoRoute for autorouting

Using SPECCTRA in the PCB Design Process

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

All SPECCTRA tools include a graphical user interface (GUI) and the same adaptive, ShapeBased technology. SMD pads, through-pins, wires, and other circuit elements are modeled as basic geometric shapes. Each shape can have rules associated with it, which enforce design constraints such as component spacing and orientation, wire width and clearance, timing, noise, and crosstalk. The following table explains the benefits of ShapeBased technology.

Benefits of ShapeBased Technology

Benefit	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, SPECCTRA gathers information and “learns” about the problem areas where conflicts exist to eliminate them and completely route the PCB.

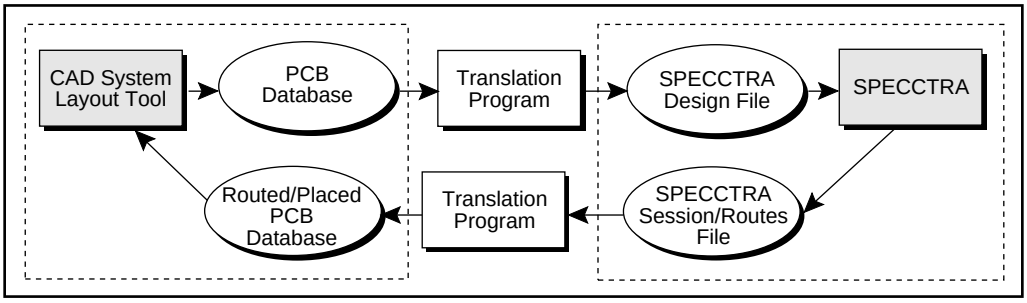
Even though it sometimes uses a large number of routing passes, the autorouter usually achieves a high completion rate in a short period of time because the conflict information is used from each pass to achieve an overall solution.

Understanding the Interface to Your PCB Layout System

SPECCTRA works with your PCB layout tools to produce a placed and routed design. After creating a PCB database in your layout system, you translate the data 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 transferred to the SPECCTRA design file.

After placing and routing in SPECCTRA, you translate the routes file or session file. A translator merges the routes or

session file with your original layout system files, creating a placed and routed design that you use to update the PCB database in your layout system. The flow from layout system to SPECCTRA, and back again to the layout system, is illustrated in the following diagram.



The PCB Design Process Using SPECCTRA

The SPECCTRA design file is an ASCII text file that contains a netlist, boundary outlines, keepout areas, and all component libraries required to place and route your PCB. It also contains any rules you set in the layout system to constrain 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 a netlist. The design file comprises four main elements:

- structure, which contains layer definitions, PCB boundary, via padstack ids, rules, and grid definitions.
- placement, which contains component locations and reference designators.
- library, which contains image (component footprint) definitions and padstack definitions.
- network, which contains the netlist.
- wiring, which contains preroute data. This section is optional if no prerouting is performed in the layout system.

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 to view the *SPECCTRA Design Language Reference* online book.

Using Commands to Control SPECCTRA

SPECCTRA tools are controlled by commands that set rules and determine how a design is placed and routed. There are several ways you can enter commands. You can

- Use the mouse to choose commands from menus and dialog boxes
- Type commands at the keyboard
- Run a do file that contains a sequence of commands

When you are learning about the SPECCTRA tools and learning how to construct commands, it is useful to use the graphical user interface (GUI). Each GUI (menu) command generates a syntactically correct SPECCTRA command. You can examine a *did* file or the output window where the commands are recorded and become familiar with the correct command syntax. A did file is a record of all commands used during a session. You will often generate commands through a menu, then cut and paste the resulting commands from a did file or from the output window into a *do* file. The do file is what you use to control the autorouter.

The preferred method of running the SPECCTRA autorouter is to use a do file. The do file not only allows you to run the autorouter unattended , but it also serves as a record of all rules and commands 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.

Use and understand the commands and their sequence in the basic do file before you create your own do file, or use the **smart_route** command. The **smart_route** command automatically evaluates a design and often routes it to 100% completion.

If you need to apply the same commands to multiple nets, classes, or groups of fromtos, you can use the * and ? characters as wildcards. 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 EditPlace tool interactively places and relocates components in a PCB design. EditPlace includes the following interactive capabilities:

- Interactively placing and relocating components
- Floorplanning by clustering components and defining rooms
- Setting placement rules for manufacturability
- Placement by connectivity
- Relocating placed components
- Locking components and pins
- Pushing, trading, and aligning components
- Density analysis
- Generating reports
- Undo and Redo

The EditPlace tools also supports automatic placement . The automatic placement capabilities include

- Grouping components by connectivity or net connections
- Automatic double-sided, large and small, initial component placement
- Placing decoupling capacitors in patterns
- Automatically interchanging and rotating components to reduce manhattan lengths
- Interactively or automatically swapping gate and pin connections to reduce manhattan lengths
- Automatic placement by room

Interactive Editing and Routing

The EditRoute tool interactively routes and edits wiring in a PCB design. EditRoute includes the following capabilities.

- Interactive routing and rerouting with push and shove, including blind and buried via support
- Pushing and shoving wires and vias
- Grid or gridless routing
- Design rule checking control
- 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 tool also supports a fast-circuit rules option. The EditFST option includes

- Minimum and maximum length rule checking
- Minimum and maximum length rule status display
- Automatic single net route finishing
- Add, move, remove virtual pins
- Net topology editing
- Net bundle (bus) routing
- Copper fill editing

Autorouting

The AutoRoute tool automatically routes wires and vias in a PCB design. You apply rules and control routing by using menu commands or by using a command file. AutoRoute includes the following capabilities.

