Master Designer

Version 8.5

Autorouter
Volume 2

October 1995

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means--electronic, mechanical, photocopying, recording, or otherwise--without the prior written permission of ACCEL Technologies, Inc.

ACCEL Technologies, Inc. provides this manual "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. ACCEL Technologies, Inc. may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time and without notice.

Although ACCEL Technologies, Inc. has gone to great effort to verify the integrity of the information herein, this publication could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes will be incorporated in new editions of this publication.

Copyright © 1994-95 Altium, under exclusive rights to ACCEL Technologies, Inc. Dervative Copyright © 1995, ACCEL Technologies, Inc.

All trademarks and registered trademarks are properties of their respective companies.

October 1995

Contents

Contents 3
Defining Wiring Rules *8* 7
Wiring Rule Definitions Screen 7
Defining or Changing the Pad-to-Pad Clearance 8
To define or change the pad-to-pad clearance 8
Adding a Wiring Rule 8
To add a wiring rule 8
Deleting a Wiring Rule 9
To delete a wiring rule 9
Modifying a Wiring Rule 9
To modify a wiring rule 9
Exiting the Wiring Rule Definitions Screen 9
To return to the previous screen (excluding the Autorouter openingscreen) 9
To continue to the Net Class Definitions screen 9
To return to the Autorouter opening screen 10
Examples of Using Wiring Rules 10
Defining Wiring Rule Clearances 10
Using the Trace Width to Name a Wiring Rule 11
To use the trace width to name a wiring rule 11
Exiting the Wiring Rule Definitions Screen 11
To return to the previous screen (excluding the Autorouter opening screen) 11
To continue to the Net Class Definitions screen 12
To return to the Pad Descriptions screen 12
Defining Net Classes *9* 13
Defining the Net Class Name 15
Defining the Wiring Rule 15
Defining the Priority 15
Defining the Layers 16
Allowing *Autorouter* to Alter the Prerouted traces 16
Defining the Type of Class 17
Working With the Type of Class Cycle Box 17
Adding a Net Class Definition 17
To add a net class 17
Defining a Net Class by Net List 18
To add net names to the Nets list box 18
Deleting a Net Name From the Net List 18
To delete a net name from the Nets list box 18
Defining a Net Class by Reference Designator 18
To add reference designators to the Designators list box 19
Deleting a Reference Designator From the Designators List 19
To delete a reference designator from the Designators list box 19
Defining a Net Class by Component 19

To add components to the Components list box..... 20
Deleting a Component Name From the Components List 21
To delete a net name from the Components list box 21
Defining a Net Class by Window 21
To define the net class by window 21
Deleting a Net Class Definition..... 22
To delete a net class definition from the Nets list box..... 22
Exiting the Net Class Definitions Screen..... 22
To return to the previous screen (excluding the Autorouter opening screen)..... 22
To continue to the Layer Descriptions screen 22
To return to the Autorouter opening screen..... 22
Examples of Routing Using Net Class Definitions..... 23
Routing Wide Power and Ground Traces..... 23
To route power and ground nets with a trace width wider than other nets 23
Ignoring Power and Ground Nets..... 23
To route all nets except the power and ground nets and have no traces on the internal power and ground planes.. 23
Routing Congested Areas by Reference Designator 24
To route congested areas by reference designator 24
Routing Only Clock Nets..... 24
To route only clock nets..... 24
Routing Only the Noncritical Parts of Nets 25
To route only the noncritical parts of nets..... 25
Exiting the Net Class Definitions Screen 25
To return to the previous screen (excluding the Autorouter opening screen)..... 25
To continue to the Layer Descriptions screen 25
To return to the Autorouter opening screen..... 25
Defining Layer Descriptions *10* 27
Defining the Number of Layers..... 28
To define the total number of layers..... 28
To define the number of routing layers..... 28
Defining Via Costs 28
To define the via costs..... 29
Editing Layers..... 30
Editing the Layer Name 30
Editing the Type 31
To select a preferred direction for a layer..... 31
Editing the Costs 31
To change a cost value..... 32
Editing a Layer Description 32
To edit a layer description..... 32
Exiting the Layer Descriptions Screen..... 33
To return to the previous screen (excluding the Autorouter opening screen)..... 33
To continue to the *Net Class Definitions* screen..... 33
To return to the Autorouter opening screen..... 33
Using the Pad Builder Utility *11* 35
Defining Padstack Data..... 35

Running the Pad Builder Utility 38
To use the pad builder utility 38
Using Mouse Commands and
Check Boxes in the Route Phase *12* 39
Using This Command 40
To resume automatic routing 40
Things to Remember 40
Using This Command 41
To leave the *Route* phase 41
Things to Remember 41
Using This Command 42
To display the entire PCB 42
Things to Remember 42
Using This Command 43
To move the viewing area 43
Things to Remember 43
Using This Command 44
To regulate scoreboard updating 44
Things to Remember 44
Using This Command 45
To redraw the editing area 45
Things to Remember 45
Using This Command 46
To skip the current pass 46
Using This Command 47
To skip the current subnet 47
Things to Remember 47
Using This Command 48
To choose and scale an area to view 48
Things to Remember 48
Using This Command 49
To choose whether the the wavefront is displayed 49
Using This Command 50
To increase the magnification 50
Things to Remember 50
Using This Command 51
To decrease the magnification 51
Things to Remember 51
Using Advanced Techniques *13* 53
Connecting Electrically Equivalent Points 53
Using Network Treeing 53
Controlling the Order in Which You Route Connections 55
Using Memory Routing 56
Using the Maze Algorithm 56
Using Post-Route Processes 58
Using Advanced Routing Algorithms 59
Using the *Rip-Up* Routing Algorithm 59
Using the *Optimize* Algorithm 60
Using Trace-Hugging 60
Using Corner Penalties 60
Understanding the *Autorouter* Architecture 61

Understanding the *Autorouter* Files 62
Controlling the *Autorouter* System (ROUTE0 and PCROUTE) 63
Editing Strategy Files (ROUTE1) 64
Extracting Databases (ROUTE2) 64
Running *Autorouter* (ROUTE3) 64
Creating a New Database (ROUTE4) 64
Generating Pad Descriptions (ROUTE5) 65
Running a Batch File 65
Creating Command Lines 65
Creating a Batch File 66
To run a batch file 66
Trading Off Speed and Results 66
Managing Memory and Grid Size 67
Using Re-Entrant Routing 69
Improving Stringer Handling 70
Glossary 71
Index 75

Defining Wiring Rules 8

Wiring rules specify trace widths and clearances. Every trace width on your board, including prerouted traces and those generated by *Autorouter*, must have a wiring rule. Net classes depend on wiring rules. (See Chapter 9, “Defining Net Classes.”)

One way to set up wiring rules for use by the Net Class Definitions screen is to define wiring rules by the type of signal a trace carries. For example, if you have one wiring rule for power, one for ground, and one for each signal in the design, you can specify a different width and clearance for each type of trace you design.

Wiring Rule Definitions Screen

The Wiring Rule Definitions screen provides a means of defining your pad-to-pad clearances and adding, deleting, and modifying the wiring rules. When you select the *Wiring Rules* button on the Autorouter opening screen, the Wiring Rule Definitions screen appears, as shown in Figure 8-1.

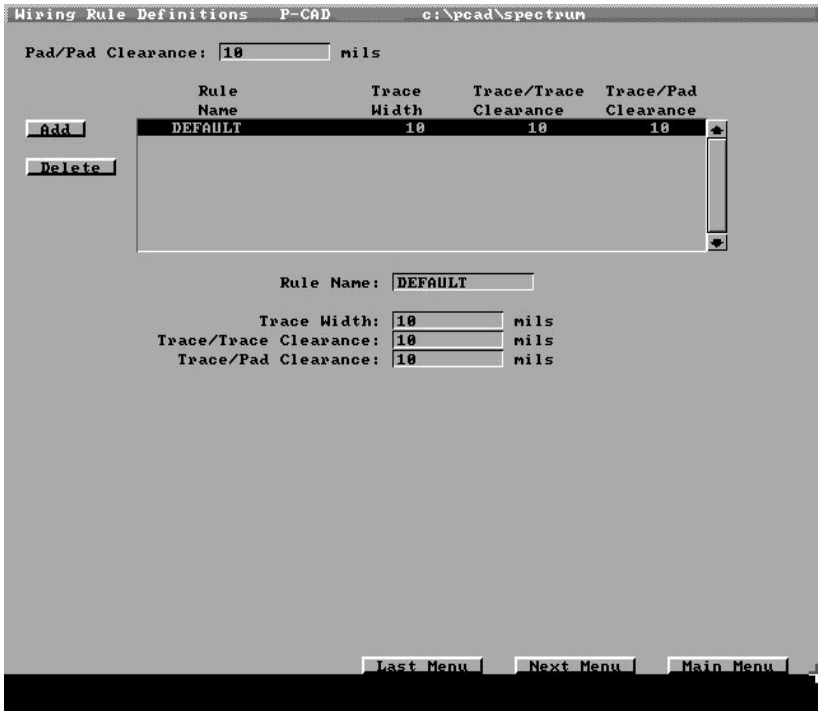

Figure 8-1. Wiring Rule Definitions screen

The Wiring Rule Definitions screen includes the following data entry boxes to define clearances, widths, and rules:

- The Pad/Pad Clearance data entry box specifies how close a via can be to another via, pin, or barrier. One pad-to-pad clearance is defined for all wiring rules in the routing strategy.
- The Rule Name data entry box is the name of the wiring rule associated with the net classes. It's best to assign *Default* as the rule name for predominant trace widths, although this assignment isn'trequired.
- The Trace Width data entry box is the width of the trace in mils or millimeters.
- The Trace/Trace Clearance data entry box specifies how close a trace can be to another trace.
- The Trace/Pad Clearance data entry box specifies how close a trace can be to vias, pins, and barriers.

Defining or Changing the Pad-to-Pad Clearance

To define or change the pad-to-pad clearance

1. Select the Pad/Pad Clearance data entry box.
2. Enter the value for the pad-to-pad clearance.

Adding a Wiring Rule

To add a wiring rule

1. Select *Add*.
2. Enter the name of the wiring rule in the Rule Name data entry box. The name is added to the list and the cursor moves to the Trace Width data entry box.
3. Enter the value for the trace width. The cursor moves to the Trace/Trace Clearance data entry box.
4. Enter the value for the trace-to-trace clearance. The cursor moves to the Trace/Pad Clearance data entry box.

5. Enter the value for the trace-to-pad clearance.

Deleting a Wiring Rule

To delete a wiring rule

1. In the list box, highlight the wiring rule you want to delete.
2. Select *Delete*.

Note: You can't delete the default wiring rule or a wiring rule that's used in a net class description.

Modifying a Wiring Rule

To modify a wiring rule

1. In the list box, highlight the wiring rule you want to modify.
2. Enter the new name in the Rule Name data entry box. If you're only changing the name, you're through with this procedure. If you want to change the values associated with the wiring rule name, go to step 3.
3. Enter the value of the trace width. The cursor moves to the Trace/Trace Clearance data entrybox.
4. Enter the value for the trace-to-trace clearance. The cursor moves to the Trace/Pad Clearance data entry box.
5. Enter the value for the trace-to-pad clearance.

Exiting the Wiring Rule Definitions Screen

After you finish defining the wiring rules, you can exit to one of threescreens.

To return to the previous screen (excluding the Autorouter openingscreen)

- Select *Last Menu*.

To continue to the Net Class Definitions screen

- Select *Next Menu*.

To return to the Autorouter opening screen

- Select *Main Menu.*

Examples of Using Wiring Rules

This section contains examples of defining wiring rule clearances and using the trace width to name a wiring rule.

Defining Wiring Rule Clearances

Figure 8-2 shows an example of wiring rule clearances.

