# University Of Manchester PCB Software
# Department of Computer Science
# Version 2.4

Doug Edwards Alistair MacIntosh Zahir Moosa Fred Hoyle

October 6, 1994

## Contents

# 1 Introduction

## 1.1 Overview

The Manchester University Computer Science (MUCS) PCB software allows designs to be entered schematically (using ViewDraw) or textually; designs can be placed using a graphical interactive editor and then rapidly routed using a highly effective router. Plots of the resulting board can be printed on a laser printer or HP pen-plotter. Finally industry standard GERBER photo-plotter files can be produced to interface to any PCB manufacturing house. MUCS possesses a fabrication facility specially equipped for low-volume, low-cost prototypes. Board sizes up to 18in square, with up to 10 layers, can be fabricated (see Appendix F).

The tracker employs a variety of algorithms under user control. Various design rules can be specified (see Section 5.1, page 15.

## 1.2 Environment Variables

The software requires a number of environment variables to be set and exported.

**MUCS_PCB_DIR** This variable must point to the directory in which the pcb software is installed. For convenience only, in the rest of this document, the directory is assumed to be *pcb*.

**OUTLINE_LIBS** This variable is optional. It should point to a list of directories containing locally defined component outlines and pin positions (see section 5.4, page 17). Note that list of directories is *space* separated and therefore must be quoted when defining OUTLINE_LIBS

**NCAP_PRINT** This variable is optional. It may be required if listing ncap data (section 2.6, page 8) on a laser printer.

## 1.3 Search Path

The various binaries used in the software suite must be added to your normal search path. They are to be found in *pcb/bin/sun3* or *pcb/bin/sun4*.

## 1.4 Documentation

A copy of this text can be found in the *£MUCS_PCB_DIR/doc/design_tools* directory. **Figure 1** shows the how the various *pcb* binaries communicate with one another and the different files and formats generated.

## 1.5 Examples

A number of examples of real boards that have been designed using this system are to be found in *pcb/examples* directory.

## 1.6 Deficiencies

- Although the system is believed to be robust, no support can be guaranteed.
- Routing results are sometimes not pretty. A post processor is required to eliminated unnecessary “jogs”
- The layout editor only runs under sunview (or Openwindows + compatibility mode) and is rather clumsy (but the autorouter is so good you don’t need a layout editor!!!).
- There is no *graphical* way of defining component footprints
- Text on boards is not supported, nor silk screenlayers generally.
- Only rectangular board definitions are supported.
- Boards for analogue circuits are not well catered for.
- An understanding of basic UNIX commands is an advantage.