- Rip-up and retry with push and shove routing for maximum completion rates
- Routing with or without grids

- Definable wire and via grids
- SMD fanout to vias or pins
- Bus routing
- Hierarchical rules-based routing
- Automatic route improvement for manufacturability
- ECO 45-degree routing
- Definable resolution
- Stub length control
- Staggered pin support
- Split power and ground plane support
- Did file command editing

The AutoRoute tool supports advanced rules, design for manufacturing, hybrid design, and fast-circuit rules options.

ADV (advanced rules)

- Signal assignments by layer
- 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 wire spreading
- 90 or 45 degree mitering
- Automatic test point generation
- Test point clearance rules

HYB (hybrid design)

- Blind and buried via control
- Vias under SMDs control
- Automatic wire bond control

FST (fast circuit rules)

- Crosstalk control by parallelism, tandem layer, and accumulated noise rules
- Timing control by minimum, maximum, and matched length or delay rules
- Automatic differential pair routing
- Rules and clearance by area
- Net shielding
- Virtual pin topology control
- Rounded corners
- Test point antenna control

Starting SPECCTRA

Chapter content

This chapter explains how to begin a SPECCTRA session. The procedures differ dependent on whether you are using a UNIX system, Windows NT, or Windows 95.

To start SPECCTRA, you can do one of the following:

- Use the Startup dialog box
- Enter the **specctra** command from a shell or command line
- Run a batch script

This chapter explains all three of the start up methods.

The Startup dialog box uses the mouse and pointer to set start up options locate files. Command line switches are useful if you know the options you want to use and the paths and filenames you want to specify. A batch script can run several SPECCTRA sessions unattended.

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

Note: Before you can run SPECCTRA on a Windows 95 or Windows NT, you must attach a software license key to a parallel port or install the FLEXlm license server.

Running SPECCTRA From the Startup Dialog Box

You can use the SPECCTRA Startup dialog box to start a session. Use the Startup dialog box when you want to browse start up options or locate files. The following procedures explain how to open the Startup dialog box.

- **To open the Startup dialog box on a UNIX system**
  - Enter specctra in a shell window

- **To open the Startup dialog box on a Windows NT or Windows 95 system**
  - Click **Start - Program - SPECCTRA 8.0 – SPECCTRA 8.0**.

In the Startup dialog box, you can either enter a filename in the filename data entry box, or click a Browse button and locate a file using the Select File dialog box (on UNIX systems) or using the Open dialog box (on Windows 95 or Windows NT systems). If you enter a filename, you must include a path name if the file is not located in the current directory.

The Startup dialog box initially displays only basic startup options. To see additional options, click More Options. The dialog box expands to display the additional Startup options.

- **To start SPECCTRA using the Startup dialog box**

  1. Enter a design file or a session file in the Design / Session data entry box.
  2. Set the startup options you want to use.
     Click the More Options button if you want to use additional startup options.
  3. Click the Start SPECCTRA button.

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

When you click a browse button, a file browser dialog box opens. You can use the mouse and pointer to choose 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, you can use any of the following methods to change directories:
   - Double-click a directory name in the Directories list, and repeat until the directory you want is displayed.
   - Enter a directory path in the File Selection data entry box.
   - Type the directory path in the Filter data entry box, and press `[Enter]`. SPECCTRA updates the filenames listed under Directories and Files.
2. Double-click a filename in the Files list, or type a filename in the File Selection data entry box.
3. Click OK.

On Windows 95 and Windows NT systems, SPECCTRA uses the standard Open dialog box for file browsing.

➢ **To choose a file on Windows 95 or Windows NT**

1. If the file you want is not in the current (design) directory, you can
   - Choose a drive in the pulldown menu.
   - Double-click a directory name, and repeat until you see the directory you want.

2. Double-click a filename, or type a filename in the data entry box and click OK.

Starting SPECCTRA Using Command Line Switches

You can use command line switches to start a SPECCTRA session. Start **specctra** from the command line to bypass the Startup dialog box. To use this method, you must know the paths and filenames for all the files you want to load, and you must know the startup switches you want to use.

The **specctra** command line syntax is

specctra [<design_or_session_file>] {[<*switch*>]}

- 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 with data from a previous session.
- A <*switch*> is one of the optional SPECCTRA command line switches. You can use switches to specify a routes, session, or wires file, a placement file, a do file, and to set other start up options.

See “Understanding Startup Options” later in this chapter for explanations of all the SPECCTRA command line switches. If a file is not located in the current directory, you must include a complete path with the filename.

If the SPECCTRA bin directory is not included in your path variable, you must include the full path name with the **specctra** command.

- A typical installation for a SPARC workstation running Solaris is located at /cds/tools.sun4v/specctra/bin. To **specctra** with this path name, enter

/cds/tools.sun4v/specctra/bin/specctra

- The default installation directory on Windows 95 and Windows NT systems is c:\cds_cct\tools\specctra\bin. To run **specctra** with this path name, enter

  c:\cds\tools\specctra\bin\specctra

➢ **To start SPECCTRA on a UNIX system**

- In a shell or command window, enter **specctra** followed by the design filename or session filename, and include any switches you want to use.

➢ **To start SPECCTRA on a Windows 95 or Windows NT system**

You can start SPECCTRA using the Command Prompt shell.

1. Open the Start menu and click Programs – Command Prompt.
2. Change to the bin directory where SPECCTRA is installed. Skip this step if the SPECCTRA bin directory is included in your path variable.
3. Enter **specctra** followed by a design filename or session filename, and include any switches you want to use.
4. Press [Enter].

You can also start SPECCTRA by using the Run dialog box.

On a Windows 95 or Windows NT system, you can

1. Click **Start > Run** to open the Run dialog box.
2. In the Open data entry box, enter **specctra** followed by the design filename or session filename, and include any switches you want to use.

   If the SPECCTRA installation directory is not in your path variable, you must include the full path name with the **specctra** command. For example

   ```
   c:\cds_cct\tools\specctra\bin\specctra design1.dsn -do cmds.do
   ```

3. Click OK.

Running SPECCTRA With a Batch Script

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 filename, a do filename, and any switches you want to use.