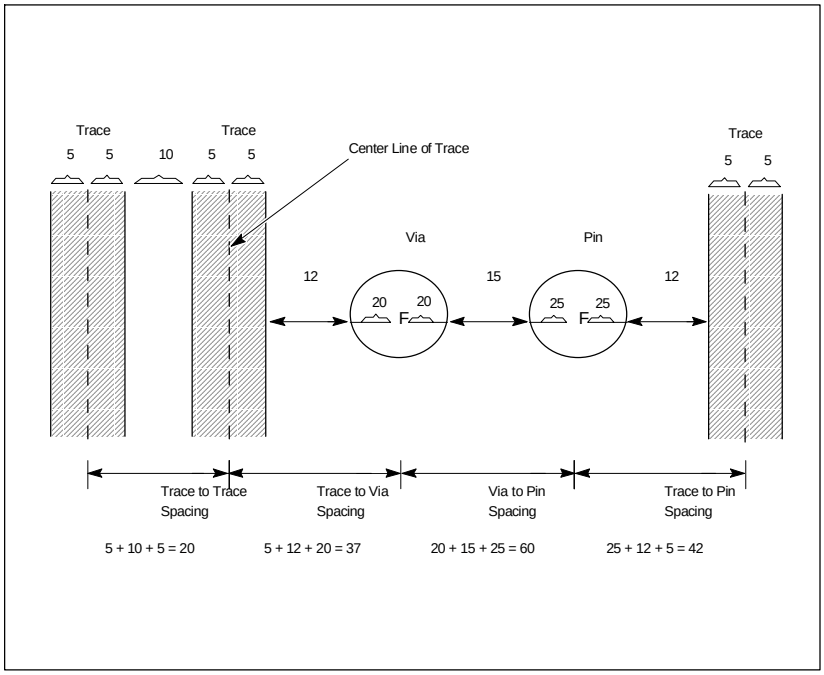

Figure 8-2. Wiring Rules Clearance Example

The sample clearance values are

- trace width = 10
- via diameter = 40
- pin diameter = 50
- trace-to-trace clearance = 10

- trace-to-pad clearance = 12
- pad-to-pad clearance = 15

The minimum spacing requirements for this example are shown at the bottom of Figure 8-2.

The center line of a 10-mil trace created by *Autorouter* must be at least 20mils from the center line of another 10-mil trace, at least 37 mils from the center of a 40-mil via, and at least 42 mils from the center of a 50-mil pinpad.

The center of a 40-mil via created by *Autorouter* must be at least 55 mils from the center of another 40-mil via and at least 60 mils from the center of a 50-mil pin pad.

Using the Trace Width to Name a Wiring Rule

This example uses the data from Figure 8-2.

To use the trace width to name a wiring rule

1. Enter **15** in the Pad/Pad Clearance data entry box.
2. Select *Add*.
3. Enter **SIGNAL10** in the Rule Name data entry box.
4. Enter **10** in the Trace Width data entry box.
5. Enter **10** in the Trace/Trace Clearance data entry box.
6. Enter **12** in the Trace/Pad Clearance data entry box.

For all priority settings except *Ignore*, all nets you assign to the SIGNAL10 wiring rule on the Net Class Definitions screen are routed with a 10-mil trace, a 10-mil trace-to-trace clearance, and a 12-mil trace-to-pad clearance.

Exiting the Wiring Rule Definitions Screen

After you modify the wiring rules, you can exit to one of three screens.

To return to the previous screen (excluding the Autorouter opening screen)

- Select *Last Menu*.

To continue to the Net Class Definitions screen

- Select *Next Menu*.

To return to the Pad Descriptions screen

- Select *Main Menu*.

Defining Net Classes 9

You may not need to define net classes. Skip this chapter if you're a new user.

The Net Class Definitions screen is used to define special handling for a group of nets. On the Net Class Definitions screen you can

- define new net classes
- assign trace widths to nets
- specify nets that aren't to be routed
- specify net routing priorities
- restrict nets to specific layers
- make specific prerouted traces unalterable

This chapter explains how to specify and define net classes and includes examples of how to use net classes. *Autorouter* assigns a default name to each net class. You can then modify each net class by editing

- Name
- Wiring Rule
- Priority
- Layers
- Prerouted Traces
- Type of Class

When you select the *Net Class Definitions* button on the Autorouter opening screen, the Net Class Definitions screen is displayed, as shown in Figure 9-1.

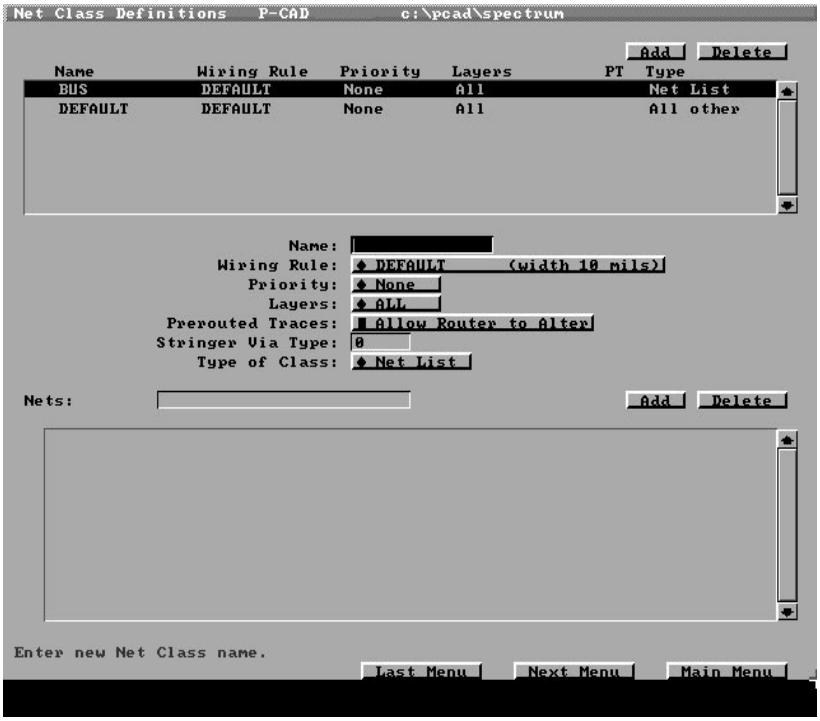

Figure 9-1. Net Class Definitions Screen With Nets List Box

The list box on the upper half of the Net Class Definitions screen displays a list of all defined net classes. The list order is important. If a net belongs to more than one class, *Autorouter* assigns the net to the first net class on the list to which the net belongs. If the net is assigned to more than one net class, *Autorouter* prints a warning in the extract section of the report file.

The last net class on the list is always *DEFAULT* and contains all nets not assigned to any other net class.

Pay particular attention to the values in the *DEFAULT* net class because most of the nets are routed according to the values in this class. For example, if you restrict the *DEFAULT* class to the COMP and INT1 layers, most traces are routed on those layers only.

The Net Class Definitions screen contains the following areas:

- Name data entry box
- Wiring Rule cycle box
- Priority cycle box
- Layers cycle box

- Prerouted Traces check box
- Type of Class cycle box

Defining the Net Class Name

The Name data entry box displays the net class name highlighted in the upper list box.

Defining the Wiring Rule

The Wiring Rule cycle box displays the wiring rule names defined in the Wiring Rules screen. The cycle box also defines the trace widths and clearances that *Autorouter* uses when it routes nets in the current class. (See Chapter 8, "Defining Wiring Rules.") Prerouted traces contained in these nets aren't restricted to the wiring rule trace width, but can be any width so long as a wiring rule with the specified width is defined.

Note: The wiring rule associated with a net class specifies the wire width only for wires that *Autorouter* creates.

If you have a trace width in the database for which no wiring rule exists, the following occurs:

1. During the *Extract* phase, a fatal error message is generated in the report file.
2. *Autorouter* doesn't continue.
3. You exit *Autorouter*.

Defining the Priority

The Priority cycle box assigns the order in which nets are routed. The choices are

- Low
- Medium
- High
- Ignore
- None

Generally, the program processes high-priority nets before medium-priority nets, medium-priority nets before low-priority nets, and low-

priority nets before no-priority nets. However, there are exceptions. Nets processed by the memory router are always routed first. (See the Performing the Memory Route section in Chapter 4, "Defining Routing Parameters.") Wider traces are always routed before narrower traces. If you specify *Ignore*, *Autorouter* doesn't add or modify traces in the nets assigned to the net class.

Defining the Layers

The Layers cycle box restricts routing of nets in a specific net class to certain layers. The choices are

- All
- COMP to COMP
- INT1 to COMP
- INT2 to COMP
- INT3 to COMP
- INT 4 to COMP
- SOLDER to COMP
- the range of layers specified on the Routing Parameters screen

Allowing *Autorouter* to Alter the Prerouted traces

The Prerouted Traces check box is used to protect critical prerouted traces belonging to specific net classes. The choices are

- A filled check box allows prerouted traces in the specified net class to be processed exactly like traces generated by *Autorouter*. The traces are cleaned up. Antennas, loops, and redundant lines are removed. Traces can be moved, optimized, beveled, via minimized, and can have acute angles removed from them.
- An empty check box indicates that prerouted traces in this net class are unalterable, as if they had an *Ignore* priority. However, unlike nets with an *Ignore* priority, any remaining unconnected subnets in the net are completed and are alterable.

Defining the Type of Class

The Type of Class cycle box specifies how the nets belonging to each net class are organized. You can choose from the following:

- *Net List* is a simple list of net names you specify.
- *Desg List* is a list of component reference designators (examples: U1,C25).
- *Comp List* is a list of component device types (examples: 7400, 7408).
- *Window* restricts net classes to an area of nets with pins enclosed by a window.

Note: The net class definitions list scrolls to accept any number of names. You can use the wildcard characters * and ? to select names, as described in the section on using a file selector window in the *Getting Started* manual.

Working With the Type of Class Cycle Box

Selecting the Type of Class cycle box allows you to add or define a net class by net list, reference designator, component, or window. Selecting the Type of Class cycle box also allows you to delete a net name from the net list, a reference designator from the designators list, or a component name from the component list.

Adding a Net Class Definition

To add a net class

1. Select *Add*. The cursor moves to the Name data entry box.
2. Enter the net class name.
3. Select a wiring rule that you defined on the Wiring Rules screen.
4. Cycle the Priority box to *None*, *Low*, *Medium*, *High* or *Ignore*.
5. Cycle the Layers box to *All* or to the appropriate range of layers.
6. Fill or empty the Prerouted Traces check box.
7. Cycle the Type of Class box to *Net List*, *Desg List*, *Comp List*, or *Window*.
8. For each entry in the Type of Class box, enter net names in the Nets list box.

Defining a Net Class by Net List

You can use the Nets list box to define a net class. See Figure 9-1.

To add net names to the Nets list box

1. Select *Net List* in the Type of Class data entry box. The Nets list box is displayed and highlighted.
2. Select the *Add* button next to the Nets list box. The cursor moves to the Nets data entry box.
3. Enter a net name.
4. Repeat step 3 until you complete the list.

Deleting a Net Name From the Net List

To delete a net name from the Nets list box

1. In the Nets list box, highlight the net name you want to delete.
2. Select the *Delete* button next to the Nets list box.

Defining a Net Class by Reference Designator

You can use the Designators list box to define a net class. See Figure 9-2.

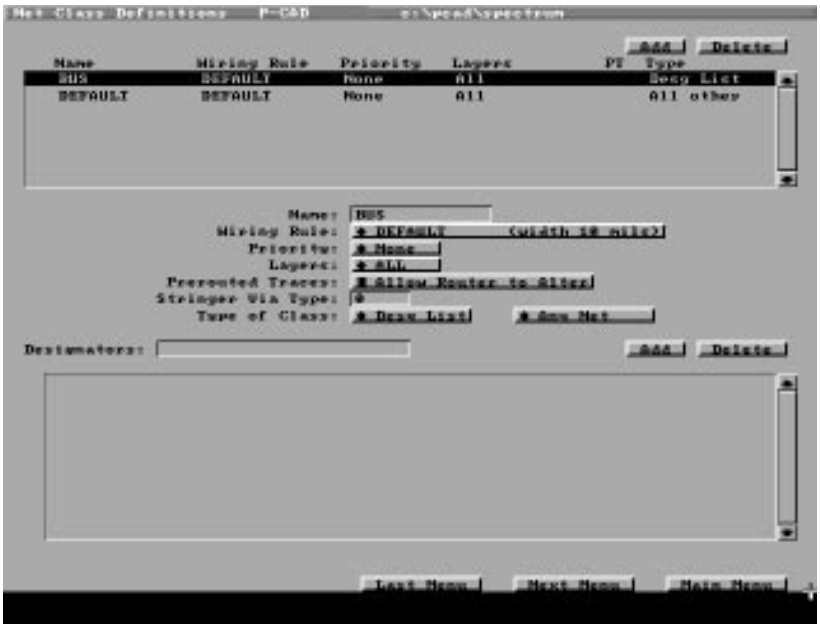

Figure 9-2. Net Class Definitions Screen With Designators List Box

To add reference designators to the Designators list box

1. Select *Desg List* in the Type of Class data entry box. The Designators list box is displayed.

Set the cycle box to the right of the Type of Class cycle box to one of the following:

- *Any Net* indicates that the net class includes all nets belonging to the specified components.
- *Common Nets* indicates that the net class includes only those nets common to all components specified in the list.

3. Select the *Add* button next to the Designators list box. The Designators data entry box is highlighted.
4. Enter the reference designator you want included in the net class.
5. Repeat steps 3 and 4 until you complete the list.

Deleting a Reference Designator From the Designators List

To delete a reference designator from the Designators list box

1. In the Designators list box, highlight the reference designator you want to delete.
2. Select the *Delete* button next to the Designators list box.

Defining a Net Class by Component

You can use the Components list box to define a net class. See Figure 9-3.

Figure 9-3. Net Class Definitions Screen With Components List Box

To add components to the Components list box

1. Select *Comp List* in the Type of Class data entry box. The Components list box is displayed.
2. Set the cycle box to the right of the Type of Class cycle box to one of the following:

- *Any Net* indicates that the net class includes all nets belonging to the specified component types.
- *Common Nets* indicates that the net class includes only those nets common to all components in the list.

3. Select the *Add* button next to the Components list box. The Components data entry box is highlighted.
4. Enter a component name (such as 74LS04, 7408).
5. Repeat steps 3 and 4 until you complete the list.

Deleting a Component Name From the Components List

To delete a net name from the Components list box

1. In the Components list box, highlight the net name you want to delete.

2. Select the *Delete* button next to the Components list box.

Defining a Net Class by Window

If you selected *Window* as the class type, the Window Coordinates data entry boxes are displayed, as shown in Figure 9-4.

Figure 9-4. Net Class Definitions Screen With Coordinates Data Entry Boxes

To define the net class by window

1. Set the cycle box to the right of the Type of Class cycle box to one of the following:

· *One Pin Inside* indicates the net class includes all nets with at least one pin inside the window.

- *All Pins Inside* indicates the net class includes only nets with all pins inside the window.

  The data entry boxes that define the window's lower left and upper right corners are

- llx (lower left X coordinate)
- lly (lower left Y coordinate)
- urx (upper right X coordinate)
- ury (upper right Y coordinate)

2. Select the llx data entry box.
3. Enter the llx value. The cursor automatically moves to the next data entry box.
4. Enter the value in each of the other three data entry boxes.

Deleting a Net Class Definition

To delete a net class definition from the Nets list box

1. In the upper list box, highlight the net class you want to delete.
2. Select *Delete* above the upper list box.

Exiting the Net Class Definitions Screen

After you finish defining the net classes, you can exit to the previous screen you used (excluding the Autorouter opening screen), the Layer Descriptions screen, or the Autorouter opening screen.

To return to the previous screen (excluding the Autorouter opening screen)

- Select *Last Menu*.

To continue to the Layer Descriptions screen

- Select *Next Menu*.

To return to the Autorouter opening screen

- Select *Main Menu*.

Examples of Routing Using Net Class Definitions

Following are five examples of routing using net class definitions.

Note: Since you can use several techniques to route using net classes, the five examples are general procedures rather than the step-by-step procedures found in the rest of the manual.

Routing Wide Power and Ground Traces

To route power and ground nets with a trace width wider than other nets

1. Use the Wiring Rule Definitions screen to create a wiring rule specifying the width and clearances you want.
2. Use the Net Class Definitions screen to add a net class to the upper list box and name it *PWR*.
3. Cycle to the wiring rule you created in step 1.
4. Cycle to *Net List* in the Type of Class cycle box.
5. In the Nets list box, list the power and ground nets that use the wiring rule you defined in step 1.

Ignoring Power and Ground Nets

You can ignore power and ground nets because you want to put them on internal power and ground planes.

To route all nets except the power and ground nets and have no traces on the internal power and ground planes

1. Use the Routing Parameters screen to specify the number of layers as the total number of signal layers on the board minus 2; don't include the power and ground planes.
2. Use the Wiring Rules screen to create a wiring rule specifying the width and clearances you want. Don't use zero as a width or clearance.
3. Add a net class to the upper list box and name it *PWR*.
4. Cycle to the wiring rule you created in step 2.
5. Cycle the Priority box to *Ignore*.

6. Cycle the Layers box to *All*.

7. Cycle to *Net List* in the Type of Class cycle box.

8. In the Nets list box, list the power and ground nets that use the wiring rule you defined in step 1.

Note: Make sure you assign the proper pin types to the pins that go to the inner plane.

Routing Congested Areas by Reference Designator

Areas with several pin grid arrays or several connectors are often congested. Routing these areas first increases the completion percentage.

To route congested areas by reference designator

1. Use the Wiring Rules screen to create a wiring rule specifying the width and clearances you want.

   or

   Use the default wiring rule.

2. Assign a high priority to the net class definition you choose.

3. Cycle to *Desg List* in the Type of Class cycle box.

4. Use the Designators list box to list your connectors and pin grid arrays by reference designator.

Note: If all your connector part names start with the letters conn you can use a component list; type net class and enter the pattern conn* to include all the connectors.

Routing Only Clock Nets

To route only clock nets

1. Use the Wiring Rules screen to create a wiring rule specifying the width and clearances you want.

2. Add a net class to the upper list box and name it *CLK*.

3. Cycle to the wiring rule you created in step 1.

4. Cycle to *Net List* in the Type of Class cycle box.

5. Cycle the priority in the *DEFAULT* net class to *Ignore* so that the program ignores all but the clock nets.

6. In the Nets list box, list the clock nets.

Routing Only the Noncritical Parts of Nets

To route only the noncritical parts of nets

1. Manually route critical parts of certain nets (for example, clock nets).

2. For the *DEFAULT* net class, empty the Prerouted Traces check box. The program then routes all nets but leaves the critical prerouted traces alone.

Exiting the Net Class Definitions Screen

After you define net classes, you can exit to one of three screens.

To return to the previous screen (excluding the Autorouter opening screen)

· Select *Last Menu*.

To continue to the Layer Descriptions screen

· Select *Next Menu*.

To return to the Autorouter opening screen

· Select *Main Menu*.

Defining Layer Descriptions 10

The Layer Descriptions screen has several purposes. You can specify a number of total layers different from the number of routing layers, specify a preferred direction for each routing layer, and make detailed adjustments to the cost values used by the maze routing algorithm.

For most applications, you can use the default values *Autorouter* supplies. You can skip this chapter if you're a new user.

You describe layers to the *Autorouter* by defining

- total layers
- routing layers
- through-hole and interstitial via costs
- routing directions
- trace length costs

When you select the *Layer Descriptions* button on the Autorouter opening screen, the Layer Descriptions screen appears, as shown in Figure 10-1.

Figure 10-1. Layer Descriptions Screen

Defining the Number of Layers

The Total Layers data entry box specifies the total number of trace layers *Autorouter* recognizes. The Routing Layers data entry box specifies the number of layers on which *Autorouter* places routing traces.

Usually, the value in the Total Layers data entry box is the same as the value in the Routing Layers data entry box. However, if you want to place prerouted traces on a layer but don't want *Autorouter* to use those layers, specify a total layer count larger than the routing layer count.

On a six-layer board, for example, if you choose to restrict two of the layers to power, ground, and some high-speed nets that you'll route manually, the total layer count is six and the number of routing layers is four.

In this example, because of the two restricted layers, the pins connected by the prerouted traces aren't routed to any other layer and no vias are generated that illegally touch the prerouted traces on those layers.

To define the total number of layers

1. Select the Total Layers data entry box.
2. Enter the total number of layers you want *Autorouter* to acknowledge.

To define the number of routing layers

1. Select the Routing Layers data entry box.
2. Enter the number of routing layers you want *Autorouter* to use. If you want to leave layers for manual routing only, make the Routing Layers value smaller than the Total Layers value. The default is the same as the Total Layers value.

Defining Via Costs

Costing is a method for calculating the expense of a particular path. You assign cost values to six cost elements and *Autorouter* calculates the least expensive legal path. The six cost elements are North/South, East/West, NorthEast/SouthWest, NorthWest/SouthEast, through-hole via, and interstitial via. You can also assign different cost element values for each layer.

The two via cost data entry boxes are

- Through Via Cost
- Interstitial Via Cost

The Through Via Cost data entry box is used to make trade-offs between the number of through vias and the trace length. If the Via Type cycle on the Detailed Routing Parameters - Part 1 screen is set to *Through*, the Through Via Cost data entry box is displayed on the Layer Description screen.

The Interstitial Via Cost data entry box is used to make trade-offs between the number of interstitial vias and the trace length. If the Via Type cycle box on the Detailed Routing Parameters - Part 1 screen is set to *Interstitial*, the Interstitial Via Cost data entry box is displayed on the Layer Description screen.

If the Via Type cycle box on the Detailed Routing Parameters - Part 1 screen is set to *Mixed*, the Through Via Cost and the Interstitial Via Cost data entry boxes are displayed on the Layer Description screen.

To define the via costs

1. Select the Through Via Costs data entry box.
2. Enter the value. The range is 0 to 999.
3. Select the Interstitial Via Costs data entry box.
4. Enter the value. The range is 0 to 999.

Note: The Detailed Routing Parameters—Part 1 and Detailed Routing Parameters—Part 2 screens show the same via cost values. If you change a via cost value on one screen, the via cost value on the other screen is also changed.

The system displays a default layer list in the list box. *Autorouter* creates the list based on the number of total layers and the number of routing layers. An example of the layer list is shown in Figure 10-2.

Figure 10-2. Routing Layers Via Costs List Box

In the figure, the number of total layers is six and the number of routing layers is four, so the third and fourth internal layers are listed as nonrouting.

Editing Layers

As shown in Figure 10-2, each layer has one entry in the list box. Each entry consists of the following:

- Layer Name data entry box
- Type cycle box
- Costs data entry boxes

Editing the Layer Name

Layer Name is the layer name in the PCB database, which is the same as the layer name that was assigned in the *PCB Editor*.

Editing the Type

Type is the preferred direction of the layer. The choices are

- *North/South*
- *East/West*
- *NorthEast/SouthWest*
- *NorthWest/SouthEast*
- *No Preferred Direction*
- *Non-Routing*

If the layer is a routing layer, the type is the preferred routing direction for the specified layer. If the layer number is greater than the number of routing layers, the layers that aren't used for routing show *Non-Routing* as the type.

NorthEast/SouthWest and *NorthWest/SouthEast* are available only if you fill the Perform Diagonal Route check box on the Routing Parameters screen. See Chapter 4, "Defining Routing Parameters."

To select a preferred direction for a layer

- Cycle the Type box to the preferred direction.

Editing the Costs

A cost value indicates the cost of a trace with a length of one grid in the named direction on the specified layer. The range is 0 to 999.

Enter a zero cost value to prevent a trace from being created in a specific direction. A zero value represents an infinite cost. For instance, entering a zero for the *North/South* cost on a layer means that *Autorouter* is not to generate any vertical traces on the layer.

No Preferred Direction gives equal values for each direction on a layer.

Changing a cost value can change a layer's preferred direction since, by definition, the preferred direction is the one with the lowest cost. The higher the cost, the less likely traces in the specified direction will be routed on the specified layer.

Caution: You can adversely affect algorithm performance with incorrect cost values. For most applications, use the default costs generated by *Autorouter*.