To avoid license failures during a batch session on a network, you can use the `-lr` switch to reserve the licenses you need for the session. See "Understanding Startup Options" later in this chapter for explanations of the SPECCTRA command line switches.

Creating a batch script

A batch script is a text file that contains a series of **specctra** commands. Each command line in the script controls a SPECCTRA session by using a do file and any switches you need. 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 that contains commands to control the session. If a file is not located in the current directory, you must specify the path to the file. The `-quit` switch ends a session after the last command in the do file. You must include -quit at the end of each command line in the script.

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
```

The following example shows a batch script for Windows 95 systems.

```
start /wait /b specctra design1.dsn -do des1.do -quit
start /wait /b specctra design2.dsn -do des2.do -quit
start /wait /b specctra design3.dsn -do des3.do -quit
start /wait /b specctra sample.dsn -do sample.do -quit
```

You can use the $/ symbols on a UNIX system (\$/ in a C shell) to specify a file in the same directory as the design file. For example:

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

Use the $\ symbols on Windows 95 and Windows NT systems. For example:

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

If the SPECCTRA installation directory is not included in your path variable, your script must include the full path to the **specctra** executable file. For example, if you installed SPECCTRA in the usual location on a SPARC workstation running Solaris:

- Enter

/cds/tools.sun4v/specctra/bin/specctra sample.dsn -do sample.do -quit

- On Windows 95 and Windows NT systems, enter

c:\cds_cct\tools\specctra\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*>

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 or command window.

➢ **To start a batch file on Windows 95 or Windows NT systems**

1. Double-click the Command Prompt icon in the Main window.
2. Change to the directory where your batch file is located.
3. Enter the batch filename.

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*>	Loads the named wires or routes file.
Placement File	-place <*file*>	Loads the named placement file. This data overrides the placement data in the design file.
Do File	-do <*file*>	Begins the session by running commands from this do file.
No Graphics	-nog	Runs SPECCTRA in non-graphics mode (without the GUI).
Quit After Do File	-quit	Exits SPECCTRA after running the last command in the do file.
No Preroutes	-nowire	Ignores prerouted wires in the design file.
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

Startup dialog box field	Switch	Use this option to...
		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 either a network license file or a node-locked password file. Use this option only to specify a file that is not in the usual location or in your search path. You can choose Default, Node, or Network.
• *Default*		• SPECCTRA uses the default search path, and looks first for a network license file.
• *Network*	-lf <*file*>	• SPECCTRA uses the network license file specified by the path and filename that you enter for <*file*>. See also the -lr switch in the following section, "Using additional command line switches."
Did File	-did	Specifies the file where

Startup dialog box field	Switch	Use this option to...
	<*file*>	SPECCTRA commands are logged during a session. The default filename is the month, day, and time with a .did extension.