To change a cost value

1. Select the Costs data entry box for a particular direction.
2. Enter the cost value.

Editing a Layer Description

To edit a layer description

1. In the Layer list box, highlight the layer you want to change.
2. Select the Layer Name data entry box.
3. Enter the new layer name.
4. Select the Type data entry box.
5. Cycle the Type cycle box to the option you want.
6. Select the North/South data entry box.
7. Enter the North/South value.
8. Select the East/West data entry box.
9. Enter the East/West value. If you set the Perform Diagonal Route cycle box to *Yes* on the Routing Parameters screen, the NorthEast/SouthWest and NorthWest/SouthEast data entry boxes are displayed.
10. Select the NorthEast/SouthWest data entry box.
11. Enter the value.
12. Select the NorthWest/SouthEast data entry box.
13. Enter the value.

Exiting the Layer Descriptions Screen

After you finish defining the wiring rules, you can exit to one of three screens.

To return to the previous screen (excluding the Autorouter opening screen)

- Select *Last Menu*.

To continue to the *Net Class Definitions* screen

- Select *Next Menu*.

To return to the Autorouter opening screen

- Select *Main Menu*.

Using the Pad Builder Utility 11

You start the pad builder utility only from the Autorouter opening screen.

The pad builder utility gives *Autorouter* a description of the size and shape of each pad type so *Autorouter* leaves enough clearance between pads and traces. The pad builder utility uses the padstack information in the database to automatically generate pad descriptions in the strategy file. It also uses the aperture list specified with the *PCB Editor Environment*! *Inner Plane Apertures* command to generate strategy file information indicating which pin types are connected to an inner plane. The pad builder utility also corrects the strategy file according to information it reads from the database.

The pad descriptions are for the pin types in the database. Any pad descriptions in the strategy file that aren't used in the database are retained in the strategy file. The program tries to avoid a situation where no pad descriptions exist. If the pad builder can't build a pad in the database, it uses the pad description previously defined for the pin type, as described in Chapter 7, “Defining Pad Descriptions.”

The pad builder utility processes square, rectangular, and circular pads. If you created padstacks with flash layer data associated with the Integrated Aperture Table, and have pad shapes such as ovals and polygons, the pad builder utility adds a “bounding box” to the shape to make it appear square, rectangular, or circular. Figure 11-1 shows how the Autorouter adds bounding boxes to oval and polygon apertures.

Defining Padstack Data

If you've already created padstacks for your database, the utility adds the padstack data to your strategy file. If your database doesn't have padstack data, you can define your pad information by using the Pad Description screen described in Chapter 7, “Defining Pad Descriptions.”

The Pad Builder Utility screen is shown in Figure 11-2.

The Pad Builder Utility screen contains the following:

- Design name is the directory from which you are running *Autorouter* (your current working directory).
- Input Database Filename is the PCB database from which you're building the pad descriptions.

- Input Strategy Filename is the strategy filename used in the Autorouter opening screen.

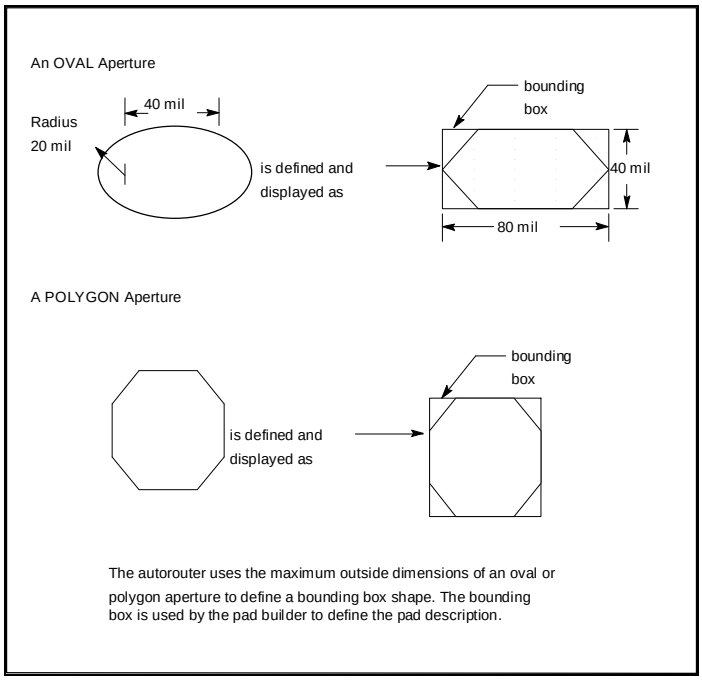

Figure 11-1. Processing Flash Data

Figure 11-2. Pad Builder Utility Screen

- Output Strategy Filename data entry box is the name of the strategy file that is to contain the pad descriptions from running the pad builder utility. You must enter this filename before you can use the Use Integrated Apertures check box, and the Layer name data entry box and list box. The default value is the input strategy filename. If you change the output strategy name, be sure to change it in the Strategy Filename data entry box when you exit back to the Autorouter opening screen.
- The Use Integrated Apertures check box specifies whether you are using the Integrated Aperture Table to describe flash data for your padstacks. Fill this check box if you have the Integrated Aperture Table loaded in your database. Empty it if you created your padstacks by drawing the pad shapes on graphic layers.
- The layer name list box defines the padstack layers to be used to generate the pad definitions. If the Use Integrated Apertures check box is filled, the program will load default layers for flashes. If the Use Integrated Apertures check box is empty, the program will load default layers for graphics. The default flash layers may not match the ones in your database. If they don't match, edit them to match the layers used in your padstacks.

Note: You can't overwrite pcad1.ctl or pcad2.ctl, the strategy files supplied with *Autorouter*. If you plan to use this file, you must save it under a new name before you can run the pad builder utility.

Also, before you use the output strategy file, use the Pad Descriptions screen to examine the automatically generated pad descriptions carefully and make any necessary changes to them, as described in Chapter 7, “Defining Pad Descriptions.”

For each routing layer, you specify which database layer has information for pad shapes, through-hole vias, and interstitial vias. An example which doesn't use integrated apertures is shown in Table 11-1. In this example, the padstacks were created by drawing the pad shapes on graphic layers.

Table 11-1. Sample Database Layers With Graphic Information

Routing Layers	*Thru Pin Layers*	*SMD Pin Layers*	*Thru-Hole Via Layers*	*Interstitial Via Layers*
COMP	PADCOM	PINTOP	PADCOM	PADCOM
INT1	PADINT	-------	PADINT	PADIN1
INT2	PADINT	-------	PADINT	PADIN2
SOLDER	PADCOM	PINBOT	PADCOM	PADCOM
INT3	PADINT	-------	PADINT	PADIN3
INT4	PADINT	-------	PADINT	PADIN4

An example which uses integrated apertures is shown in Table 11-2.

Table 11-2. Sample Database Layers With Integrated Apertures

Routing Layers	*Thru Pin Layers*	*SMD Pin Layers*	*Thru-Hole Via Layers*	*Interstitial Via Layers*
COMP	FLCOMP	PINFTP	FLCOMP	FLCOMP
INT1	FLINT	-------	FLINT	FLINT
INT2	FLINT	-------	FLINT	FLINT
INT3	FLINT	-------	FLINT	FLINT
INT4	FLINT	-------	FLINT	FLINT
SOLDER	FLSOLD	PINFBT	FLSOLD	FLSOLD

Running the Pad Builder Utility

To use the pad builder utility

1. Select Pad Builder on the Autorouter opening screen.
2. Enter the output strategy filename.

   or

   Press to accept the input strategy filename as the output strategy filename.
3. Enter the layer name.
4. Enter a value for each entry in the data entry box.

Note: If this is the first time you're using the pad builder utility for this database, asterisks appear in the Layer name data entry box and in the Layer name list box.

5. Select *RUN* to run the pad builder utility.

   or

   Select *EXIT* to return to the Autorouter opening screen.

The pad builder utility appends error messages and warnings to the report file, *filename*.rep. Review the error messages and warnings before using the output strategy file.

Note: If you use padstack graphics to draw elements with widths, the Pad Builder Utility generates information without the width data. For example, if .060 pad graphics consists of a .030 diameter circle with a width of .015 (to get a filled pad), the Pad Builder Utility generates a .030 dia pad in Pad Descriptions screen.

Using Mouse Commands and Check Boxes in the Route Phase 12

This chapter lists the mouse commands and check boxes you use in the *Route* phase. The entries are listed in alphabetical order.

Note: You can also enter commands with the keyboard. To use the keyboard, type a forward slash followed by the command. (In most cases, the first three letters of the command are sufficient to activate it.)

Continue

Restarts automatic routing after you pause Autorouter

Using This Command

To resume automatic routing

- Select *Continue*. Routing resumes.

Things to Remember

The *Continue* button disappears after you select it. If you press Esc and then don't enter another command within 30 seconds, autorouting resumes and the *Continue* button disappears.

Exit

Terminates the *Route* phase but leaves the router in a restartable state

Using This Command

To leave the *Route* phase

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Exit*. A decision box appears with the message

   EXIT ROUTER: Are you sure?

3. Select *No* if you don't want to exit, or select *Yes* if you want to exit. If you select *Yes*, the program displays

   Router was halted by user.
   Acknowledged

   Then the screen is cleared and, if *Return* is set to *Yes*, a decision box appears with the message

   Route was interrupted. Create output database?

4. Select *Yes* if you plan to use the *PCB Editor* or *Design Rules Check* to work on the incomplete results.

   or

   Select *No* to end the *Route* phase.

Things to Remember

When you resume routing, the Autorouter opening screen is set to start the router where you left off. (Route is set to *Resume*, and Return is set to the same value as before.) To resume routing, select *RUN*.

Fit View

Displays the entire PCB within the editing area

Using This Command

To display the entire PCB

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Fit View*. The entire database appears within the viewing area.

3. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter Fit View* command is not the same as the *PCB Editor Fit View* command. The *Autorouter Fit View* command uses the board outline; the *PCB Editor* uses all graphics.

Pan

Moves the current viewing area to a new location

Using This Command

To move the viewing area

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Pan*. The program prompts

   Pan: Select view center.

3. Select the center of the area you want to view. The program redraws the editing area. You can continue clicking left until the view is correct, then click right to end the command.

4. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter Pan* command is the same as the *PCB Editor Pan* command.

Quiet

Specifies how often the scoreboard displays the progress of the router

Using This Command

To regulate scoreboard updating

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. To have the scoreboard display progress less frequently, fill the Quiet check box.

   or

   To have the scoreboard display progress more frequently, empty the Quiet check box.

3. Select *Continue*. Routing resumes.

Things to Remember

An empty Quiet check box updates all the scoreboard options as they change. This is the default.

A full Quiet check box updates the ELAPSED, ROUTED SUBS, VIA COUNT, and % COMPLETED scoreboard options only when *Autorouter* reads a new file, enters the edit mode, initializes, or starts a new router pass. *Autorouter* runs faster when you fill the Quiet check box. You may want to fill the Quiet check box for unattended or overnight runs.

Redraw

Redraws the editing area; also restores graphics that are partially displayed or aren't visible

Using This Command

To redraw the editing area

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Redraw*. The program redraws the screen.

3. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter Redraw* command is the same as the *PCB Editor Redraw* command.

You can press Esc to interrupt *Redraw* if you don't need to see all the wires on the board.

Skip Pass

Terminates the current pass and continues with the next pass

Using This Command

To skip the current pass

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Skip Pass*. A decision box appears with the message

   SKIP PASS: Are you sure?

3. Select *Yes* to skip the pass, or select *No* to continue the pass. If you select *Yes*, the program prompts

   PASS IN TERMINATION

   until the next pass begins.

The program continues with the next pass.

Skip Subnet

Skips the current subnet during a *Maze, Rip-Up*, or *Optimize* pass and continues with the next subnet

Using This Command

To skip the current subnet

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Skip Subnet*. The program prompts

   SUBNET ABORTED

   on the bottom line of the scoreboard.

The program continues with the next subnet.

Things to Remember

You can only skip a subnet when a *Maze, Rip-Up*, or *Optimize* pass is running. The *Skip Pass* button is displayed only during these three passes.

View Window

Scales a window you select to fill the editing area

Using This Command

To choose and scale an area to view

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *View Window*. The program prompts

   View Window: Corner 1. . .