Message Output File	-o <*file*>	Redirects the output transcript 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 colors and patterns for design objects and graphic features in the work area. If you do not use this switch, SPECCTRA uses color and fill pattern definitions from the colors file in your .cct directory, from the design file, from the color.std file in the current directory, or from internal defaults. See the -noinit switch in the following table for details about the color file in the .cct directory.

Using additional command line switches

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

Switch	Description
-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. This switch is available only on UNIX systems.
-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.

Switch	Description
-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. This switch is available only on UNIX systems.
-ignore_net	Causes SPECCTRA to ignore the input net names of wires read from a wires file. Instead, the wires are identified with the net names associated with the pins they connect to.
-noinit	Prevents SPECCTRA from reading the colors file and keys file in the .cct directory under your home directory. SPECCTRA reads these files by default, if they exist, before reading any do files you specify with the -do switch. When you use the -noinit switch, SPECCTRA uses color definitions from the file you specify with the -c switch, from the design file, or from internal defaults.

<table>
<tr><th>Switch</th><th>Description</th></tr>
<tr><td>-lr <feature></td><td>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 the -lr switch are

<table>
<tr><th>Feature</th><th>Description</th></tr>
<tr><td>ViewBase</td><td>Basic SPECCTRA license</td></tr>
<tr><td>RouteBase</td><td>Basic AutoRoute license</td></tr>
<tr><td>RouteADV</td><td>AutoRoute ADV license</td></tr>
<tr><td>RouteDFM</td><td>AutoRoute DFM license</td></tr>
<tr><td>RouteHYB</td><td>AutoRoute HYB license</td></tr>
<tr><td>RouteFST</td><td>AutoRoute FST license</td></tr>
<tr><td>EditBase</td><td>Basic EditRoute license</td></tr>
<tr><td>EditFST</td><td>EditRoute FST license</td></tr>
<tr><td>IPlaceBase</td><td>Basic EditPlace license</td></tr>
<tr><td>PlaceBase</td><td>AutoPlace license</td></tr>
</table>

See also the Password File field description in the previous section,</td></tr>
</table>

Switch	Description
	"Understanding SPECCTRA startup options."

Locating the .cct directory on Windows systems

The location of the .cct directory on Windows systems depends on how certain environment variables are set.

- On Windows NT systems, the .cct directory is located under the directory defined by the %homepath% and %homedrive% environment variables. For example

  HOMEDRIVE=D:
  HOMEPATH=\users\myname

  In this example, SPECCTRA reads the colors and keys files in D:\users\myname\.cct.

- On Windows 95 systems, the .cct directory is located under the Windows directory (declared in the WINDIR environment variable). For example

  WINDIR=C:\win95

  In this example, SPECCTRA reads the colors and keys files located in the C:\win95\.cct directory.

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 placement.

Using the Placement Status Report

The placement status report includes a 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 review 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

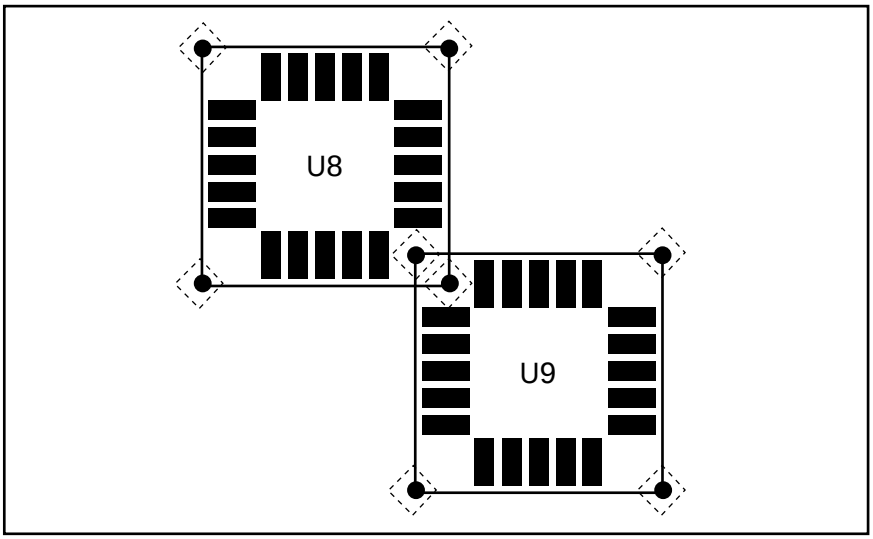
- Outside the PCB boundary.
- With less intercomponent spacing than a rule permits.
- Overlapping keepout areas.
- In the wrong orientation.
- On the wrong side of the PCB.
- In an area where a power dissipation or height rule is exceeded.
- In an area where the components are excluded.
- Outside the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

1. Click on the component in question.