3. Select the first corner.

4. The program prompts

   View Window: Corner 2. . .

5. Select the second corner. The program redraws the screen with the new view.

6. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter View Window* command is the same as the *PCB Editor View Window* command.

Wave

Specifies whether the editing area displays the wavefront

Using This Command

To choose whether the the wavefront is displayed

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. To show the wavefront, fill the Wave check box.

   or

   To hide the wavefront, empty the Wave check box.

3. Select *Continue*. Routing resumes.

Zoom In

Centers the editing area around the cursor and increases the magnification of the area shown

Using This Command

To increase the magnification

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Zoom In*. The program prompts

   Zoom In: Select view center.

3. Select the center of the area to be shown. The program increases the display magnification and centers the display around the selected point. The program prompts

   Zoom In: Select view center.

4. Repeat step 3 each time you want to increase the magnification.

5. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter Zoom In* command is the same as the *PCB Editor Zoom In* command.

Zoom Out

Centers the editing area around the cursor and decreases the magnification of the area shown

Using This Command

To decrease the magnification

1. Press Esc to interrupt the router. The program prompts

   Router is now paused. . .

2. Select *Zoom Out*. The program prompts

   Zoom Out: Select view center.

3. Select the center of the area to be shown. The program decreases the display magnification and centers the display around the selected point. The program prompts

   Zoom Out: Select view center.

4. Repeat step 3 each time you want to decrease the magnification.

5. Select *Continue*. Routing resumes.

Things to Remember

The *Autorouter Zoom Out* command is the same as the *PCB Editor Zoom Out* command.

Using Advanced Techniques 13

This chapter describes some of the internal structure and methodology in *Autorouter*. This information isn't essential for you to use *Autorouter*. If you're a new user, you can skip this chapter.

This chapter tells you about

- the routing algorithm—the basic methodology used by *Autorouter*
- advanced routing algorithms—the methodology used by the *Rip-Up* and *Optimize* algorithms available on the Rip-Up Parameters screen
- *Autorouter* architecture—a system flow of the programs and files used in *Autorouter*
- running a batch file—how to run *Autorouter* from the DOS prompt and how to set up batch files
- trading off speed and results—the *Autorouter* memory management methods and how they affect your results
- re-entrant routing—a routing technique that often produces better routing results
- improving stringer handling

Connecting Electrically Equivalent Points

Autorouter attempts to connect all electrically equivalent points in a legal manner (with no shorts, loops, or design rules violations) by using the following techniques:

- network treeing
- ordering
- memory routing
- maze routing
- post-route processing

Using Network Treeing

Autorouter breaks up each net into subnets. Each subnet consists of either two pins connected by a ratsnest or a netlist definition created in the *PCB Editor*. *Autorouter* attempts to connect one subnet at a time.

A network with more than two pins is therefore separated into several pairs of pins, with each pair denoted as a subnet. The Route Type option described in the Selecting the Route Type section in Chapter 4, "Defining Routing Parameters," controls how the subnets are generated.

Daisy-chain, Minspan, and Steiner treeing are shown in Figure 13-1.

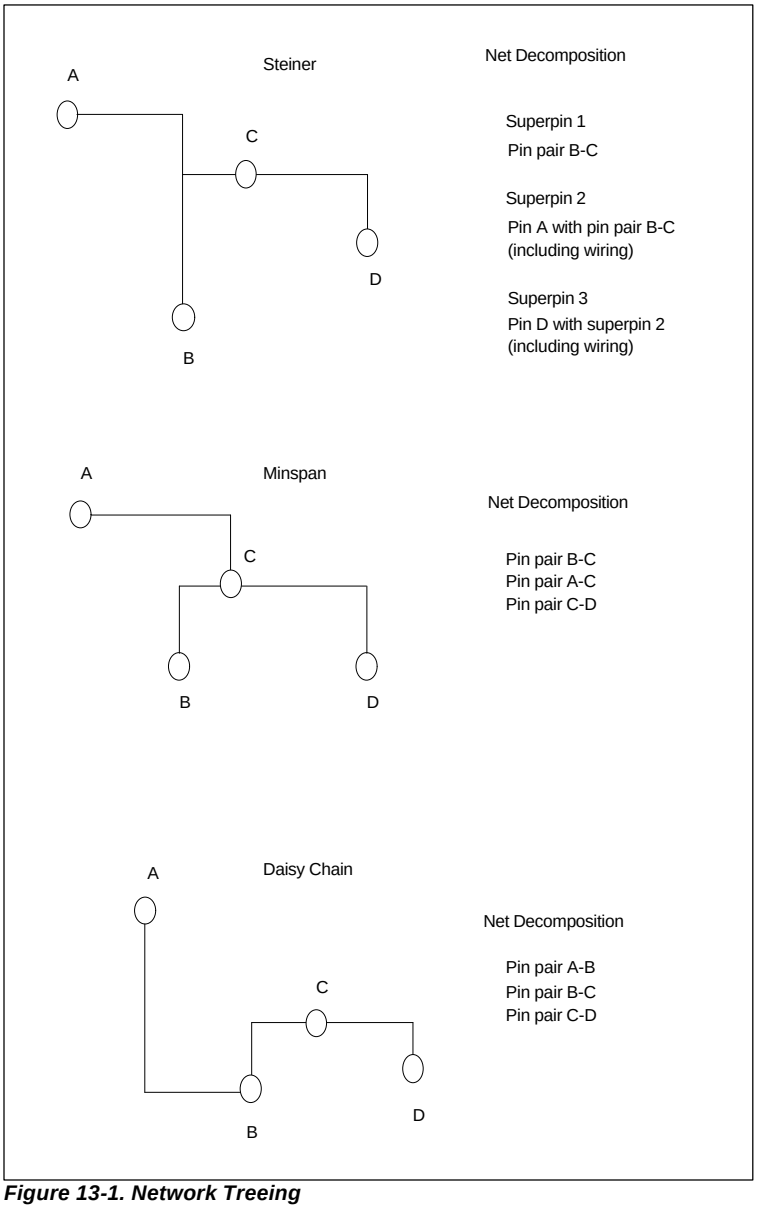

Figure 13-1. Network Treeing

Minspan treeing is the simplest. Subnets are generated so the sum of the pin pair lengths is at the absolute minimum. Assuming the distance is 7 between A and B, 3 between B and C, and 6 between C and D, the best Minspan treeing is B to C, A to C, and C to D.

Note that for a network of *n* pins with no prerouted traces, the number of subnets is *n*1*.

Daisy-chain treeing is similar to Minspan, with the additional constraint that a pin can't belong to more than two subnets.

Steiner is the most effective type of treeing. You can think of it as an extension of MinSpan treeing. As each subnet is generated, it's treated as a superpin that can be paired with other pins. This treatment allows the program to T into existing connections, rather than just route from pin to pin. Using the previous three-pin network example, the Steiner pairing is A to B and C to superpin A to B.

Prerouted traces, jumpers, and inner-plane connectivity add a special aspect to the treeing process. *Autorouter* determines which pins are already connected and doesn't try to connect them again. Prerouted traces can serve as part of superpins and be eligible for T-connections. Similarly, one superpin can be constructed from all pins in a jumper component or from all pins connected to a single inner plane; this type of construction makes it easy for the router to connect other pins in the net to the closest jumper or power pin.

Note: The total number of subnets can be different on the same board. The total depends on whether the board contains prerouted traces.

Nets with buses are handled differently. As described in the Defining Buses section in Chapter 2, "Preparing to Use *Autorouter*," buses are normally defined as large polygonal areas. Nets with buses are broken into subnets so that each pin is paired with the bus or pin closest to it. No buses are paired with any other buses; connecting two large metal areas with a thin trace isn't standard PCB layout practice. You must manually generate valid bus-to-bus connections.

Controlling the Order in Which You Route Connections

Autorouter works on a single subnet at a time. After a subnet is completed, it becomes a permanent part of the layout and is a barrier to further routing (except during rip-up routing). The order in which connections are routed can therefore significantly affect layout results. See the Using Advanced Routing Algorithms section later in this chapter.

A simple and effective ordering scheme is to route subnets from shortest to longest. As more connections are made, the routing becomes more restricted. Since the possible paths of the shortest connections are more constrained than the paths of the longer connections, do the shorter connections first.

Autorouter modifies the ordering scheme somewhat. The memory router, which routes collinear pins (pins that have the same X or Y coordinate), runs first. Wider traces are routed before narrower traces. You can also use net class priority to control the ordering yourself.

Using Memory Routing

The memory router uses a fairly simple approach: it determines all collinear pin pairs. For horizontal routing, it works from the bottom to the top row, left to right along each row; for vertical routing, it works from the left to the right column, bottom to top along each column.

For each pin pair, *Autorouter* attempts a straight route. If that fails, it attempts to route by using a jog with a 45-degree corner. The jog length can be more than one grid; the number depends on the relative sizes of pins and grids. For example, with 50-mil pads, 8-mil lines, and a 20-mil grid, the memory router generates a 2-grid (40-mil) jog. *Autorouter* aligns the 45-degree corners to make the best possible use of routing resources. Figure 13-2 shows the pattern generated by the memory router.

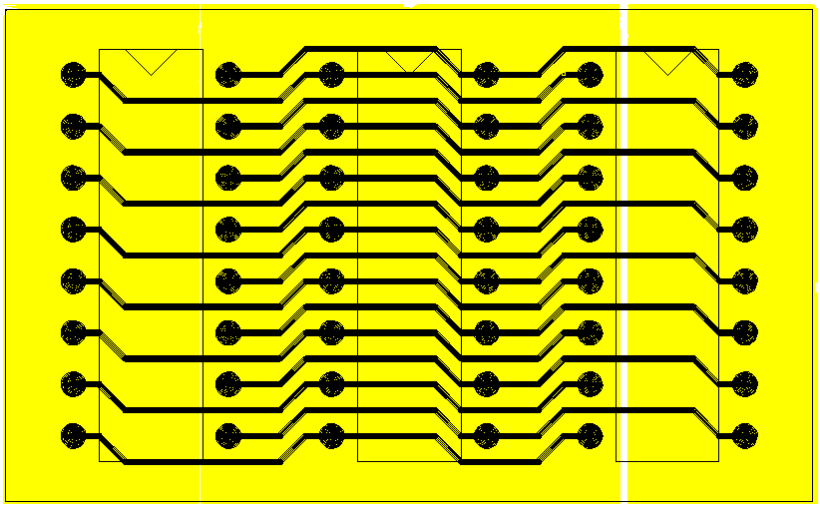

Figure 13-2. Memory Router

Using the Maze Algorithm

The classical routing algorithm is the maze (Lee) algorithm. Its strengths are its general applicability to any technology, its ability always to find a legal route (if one exists), and its performance.

The maze algorithm uses a grid-mapping scheme that treats the routing surface as a series of discrete points. For example, a 10-inch by 8-inch PCB that uses a 50-mil grid maps to a rectangle of 200 grid points by 160 grid points, for a total of 32,000 grid points per

layer. Each grid point requires several bits of information, so the router maps can become quite large. (Make sure you understand the trade-offs that are involved in grid mapping. See the Trading Off Speed and Results section later in this chapter for details.)

The maze algorithm consists of two parts: fan-out and retrace. In a simple example, the fan-out starts at a source grid point (one of the pins). In the first step, all adjacent grid points that are unobstructed by barriers are marked with a 1. In the second step, all unobstructed grid points adjacent to the 1s are marked with a 2. Next, all unobstructed grid points adjacent to the 2s are marked with a 3, and so on. The fan-out continues until the target is reached or until all empty grids are explored and no possible path to the target exists. Figure 13-3 shows fan-out in the first three grids. The retrace finds a path from the target to the source by traveling through grid points marked with successively lower numbers. Figure 13-3 also shows a retrace in the last grid. These steps are the simplified steps of the basic maze algorithm.

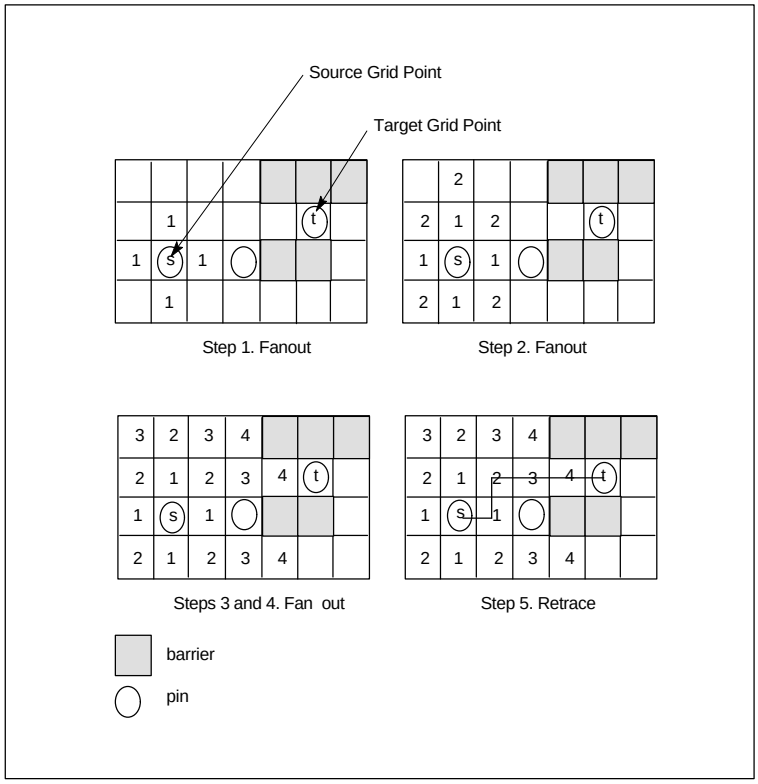

Figure 13-3. Basic Maze Algorithm

The *Autorouter* maze router is much more sophisticated than the simple algorithm described earlier. Some of the more important capabilities are these:

- Both the source and the target can be more than a single grid point.
- A better costing scheme is used. Rather than marking a grid point with its distance from the source, *Autorouter* marks a grid point with the cost to reach it. Examining each possible path, *Autorouter* calculates the number of grids used in each direction per layer and the number of vias. It multiplies each of these by the cost factors specified in the strategy and sums the results. The router finds the path with the minimum cost.

Note: Regardless of the cost values, *Autorouter* always finds a legal path (if one exists). The cost values merely determine which of several alternatives *Autorouter* chooses.

- *Autorouter* uses sophisticated heuristic techniques to defer looking at less likely paths. The program tends to head toward its target and examine fewer alternatives, yet still finds the least-cost path. As a result, the router runs faster and produces paths that are straighter.
- The maze algorithm expands to adjacent diagonal grid points as well as adjacent orthogonal grid points if you select diagonal routing.
- A special technique is used to route surface-mount boards. Just prior to beginning the maze router, *Autorouter* automatically generates a stringer from each committed surface-mount pin. Stringers improve routing results on surface-mount boards by reserving space for a via near each surface-mount pin. The stringer consists of a short trace ending in a through-hole via. Before a connection is routed, its stringers are deleted, leaving space for a via, if needed. Once maze routing is completed, all unused stringers are automatically eliminated.
- Off-grid sources and targets are allowed. *Autorouter* uses the closest grid points and automatically taps into the off-grid features.

Using Post-Route Processes

Autorouter includes several post-route processes, including via minimization, beveling, and acute angle elimination.

The via minimizer looks at each path between two vias or between a via and a pin. If the path can be moved to another layer to eliminate a via, *Autorouter* moves the path to the other layer, disregarding the cost values used in the strategy file.

Autorouter bevels right-angle corners to 45 degrees either during the maze routing process or as a postprocess. The program bevels all corners not blocked by clearance rules or T-connections. The router can also bevel right-angle corners into arcs.

Using the memory router and beveling can result in some acute angles at pins or T-points. *Autorouter* eliminates these acute angles by changing 45-degree angles to 90-degree angles.

Using Advanced Routing Algorithms

Rip-n-Route, the *Autorouter* advanced option, uses several techniques to increase the number of connections made while reducing the number of vias and layers required. The techniques are

- rip-up routing
- optimization
- trace-hugging
- corner penalties

Using the *Rip-Up* Routing Algorithm

The *Rip-Up* routing algorithm that *Autorouter* uses is similar to its maze routing algorithm. The most important distinction is that the *Rip-Up* router doesn't consider traces permanent barriers to further routing. It moves traces by ripping them up and rerouting them.

Traces that are prerouted and identified in the strategy file as not to be altered aren't ripped up or otherwise modified.

The *Rip-Up* routing process is controlled by two parameters: the cost to rip up a subnet, and the cost to reuse the area that a removed subnet formerly occupied. Ideally, every subnet that's ripped up moves to a new area after the subnet that caused it to be ripped up has been rerouted.

A subnet may have to be tried several times before it can be routed. The reuse penalty alters the arrangement of traces and improves the chance of a successful reroute on later attempts. In all cases, the total number of routed subnets is never allowed to decrease. If a

decrease appears likely, *Autorouter* restores the board to the state it was in before the last subnet was ripped up.

Using the *Optimize* Algorithm

The *Optimize* algorithm is also based on the maze routing algorithm. The concept is to remove a set of subnets, then reroute them with a very high via cost. This process tends to use fewer vias, which creates more wrong-way traces. However, as more and more vias are eliminated, the optimizer also tends to shorten traces.

Like the *Rip-Up* router, the *Optimize* algorithm is guaranteed to keep the connection count from decreasing. It also keeps the via count from increasing, unless it can complete an extra connection by using a few more vias. In general, the more *Optimize* passes *Autorouter* is allowed to use, the better the results.

Optimize and *Rip-Up* passes work very well together. *Optimize* passes tend to free board space that the *Rip-Up* router can use. Multiple *Rip-Up* router and *Optimize* passes are therefore interleaved.

Using Trace-Hugging

When trace-hugging is enabled, traces are penalized for traversing free space in all the maze algorithms. This penalty tends to force traces to stay close to (hug) other traces in the vicinity.

When vias are wider than 1 grid, trace-hugging can free considerable board space. For instance, 40-mil vias with a 10-mil clearance on a 25-mil grid are at least 3 grids wide. Without trace-hugging, nearby traces tend to stay 2 grids away from the centerline of a trace containing a via, to leave room for the via. This distance wastes 2 entire tracks (1 on each side).

With trace-hugging, nearby traces stay as close as possible, moving away only in the area near the via. Most of the tracks adjacent to the trace with the via are then available for routing.

Using Corner Penalties

With maze routing, a connection can be routed between two noncollinear pins in many ways. Corner penalties tell the maze routing algorithms to choose paths with fewer corners. Stairsteps are thus dramatically reduced.

Corner penalties don't work well with trace-hugging. Because of the costing method employed, trace-hugging has a higher priority than corner penalties, especially for long traces. If beveling is enabled,

corner penalties also tend to reduce the amount of beveling because there are fewer right angles to bevel. You can experiment to see whether corner penalties work well with your designs.

Understanding the *Autorouter* Architecture

The *Autorouter* program consists of seven subprograms. This section summarizes the *Autorouter* files and describes the subprograms:

- system control (PCROUTE and ROUTE0)
- strategy file editor (ROUTE1)
- database extract (ROUTE2)
- router (ROUTE3)
- database creator (ROUTE4), the return process
- pad builder utility (ROUTE5)

Figure 13-4 shows a flow diagram of the *Autorouter* subprograms.

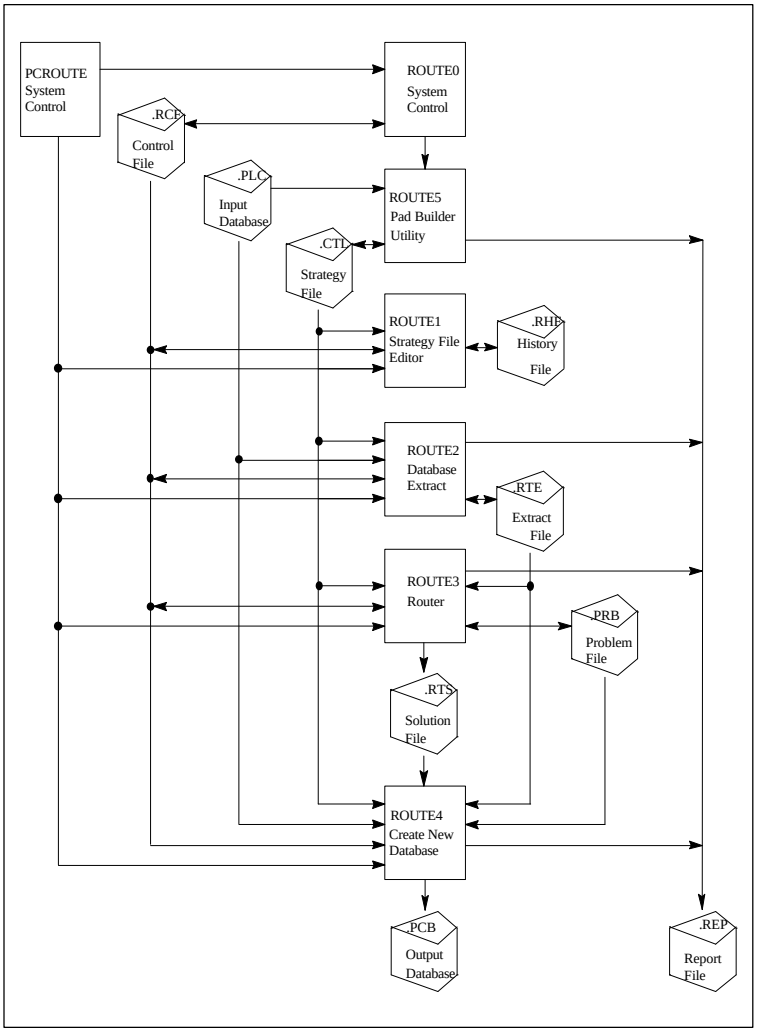

Figure 13-4. Autorouter Subprogram Flow

Understanding the *Autorouter* Files

The *Autorouter* files are as follows:

- An input database (for example, *filename*.plc) is created by using the *PCB Editor* with netlist, component placement, and other information. This file is input to ROUTE2, ROUTE4, and ROUTE5. See Chapter 2, “Preparing to use *Autorouter*.”
- A history file (*filename*.rhf) keeps track of processes run against each database for each project directory. The history file, together with the control file (*filename*.rcf), also contains a list of

strategy files and associated databases and strategy files. This file is created and updated by the strategy file editor, ROUTE1.

- A strategy file (*filename*.ctl) describes the options to be used by *Autorouter*. This file is created and modified by ROUTE1 and is an input file to the database extract (ROUTE2), the router (ROUTE3), and the database creator, ROUTE4. The file is also modified by the pad builder utility, ROUTE5. See the Selecting a Routing Strategy File section in Chapter 2, “Preparing to Use *Autorouter*.”
- An output database (*filename*.pcb) merges the information in the input database and the solution file. The output database file is created by ROUTE4.
- An extract file (*filename*.rte) contains data from the input database necessary for *Autorouter*. The extract file is created by ROUTE2 and is an input file to ROUTE3 and ROUTE4.
- A solution file (*filename*.rts) contains traces and vias created by the router. The solution file is created by ROUTE3, used for resuming the *Route* phase (ROUTE3), and is an input file to ROUTE4.
- A report file (*filename*.rep) summarizes the results of each subprogram. ROUTE2, ROUTE3, ROUTE4, and ROUTE5 append information to this file. See the Correcting Errors section in Chapter 3, “Running *Autorouter*.”
- A control file (*filename*.rcf) contains a list of files used and actions taken during the routing process.
- A problem file (*filename*.prb) contains nets, subnets, pins, and barriers used by ROUTE3 and ROUTE4.

Note: The history file has the same filename as the input database filename, but a different extension. The extract, solution, report, control, and problem files have the same filename as the output database filename, but different extensions.

Controlling the *Autorouter* System (ROUTE0 and PCROUTE)

PCROUTE and ROUTE0 provide system control for the router.