2. Click **Report - Specify** to open the Report Specify dialog box, and click **Conflicts-Placement** in the Reports list.
3. Read the violations listed in the report window.
4. Click Close when you are finished with the report window.

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

Eliminating violations

Some 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 You can also use built-in reports to determine which rules are 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 automatic placement of components, 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 - Specify** to open the Report Specify dialog box, and click Rooms on the Reports list.
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 a design and analyze the results. Use the Place and Route buttons in the tool bar to switch between the place and route applications.

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 a 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 not

routed to see if they are 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. 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 design completely.

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 > Specify** to open the Report Specify dialog box, and click Place Length in the Reports list.
  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 Route application mode before you can generate a place lengths report. SPECCTRA calculates place lengths or delays for all placed components, and uses dashed ratsnest lines to show any nets or guides that violate the maximum length rules.

Meeting Advanced Technology Design Requirements

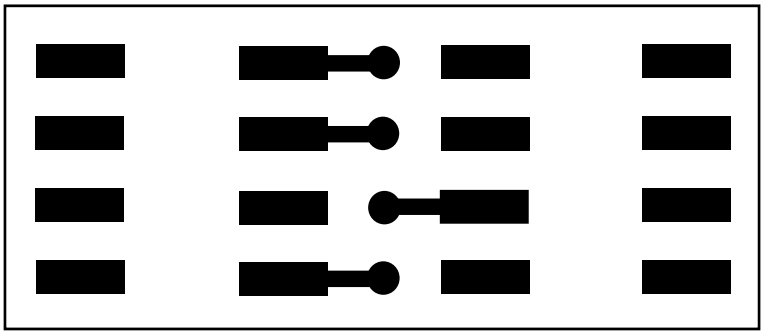
Chapter content

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.

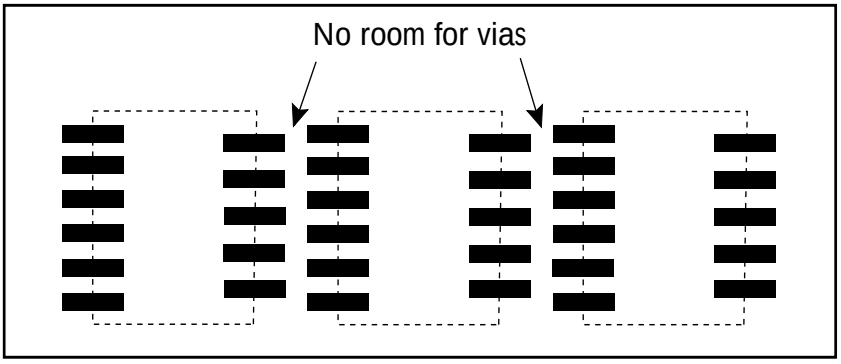
Placing Components

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

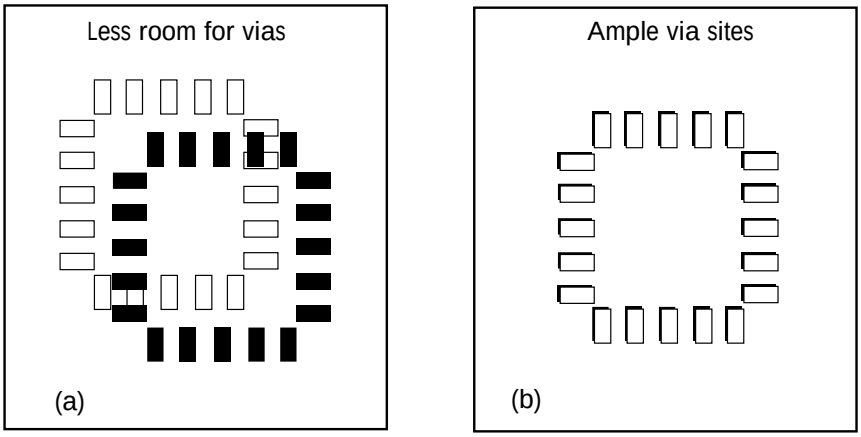
Don’t brickwall SMD components by placing components too close together. The following figure shows an example of brickwalling, where the autorouter has no chance to access the SMD pads from another layer by using vias between components.



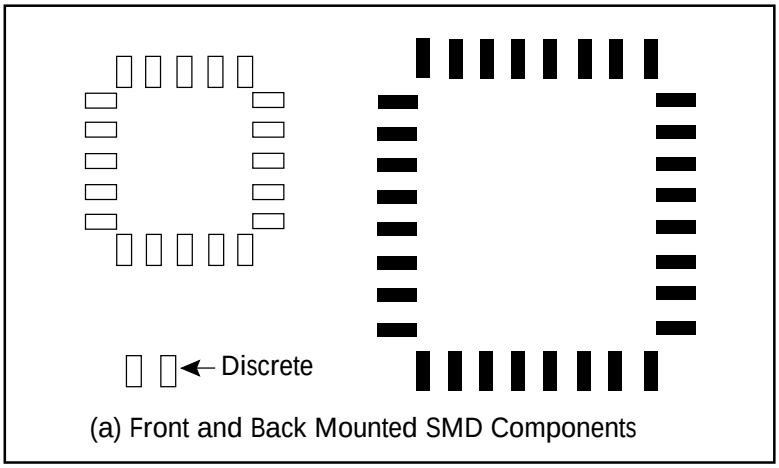
Brickwalling SMDs Leaves No Room For Vias Between Components

Mounting SMDs back-to-back

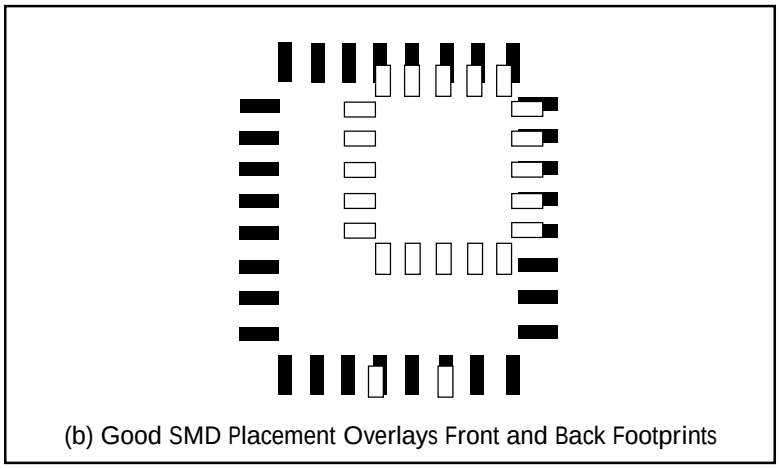
When SMD components are mounted on both sides of the PCB, the pads should be opposite each other as shown in the following figures. The goal is to create as many potential via sites as possible.



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.

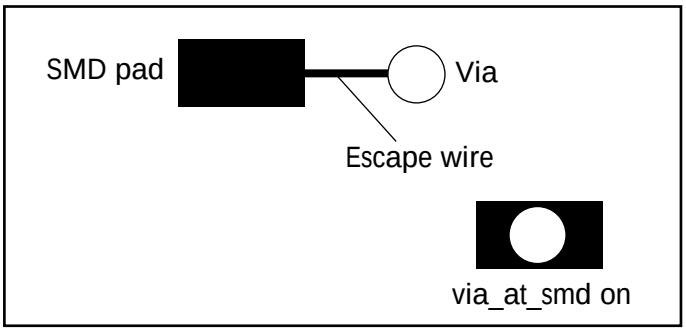
Note: You also can use the **smart_route** command, which automatically generates SMD-to-via escapes and routes your design.

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

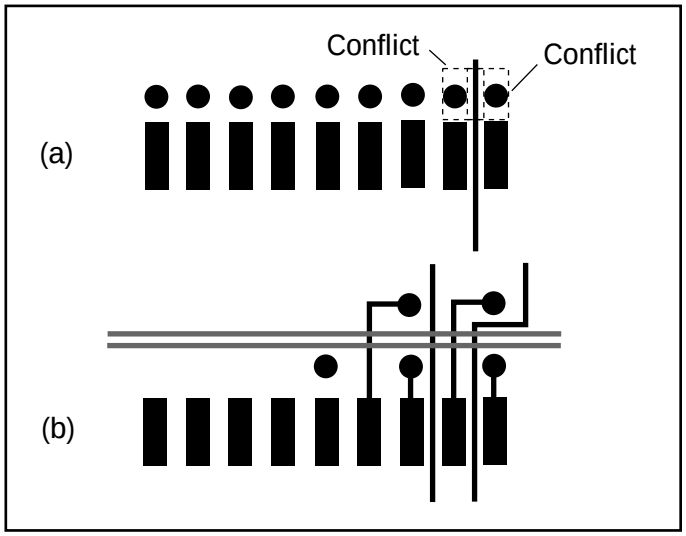
Based on your inspection of the highlighted pins, you could consider several possible improvements. Changes in the placement, via grid, via size, and clearance rules might help. If changes are made, the **fanout 5** command can be used again, followed by the **highlight no_fanout** command, to evaluate 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 first illustration shows a via barrier and the second 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.

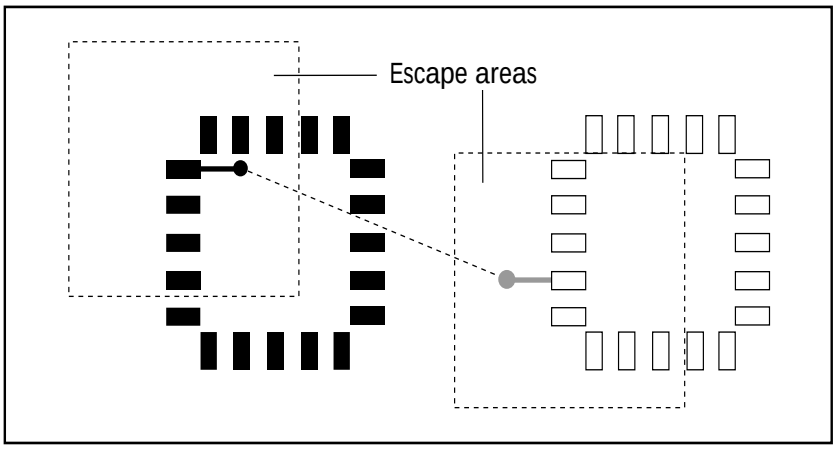
Wiring Forbidden on External Layers

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

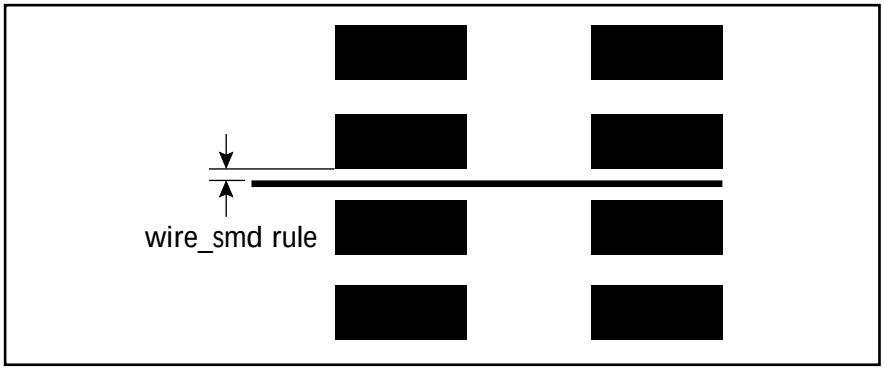
All sides of the escape area are equal in length to twice the smd_escape distance. The default smd_escape distance is 0.25 inches (0.635 cm). You can set the smd_escape distance to a different value by using the **change** command. For example, if you wanted to change the escape distance to 0.125 inches,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