PCROUTE is a very small program that remains in memory throughout *Autorouter* execution. PCROUTE is separate from ROUTE0 and ROUTE1 to minimize the size of the memory-resident portion of *Autorouter*.

PCROUTE starts ROUTE0. Then ROUTE0 tells PCROUTE to run ROUTE2, ROUTE3, ROUTE4, and ROUTE5 according to your choices on the ROUTE1 opening screen.

Editing Strategy Files (ROUTE1)

ROUTE1 creates and modifies strategy files. It uses an old strategy file as input and either rewrites it with modifications or creates a new strategy file. All strategy files are displayed by ROUTE1. If a new strategy file is created, its name is entered in the history file (*filename*.rhf). ROUTE1 also chooses the subprograms *Autorouter* will execute in order to route a board using the new strategy file.

Extracting Databases (ROUTE2)

ROUTE2 extracts data required by the router from the input database. This data includes the board outline, barriers, pins, prerouted traces, and netlist. ROUTE2 also performs network treeing, breaking up nets into subnets according to the specified route type (min-span, daisy chain, or Steiner). ROUTE2 uses the input database and strategy file as input, creates the extract file as output, and appends information to the report file.

Running *Autorouter* (ROUTE3)

ROUTE3 is the main routing subprogram. It uses the extract and strategy files as input and produces the solution and problem files. It also appends information to the report file.

Creating a New Database (ROUTE4)

ROUTE4 creates the new database by merging the information from the input database with the traces and vias created by the router. The input files are the input database, strategy, extract, solution, and problem files. The output file is the output database.

ROUTE4 also appends information to the report file and generates the can't-connects on the $CONT layer in the output database.

Generating Pad Descriptions (ROUTE5)

ROUTE5 automatically generates pad descriptions. It uses the input database and modifies the contents of the strategy file. ROUTE5 also appends information to the report file.

Running a Batch File

You can create a batch file so you can run the *Autorouter* subprograms several times consecutively. This section describes how to create command lines and a batch file.

Creating Command Lines

To use command lines, type the subprogram name at the DOS prompt, followed by filenames and options, then press . The command line formats for ROUTE2, ROUTE3, and ROUTE4 are

route2 input_database reference_name strategy [design_name]

route3 reference_name strategy [(new | resume) [design_name]]

route4 input_database output_database reference_name strategy [design_name]

where

route*n* identifies the subprogram.

input_database is the name of the unrouted database file, including the extension.

reference_name is the name that serves as a prefix to all files created by *Autorouter*. Usually, this prefix is the same as the output database filename prefix.

strategy is the name of the strategy file, without the .ctl extension.

output_database is the name of the newly created database, including the extension.

resume indicates that the route phase was interrupted and is being resumed. The resume option isn't a required option. If you're starting a new route, enter **new** instead of **resume**. (If you don't enter a design name, don't enter **new**.)

design_name is the name you assign to this project; its use is optional. Usually it's the name of your working directory (the directory from which you're running *Autorouter*).

Creating a Batch File

A batch file can contain command lines for running several programs consecutively. Use a text editor to create command lines. Instructions for creating command lines are shown above.

You must use the .bat filename extension when you're naming the batch file. You must also either store the batch file in the working directory or include a path that indicates where it's located when you run it. Figure 13-5 shows a sample batch file.

```
cd cadroj0
ROUTE2 ezboard.plc ezboard pcad1 proj0
ROUTE3 ezboard pcad1 new proj0
ROUTE4 ezboard.plc ezboard.pcb ezboard pcad1
proj0
cd cadroj1
ROUTE3 test fineline resume proj1
ROUTE4 test.pkg test.pcb test fineline proj1
```

Figure 13-5. Sample Batch File

To run a batch file

- Enter the filename (without the .bat extension) at the DOS prompt.

Trading Off Speed and Results

Autorouter uses maps of the routing surface that require several bits to represent each grid point. This section describes the size requirements for the maps, the memory management techniques that *Autorouter* uses, the effects of the grid size, and how you can use this information to trade off execution speed against routing results.

To get an idea of the size of the maps, consider a 12-inch by 10-inch six-layer board on which all the vias are through-hole vias. To run the maze routing algorithm on all layers requires either 614 KB for a 50-mil grid or 3669 KB for a 20-mil grid.

Managing Memory and Grid Size

Autorouter can store map data in two places:

- in extended memory
- on disk

Autorouter first uses extended memory to store the maps. If not enough extended memory is available, *Autorouter* uses DOS memory, and if that runs out, *Autorouter* uses a temporary disk file and swaps the data to and from disk. Disk swapping is very slow and can result in *Autorouter* running as much as five times longer than if no disk swapping is required.

Using a smaller grid size forces *Autorouter* to examine more alternatives. For example, halving the grid size quadruples the number of grid points (twice as many grid points in each direction). Halving the grid size causes *Autorouter* to run about four times slower, even if no disk swapping isrequired.

Extended memory is required to use *Autorouter*. The *Extract* phase calculates the amount of disk space and extended memory that *Autorouter* needs during the *Extract* phase. The calculations are displayed in the Route Extract Report section of the report file, *filename*.rep. The values shown in the report are based on the strategy file and the input database used when the *Extract* phase is run.

As a general rule, the amount of available disk space you need is equal to about 10 times the size of the input database. The extract report gives a more accurate measure of the disk space you need.

An example of the extract report is shown in Figure 13-6. The size of the input database used in this example is 12,034 bytes.

The extract report displays the disk file space needed and three estimates of the amount of extended memory required.

The first estimate, the optimum setup, is the amount of memory that's almost guaranteed to prevent any disk swapping having to take place because of insufficient memory. However, disk reading and writing take place to load the data initially, write results, and record intermediate results to prevent data loss in the event of a power failure. Since the optimum setup value is very large, it's not likely that the calculated amount of extended memory will be available. It often isn't practical to implement the optimum setup.

```
P-CAD    ROUTE   EXTRACT REPORT  VERSION 8.0

Start Time:          8/15/1994          11:26:14
Design:              p_build
Database:            p_build.pcb
Strategy:            p_build.ctl

WARNING              A3001 One barrier on pins (3450, 3200)  (3450, 330)  different
nets.

Nets: 128            Subnets: 322       Pins: 696
Disk file space needed: 0.49 Mbytes (estimated)
Estimated memory needed:
  (based on map size estimate of 0.11 Mbytes and 113 x 81 x 42 layers)
Optimum setup:       2.19 Mbytes
  Good setup:        1.83 Mbytes
  Fair setup:        1.23 Mbytes

End Time:            8/15/1994          11:26:26

END OF PCAD ROUTE EXTRACT REPORT
```

Figure 13-6. Extract Report

The second estimate, the good setup, is based on reasonable predictions of how the router will perform. The predictions assume that the routing can be performed in a reasonable manner, the board isn't impossible to route, and few unusually long traces will be generated. If your board conforms to these assumptions, your good setup values result in relatively infrequent disk swapping.

The last estimate, the fair setup, represents the least amount of memory you can have and still achieve tolerable results. Even less memory can be used to perform a successful route, but it may take six to ten times as long as the optimum setup, where no disk swapping takes place.

For any of the three setup values, you must have a minimum of 2 MB of available extended memory, but 3 to 7 MB are needed for most applications. In addition, regardless of the setup value, the minimum memory requirements for the *PCB Editor* must be met as well, or you won't be able to edit your routed board. Consider running some experiments to get an idea of how different grid sizes and strategy file configurations affect router performance and results. You can then develop a set of strategies to best meet your design requirements.

Using Re-Entrant Routing

Re-entrant routing is an approach to routing that lets you combine your expertise and *Autorouter's* speed and accuracy to

- improve the quality of your PCB
- reduce problems during the manufacturing process

Basically, you alternate between using *Autorouter* to work on a small section of the board and using the *PCB Editor* to modify and enhance the results. You re-enter *Autorouter* several times, hence the term re-entrant routing.

Note: Each time you enter *Autorouter*, you have to re-extract and rerun all three phases of *Autorouter*, as described in Chapter 3, “Running *Autorouter*.”

The following is an example of how you can use re-entrant routing:

1. Have *Autorouter* route only your power and ground networks. Use the *PCB Editor* to manually modify the results.

2. Have *Autorouter* route the connectors on the board. Use the *PCB Editor* to modify the results.

3. Have *Autorouter* route any difficult components, such as pin grid arrays. Use the *PCB Editor* to modify the results.

4. Have *Autorouter* complete the rest of the board. Use the *PCB Editor* to finish the can't-connects and complete the layout.

Steps 1, 2, and 3 can usually be completed in a few hours. You can then run *Autorouter* overnight to do the bulk of the routing. The next day you can complete the board.

You can use re-entrant routing several ways to develop a methodology consistent with your design style and technology requirements.

Review the discussion on net classes in Chapter 9, “Defining Net Classes,” so you can set up *Autorouter* to work on a limited number of nets or components. The idea is to create a net class containing only those nets that you wish to automatically route and to assign a priority of *IGNORE* to the default net class, so no other nets will be processed. Run *Autorouter*, edit the results in the *PCB Editor*, and change the net class definitions to route another subset of the board. Repeat this process to route all the difficult areas of the board. Then, on the last iteration, remember to set the priority of the net class to *DEFAULT* so *Autorouter* routes the rest of the nets.

Improving Stringer Handling

You can make the following improvements in stringer handling:

- You can return stringers to the database when you run the Autorouter with the maze, ripup, and optimize algorithms set to off (zero passes). For best results, you should also set all other passes (memory, via minimization, beveling, eliminate acute angles) to off. The Autorouter returns stringers that appear as antennas to the database. A subsequent router extraction deletes these and any other stringer antennas unless you've used *Net Class Definitions* to place them in a net class marked “do not alter.”

To make sure only wanted stringers are generated and returned to the database, and to avoid needless warnings about unneeded stringers, use *Net Class Definitions* to

- establish net classes for the stringers you want
- mark Prerouted Traces "do not alter"
- set the priority of all other net classes to "Ignore"
- You can set the via type assigned to the stringers in the Net Class Definitions screen. The default value for the Stringer Via Type is 0 (through-hole via).

When interstitial vias are used for the stringers, you need a net class for ground and a net class for power. If there are surface mount devices on both sides of the board, two stringer generation passes are required:

- a pass for the COMP to the ground layer and the COMP to the power layer
- a pass for the SOLDER to the ground layer and the SOLDER to the power layer

You must run the returner between the two passes, and the Prerouted Traces for the power and ground net classes must be set to “do not alter.”

Note: The stringer algorithm uses the component outline to place stringers. The algorithm gives priority to placing stringerts at right angles to the long side of the component, outside the component outline. If the stringer via type is connected to an inner layer plane, however, the algorithm attempts to place the associated stringers inside the component outline.

Glossary

This glossary contains brief descriptions of terms used throughout this manual. The body of the manual contains more detailed descriptions.

air gap—see *clearance*.

air line—see *flight line*.

algorithm—a step-by-step computational procedure for solving a problem.

Autorouter—a P-CAD tool that automatically follows the rules you define to generate legal wiring paths (traces and vias) for all unfinished nets on a printed circuit board (PCB).

barrier—a user-defined area where *Autorouter* can't place traces or vias. See also *trace barrier* and *via barrier*.

beveling—the process that changes 90-degree corners to 45-degree corners or to arcs.

blind via—see *interstitial via*.

buried via—see *interstitial via*.

bus—a rectangular or polygonal metal strip that distributes a common signal (usually power or ground) throughout the board. *Autorouter* contains special features to handle networks with buses.

can't-connect—the portion of a net that can't be completed by the automatic router.

clearance—the minimum distance required between the edges of two items (traces, vias, or pins). Line-to-line, line-to-pad, and pad-to-pad clearances have separate values. Also called an *air gap*.

costing—a method for calculating the expense of a particular path. You assign cost values to six cost elements and *Autorouter* calculates the least expensive legal path. The six cost elements are North/South, East/West, NorthEast/SouthWest, NorthWest/SouthEast, through-hole via, and intersititial via. You can also assign different cost element values for each layer. See also *preferred direction*.

daisy chain—a type of autorouting that permits routing only along paths between pins, allows a pin to be directly connected to no more than two other pins, and doesn't permit T-junctions with previously routed traces.

database—a disk file containing all the information about a particular printed circuit board. The data *Autorouter* uses includes traces, vias, board outlines, component placement information, and connectivity definitions.

database layer—a collection of related information in a database. Database layers can be thought of as clear plastic sheets that are stacked on top of each other to form a complete drawing. A database layer can correspond to a physical board layer, although several database layers are often used to describe the information for one physical board layer.

DBU—database unit. If English (US) units are selected, a DBU is 1 mil. If metric units are selected, a DBU is 0.01 millimeter (mm).

diagonal routing—an option that enables the maze routing algorithm to generate diagonal (NorthEast/SouthWest or NorthWest/SouthEast) paths as well as orthogonal (North/South or East/West) paths.

disconnect—see *can't-connect*.

disk swaps—the number of times the maps used by *Autorouter* are swapped between memory and disk. Disk swaps occur only if the maps can't fit in the available DOS memory. See also *maps* and *extended memory*.

extended memory—an option used by *Autorouter* to extend the DOS memory limits beyond 640K. You must have extended memory installed to run *Autorouter*.

feedthrough—see *via*.

fine-line technology—a board fabrication technology that uses thinner line widths and smaller clearances to allow more than one trace to run between component pins that are 100 mils apart, from one pin center to another pin center.

flight line—a straight line showing the two pins that *Autorouter* is trying to connect. Also called an *air line*.

grid—see *routing grid*.

interstitial via—a plated hole used to connect a signal between two layers in a multilayer board. An interstitial via pierces only two layers, where one or both of the layers is an internal layer. If both layers are internal, an interstitial via is also called a *blind via* or *buried via*. An interstitial via contrasts with a through-hole via, which pierces all layers.

irregular grid—a routing grid with unequal distances between grid points. See also *routinggrid*.

lattice—see *via lattice*.

layer—see *routing layer* and *database layer*.

layer pair—two routing layers that are paired for routing. You can specify that *Autorouter* is to work on either one layer pair at a time or on all layers simultaneously.

major grid—see *irregular grid*.

maps—the internal representations of the board topology. Map size is directly proportional to the number of routing grids. If the maps are too large, the system swaps them to disk. See also *disk swaps* and *extended memory*.

maze routing—a grid-based algorithm used by *Autorouter*.

minor grid—see *irregular grid*.

minspan—a type of autorouting that permits routing only along paths between pins and doesn't permit T-junctions with previously routed traces.

multilayer technology—a board fabrication technology with more than two copper or trace layers.

net—a collection of electrically equivalent features, including pins, traces, vias, and buses. *Autorouter* attempts to connect all unconnected portions of a net.

net class—a collection of nets that requires special handling by *Autorouter*, such as specifying a larger trace width or specifying a higher routing priority.

off-grid—a feature that isn't centered on the routing grid.*Autorouter* routes to off-grid pins.

optimization—a routing technique available when the Rip-n-Route advanced routing option is installed. Optimization eliminates vias after routing by rerouting existing connections. Typically, the router completes more connections during and after optimization, since reducing vias frees router resources.

orthogonal—horizontal or vertical.

pad description—the shape of the metallic area associated with a pin or via. Pad descriptions are usually circular, square, or rectangular.

pass—a single cycle of a routing algorithm.

pin—a physical point on a component where an electrical connection can be made.

preferred direction—the most likely direction that the router will use on a particular layer; the direction with the lowest cost. You can specify North/South, East/West, NorthEast/SouthWest, or NorthWest/SouthEast as the preferred direction.

prerouted traces—traces and vias generated previously. Whether or not *Autorouter* connects to prerouted traces depends on the route type you select.

regular grid—a routing grid with equal distances between all grid points. See also *routinggrid*.

re-entrant routing—a routing technique that involves alternating between an automatic router to route a small section of your design, and an interactive editor to clean up the results manually. Hence, you re-enter the automatic router several times.

rip-up—a routing technique available when the Rip-n-Route advanced routing option is installed. Rip-up completes difficult connections. With rip-up routing, a blocked trace removes other traces until it gets to its target. The removed traces are then rerouted, avoiding the original trace.

route box—a rectangular area within which *Autorouter* routes; the program restricts its search for legal paths to the route box. Modifying the size of the route box lets you trade off program run time against routing results.

routing grid—points along which *Autorouter* generates traces. *Autorouter* generates traces and vias that are centered on grid.

routing layer—a physical board layer containing traces and vias. Also called a *signal layer*.

signal layer—see *routing layer*.

Steiner—a type of autorouting that permits T-junctions with previously routed traces. This is the most efficient type of autorouting because it has the fewest restrictions.

strategy—a user-defined list of variables and rules that controls the autorouting process.

stringer—a short trace starting at a surface-mount pin and ending in a through-hole via. Stringers improve routing results by providing more layers on which to connect to a surface-mount pin. Prior to routing, the program automatically generates stringers from all surface-mount pins; after routing, unused stringers are eliminated.

stub length—the (typically short) distance from a pin that *Autorouter* can use to generate T-junctions during min-span and daisy-chain routing. See also *min-span* and *daisy-chain*.

subnet—a portion of a net.

surface-mount component—a component mounted only on the surface of the board; the connections to the pins of the component are restricted to a single layer. A surface-mount component contrasts with a through-hole component, which can have connections on any signal layer.

through-hole component—a component mounted through the surface of the board; connections to the pins of the component can occur on any signal layer. A through-hole component contrasts with a surface-mount component, which can have the connections only on a single layer.

through-hole via—a plated hole that connects a signal on two or more routing layers. A through-hole via pierces all layers. A through-hole via contrasts with an interstitial via.

trace—a line on a single routing layer that conducts an electrical signal.

trace barrier—a user-defined area where *Autorouter* can't place traces or vias. See also *barrier* and *via barrier*.

trace hugging—a routing technique that's available when the Rip-n-Route advanced routing option is installed. Trace hugging forces traces to follow the contours of previously routed traces. Trace hugging tends to free routing channels, especially when vias are two or more routing grids wide.

via—a plated hole that connects signals on two or more routing layers. Also called a *feedthrough*. See also *through-hole via* and *interstitial via*.

via barrier—a user-defined area where *Autorouter* can't place vias (but can place traces). See also *barrier* and *trace barrier*.

via lattice—a set of grid points where it's legal to place vias. If you define a via lattice, it's illegal to place vias on points outside that lattice.

via minimizer—after routing is completed, a process you can run to eliminate vias by moving traces to other layers.

wavefront—the set of points that represents the ends of the paths *Autorouter* is considering.

wiring rule—a rule, associated with a net class, that defines the trace width and clearances to be used when the nets are autorouted in the net class.

45-degree grid—two points between which *Autorouter* generates a trace. One point is one grid over and one grid down from the other point.

Index

A

adding
  net class definitions, 17
  wiring rules, 8
algorithms
  rip-up
    rip-up, 59
Autorouter architecture
  controlling the system, 63
  creating new databases, 64
  editing strategy files, 64
  extracting databases, 64
  generating pad
    descriptions, 65
  running the Autorouter, 64
  understanding file
    relationship, 61

B

batch files
  creating, 66
  creating command lines, 65

C

Check boxes
  Wave, 49
command lines
  in batch files, 65
commands
  Continue, 40
  Exit, 41
  Fit View, 42
  Pan, 43
  Redraw, 45
  Skip Pass, 46
  Skip Subnet, 47
  View Window, 48
  Zoom In, 50
  Zoom Out, 51
connect electrically equivalent
  points
  using maze routing, 56
  using memory routing, 56
  using network treeing, 53
  using ordering, 55
  using post-route processing
    using post-route
      processing, 58
Continue command, 40

D

databases
  creating, 64
  extracting, 64
defining
  net class by component, 19
  net class by net list, 18
  net class names, 15
  net class type, 17
  net classes by reference
    designators, 18
  net classes by window, 21
  net routing layers, 16
  net routing priority, 15
  padstack data, 35
deleting
  net class definitions, 22
  net class from components
    list, 21
  net names from net lists, 18
  reference designators from
    the designators list, 19
  wiring rules, 9

E

editing
  layer costs, 31
  layer description, 32
  layers, 30
    editing type, 31
    editing layer names, 30
  pad-to-pad clearances, 8
  strategy files, 64
  wiring rules, 9
examples
  naming wiring rules, 10
  routing using net class
    definitions, 23
execution speed
  leveraging, 66

Exit command, 41
exiting
  Layer Descriptions screen, 33
  Net Class Definitions screen, 22
  Wiring Rules Definitions screen, 9
extracting databases, 64

F

files
  creating batch, 66
Fit View command, 42

I

Integrated Aperture Table, 37

K

keyboard
  using to enter commands, 39

L

Layer Descriptions screen
  defining number of trace layers Autorouter recognizes, 28
  defining via costs, 28
  description, 27
  editing layer costs, 31
  editing layer descriptions, 32
  editing layers, 30
  exiting, 33
layers
  editing, 30
    costs, 31
    descriptions, 32
    name, 30
    type, 31

M

maze routing
  using to connect electrically equivalent points, 56
memory
  managing, 67
memory routing
  using to connect electrically equivalent points, 56

N

naming
  wiring rules using trace width, 11
Net Class Definitions screen
  adding net class definitions, 17
  defining
    class type, 17
    net class by component, 19
    net class by net list, 18
    net class by reference designators, 18
    net class by window, 21
    net class names, 15
    net routing layers, 16
    net routing priority, 15
  deleting
    net class definitions, 22
    net class from components list, 21
    net names from net lists, 18
    reference designators from the designator list, 19
  description, 13
  displaying wiring rule names, 15
  enabling Autorouter to alter prerouted traces, 16
  examples of naming wiring rules, 10
  examples of routing by using net class definitions, 23
  exiting, 22
  opening, 13

nets
  adding net class definitions, 17
  defining
    net class by component, 19
    net class by net list, 18
    net class by reference designator, 18
    net class by window, 21
    net class names, 15
    net class type, 17
    routing layers, 16
    routing priority, 15
  deleting
    net class definitions, 22
    net names from net lists, 18
    net names from the components list, 21
  examples of routing using net class definitions, 23
network treeing
  using to connect electrically equivalent points, 53

O

ordering
  using to connect electrically equivalent points, 55

P

pad builder utility
  description, 35
Pad Builder Utility screen
  defining padstack data, 35
  running the utility, 38
pads
  defining padstack data, 35
  editing pad-to-pad clearances
    editing pad-to-pad clearances, 8
  flashes describing, 37
  generating descriptions, 65
  running pad builder utility, 38
Pan command, 43
post-route processing
  post-route processing
    using to connect electrically equivalent points, 58

R

Redraw command, 45
Rip-n-Route
  Rip-n-Route
    corner penalties, 60
    rip-up routing algorithm
      rip-up routing algorithm, 59
    trace-hugging
      trace-hugging, 60
Route phase
  commands
    Continue, 40
    Exit, 41
    Fit View, 42
    Pan, 43
    Redraw, 45
    Skip Pass, 46
    Skip Subnet, 47
    View Window, 48
    Wave, 49
    Zoom In, 50
    Zoom Out, 51
routing
  using re-entrant
    using re-entrant, 69
running
  Autorouter, 64
  pad builder utility, 38

S

screens
  Net Class Definitions
    opening, 13
Skip Pass command, 46
Skip Subnet command, 47
speed
  leveraging execution, 66
strategy files
  editing, 64

T

trace widths
  using to name wiring rules, 11
traces
  enabling Autorouter to alter prerouted, 16
  using corner penalites, 60
  using trace-hugging
    using trace-hugging, 60

V

vias
  defining costs, 28
View Window command, 48

W

Wave check box, 49
Wiring Rule Definitions screen
  adding wiring rules, 8
  deleting wiring rules, 9
  description, 7
  editing pad-to-pad clearances
    editing pad-to-pad clearances, 8
  editing wiring rules, 9
  exiting, 9
  opening, 7
wiring rules
  adding, 8
  deleting, 9
  displaying, 15
  editing, 9
  example wiring rule clearances, 10
  examples of naming, 10
  naming using trade width, 11

Z

Zoom In command, 50
Zoom Out command, 51