```
change smd_escape 0.125
```

Optimizing Design Rules

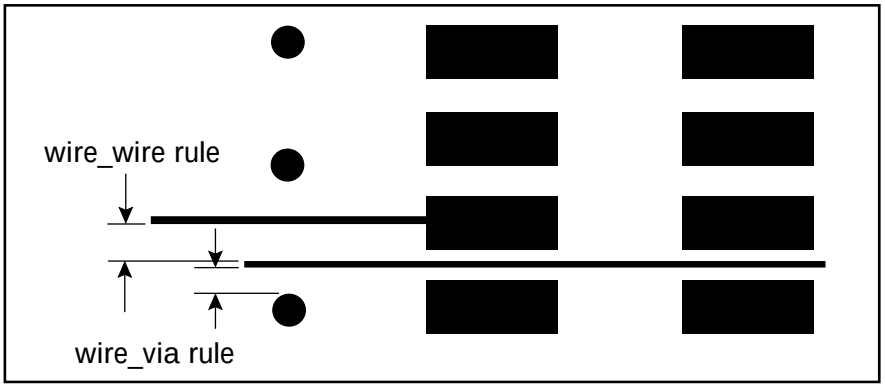
The autorouter offers more flexibility for setting rules by allowing separate control of rules such as **wire_pin** clearance, **wire_smd** clearance, and **wire_via** clearance. You can also control widths and clearances by layer, by net, and by individual connection.

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

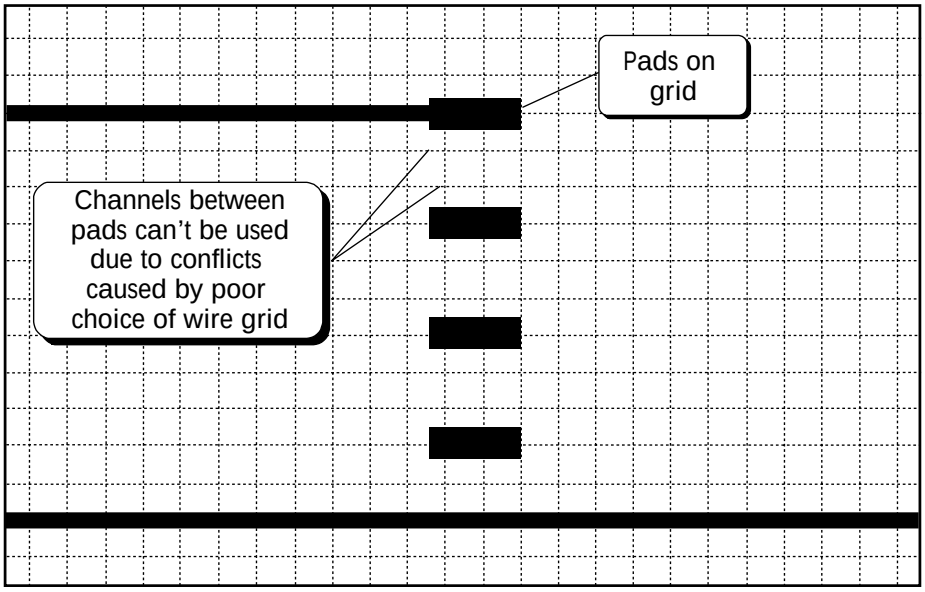
You should set the **wire_pin** and **wire_via** rules to allow two wires between adjacent vias where possible. The following figure shows a good wiring pattern that results when the **wire_via** rule is properly set.



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 route 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 shows gridding pitfalls you should avoid.



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 fast on two-layer PCBs, so the time to complete each pass is short.

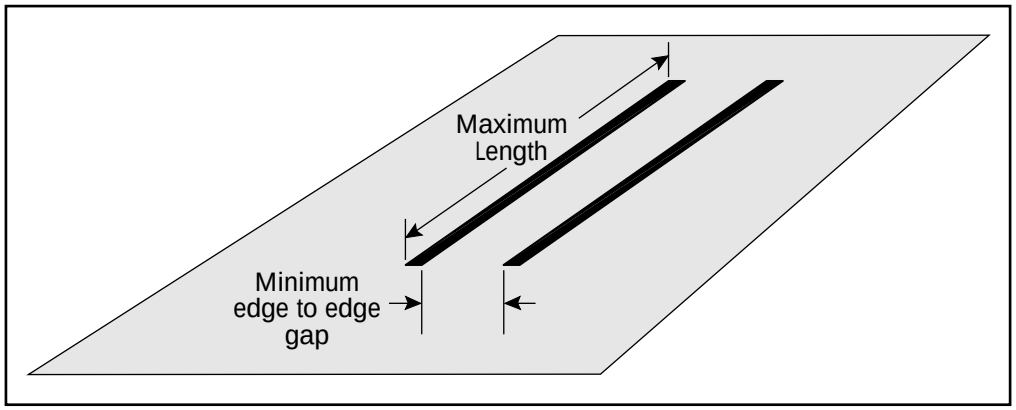
Crosstalk and Coupled Noise Concepts

SPECCTRA controls crosstalk and coupled noise with two types of rules:

- Parallel and tandem segment
- Parallel and tandem noise

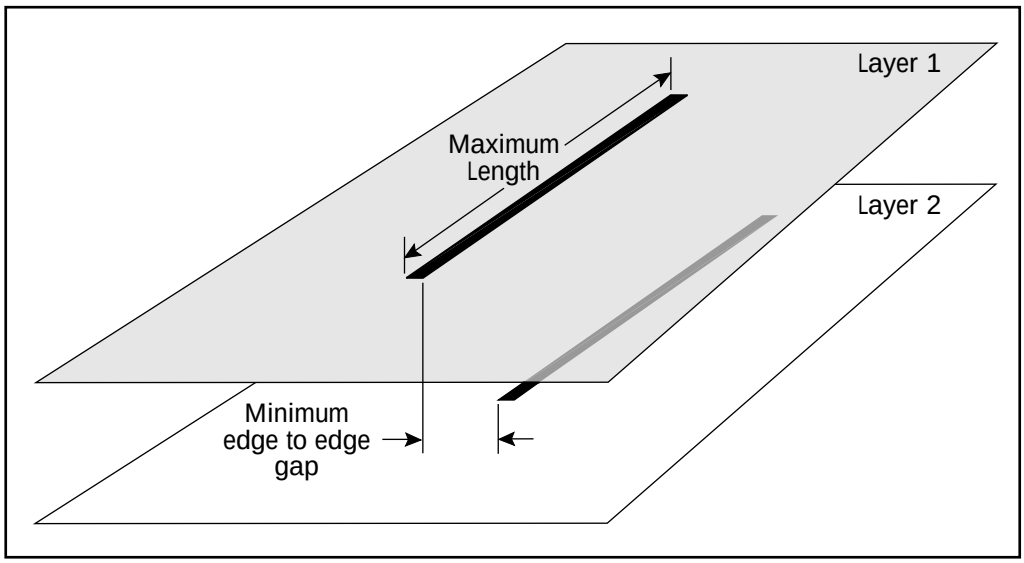
Parallel and tandem segment rules control crosstalk between the individual wire segments of nets. Parallel and tandem noise rules control the total cumulative noise that is coupled across an entire net. You can assign these rules at the layer, class, and net levels. Crosstalk at the fromto level can be controlled with parallel and tandem segment rules only.

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



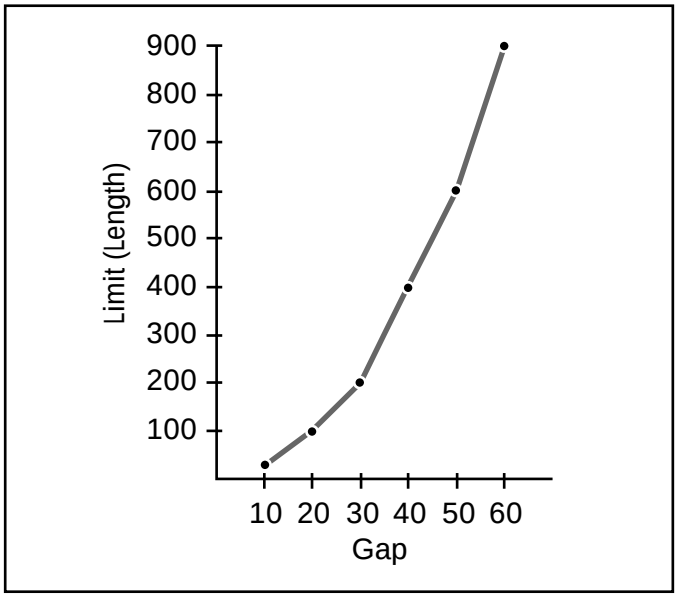
tandem_segment Crosstalk

Using Multiple Crosstalk Rules

You can set different length limits for different gaps by using multiple rules. Multiple parallel_segment and tandem_segment rules can accomodate how crosstalk varies as a function of parallel length and gap. Parallel rules do not overwrite or replace previous rules but apply collectively in an autorouting session.

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.

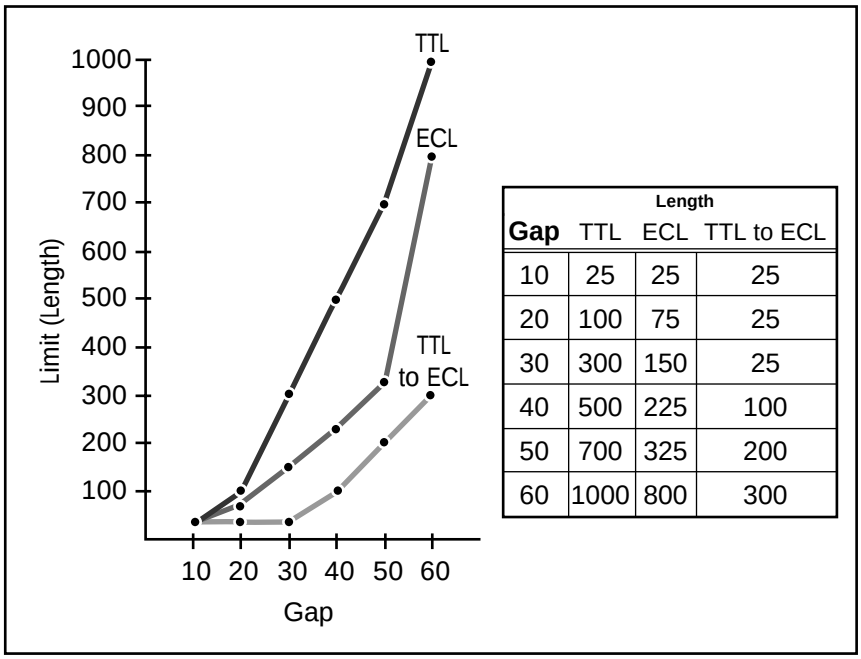
```
rule pcb (parallel_segment (limit 25) (gap 10))
rule pcb (parallel_segment (limit 100) (gap 20))
rule pcb (parallel_segment (limit 200) (gap 30))
rule pcb (parallel_segment (limit 400) (gap 40))
rule pcb (parallel_segment (limit 600) (gap 50))
rule pcb (parallel_segment (limit 900) (gap 60))
```



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 gap 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:

$$NoiseR = \Sigma (L * weight(gap) * layer\_factor)$$

where:

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

L = the measured wire length over which coupling occurs

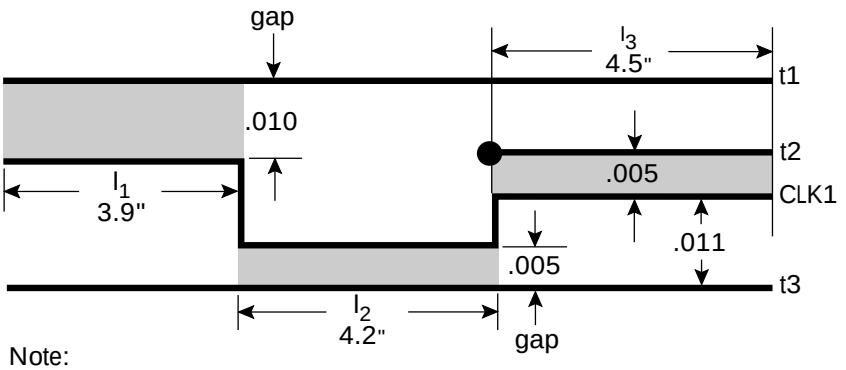
weight(gap) = the factor proportional to noise generated by the transmitting net per unit length, at a specified gap

layer_factor = the inter or intra-layer value that depends 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 cumulative 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:
$Noise_R = (l_1 * 25) + (l_2 * 83) + l_3 * 83)$
$= (3.9 * 25) + (4.2 * 83) + 4.5 * 83)$
$= 819.6$ millivolts

Coupled Noise Computation

Setting 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 and Windows systems you can change the colors and fonts.

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 in the design file or in the the SPECCTRA color map file. See chapter 2, “Starting SPECCTRA” for information about changing the colors of design objects.

Changing default colors on UNIX systems

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

The colors of the SPECCTRA window can be set by using either color names or color numbers. The color names you can use are usually listed in /usr/lib/X11/rgb.txt on your X server or /usr/openwin/lib/X11/rgb.txt for SPARC and Solaris platforms. 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 (#).

Examples of hexadecimal color entries are shown in the following table.

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.

The default values for the color resources you can set for SPECCTRA are listed in the following table.

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

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

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

Close Button	fontList:	-*-120-*=ascii

Changing default colors on Windows 95 and Windows NT systems

Use the Control Panel to change default colors on a Windows 95 or Windows NT system.

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 is usually located in your home directory.

- **To increase the default font size used by SPECCTRA**

1. Open the .Xdefaults file in your text editor.
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 95 and Windows NT systems

SPECCTRA for Windows 95 and Windows NT uses the default fonts 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 480, 800 by 600, and 1024 by 760 or higher.

Font group	Where it's used in SPECCTRA	Font Used (Face Name)	Height	Width	Weight	Display Resolution
MonoFont	Output windows	Courier	15	0	400	(1024 x 768)
		Courier	13	0	400	
	Report windows	Courier	11	0	400	(800 x 600)
	List box					(640 x 480)
MonoBold Font	Text field	Courier	15	0	600	(1024 x 768)
	Command entry area	Courier	13	0	600	
		Courier	12	0	500	(800 x 600)
						(640 x 480)
SmallText Font	Ghost text	Courier	11	7	400	(1024 x 768)
		Courier	9	7	400	
		Courier	9	7	400	(800 x 600)
						(640 x 480)
DefaultFont	All others (small)	MS Sans Serif	-11	0	400	not applicable
			-13	0	400	
	All others (large)	MS Sans Serif				not applicable

Note: The DefaultFont size depends on the font size (small or large) set in the Display Properties dialog box that you access from the Control Panel.

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. The match 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 font weight are listed in the following table.

Value	Description
0	Don't care
400	Normal/Regular
700	Bold

Changing default fonts

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

The specctra.ini file can define the fonts used for

- Output windows, report windows, and list boxes (MonoFont)
- Text fields and 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 = 400
; Resets Bold Font face name
MonoFontFaceName = Lucida Console
; Resets Small Text  Font face name and weight
SmallTextFontFaceName = Lucida Console
SmallTextFontWeight = 400
```