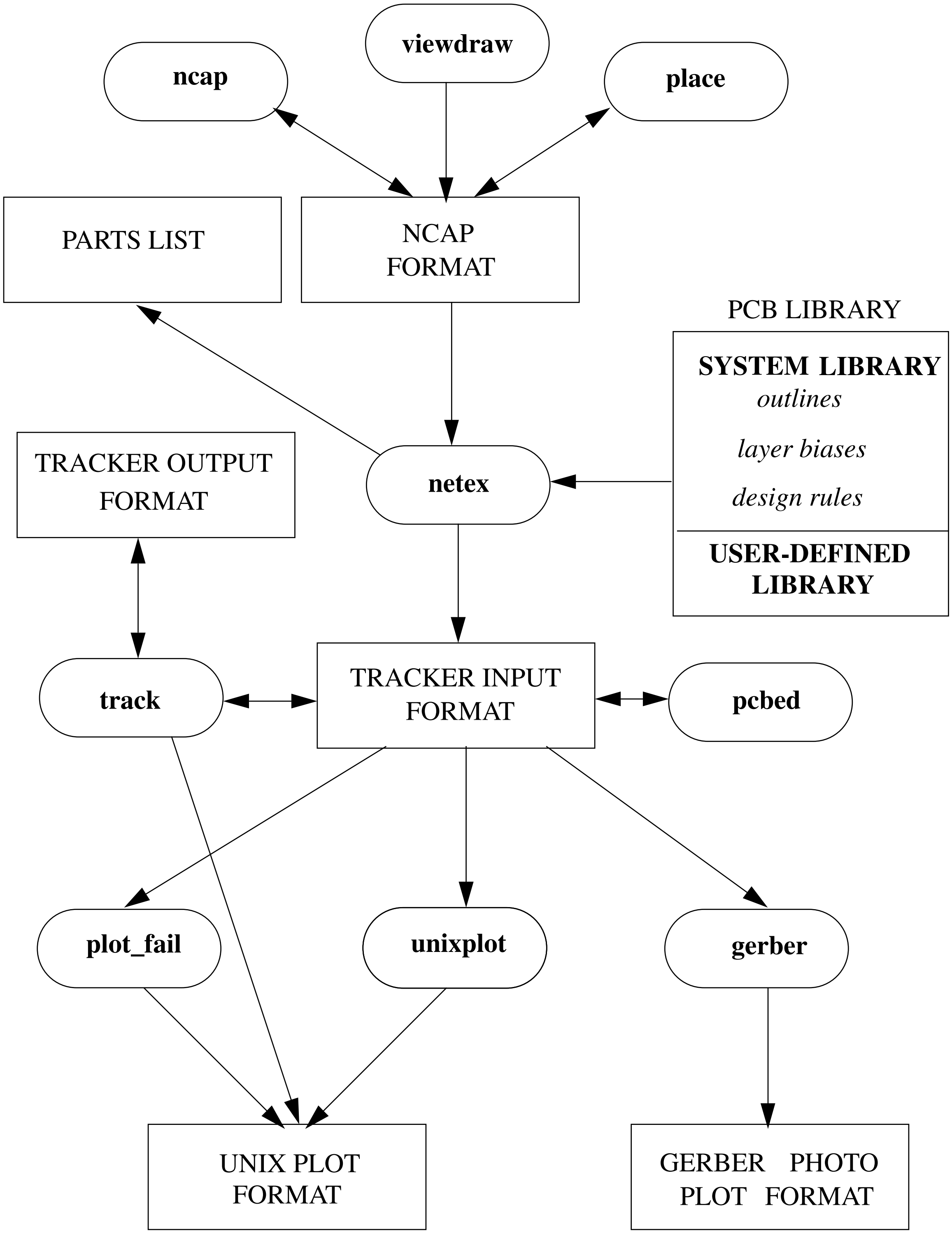



**Figure 1 : PCB design software structure**

# 2 Ncap - High level textual capture

This program is responsible for capturing a high level textual description of a board layout which is maintained in a single file. It is invoked as:

**ncap** *design*

*design* is the name of the file in which the output from **ncap** is written to and should be unique. When **ncap** is invoked to create a new file, it first requests the board type. The current options are:

- single eurocard - two and four layer
- double eurocard - two and four layer.
- treble (extended) eurocard - two and four layer.
- other - arbitrary sized board.

The above options can be toggled using the space bar (or any other key) and the required option selected using carriage return. Selection of the *other* option must be followed by additional data specifying the size and number of layers and the coordinates of the bottom left hand coordinate of the board (in thou). This additional data will be prompted for by **ncap**. The number of layers specified should include the number of power planes. Having set the size of the board, **ncap** then enters the user in command mode, distinguished by the *ncap*> prompt. **Ncap** maintains a record for each individual module i.e. dilic, resistor pack, connector etc on the board. Each module description is divided into 2 parts - a description of the module (i.e what the module is, number of pins etc.) and - a list of signal names for each pin.

## 2.1 Creating New Modules

Each module has a unique number (in the range 0 - 499) to identify it. To create, or subsequently modify a module description, the following command may be used:

**M**<*module nunber*> <*CR*>

The user then has to select the module type (again toggling through the various options using any key, and selecting using the return key), setting the number of pins and description of the module. The module description is simply a comment with a maximum of 50 characters. It is recommended to reserve the first part of the description for the device type e.g *74LS374*. Some components have an additional distinguishing feature that can be varied – for example some dilics have a pitch option, single pads have a choice of pad diameter. The user is then asked for confirmation. If not, the module may be modified. When modifying a module description, a field may be skipped using the return key or modified by overtyping. **Appendix C** lists the modules available in the default device library.

## 2.2 Entering Signal Names

Once a new module has been created, the signal names can then be entered. This is done using the screen editor *vi* by default. Alternatively the editor specified by the environment variable EDITOR can be used. Since the editor is typically opened and closed several times during **ncap** sessions, *emacs* users may wish to use a screen editor with a quicker startup time e.g *jove*. This may be achieved by running **ncap** in a sub-shell i.e:

( EDITOR=jove ; **ncap** *filename* )

The signal names should be entered in numerical pin order using separate line for each pin. The editor *vi* will automatically display the pin number next to the signal name. The following convention has been adopted for signal names. Each signal names has 3 constituents:

- The actual name which is an alphanumeric string of at most 63 characters. All characters A-Z, a-z, 0-9 and underscore are allowed.
- An optional offset - any positive integer.
- An optional polarity - one of the characters +, -, h, l, H or L.

Each of the fields must be separated by a '.'. Examples of valid signal names include:

fred
fred.1.h
fred.+
fred_0.1.1
FRED.1.L

In addition, there are 5 signal names with special meanings.

**1** NC - an unconnected pin

**2** VCC - On multi-layer boards this connects to the power layer. On 2 layer boards this is effectively the same as NC.

**3** GND - Connects to the ground plane for multi-layer boards; otherwise the same as NC.

**4** SIG_VCC - On a multi-layer board, provides a cuttable line segment leading to the power layer; otherwise the same as NC.

**5** SIG_GND - Similar to SIG_VCC but for the ground plane.

When the editor is exited, the syntax of the signal names is checked. All incorrect signal names are printed and the user is given the option to re-edit them. If the user wishes to modify only the pin names pertaining to a previously created module, the following command may be used to enter the editor directly:

**m** *<module number>* *<CR>*

## 2.3 Positioning Of Modules

To position a module the following command may be used:

**p***<module number>* *<CR>*

Once the above command has been specified, the orientation of the module is entered followed by the x,y coordinates of pin 1. The orientation is made relative to an arrow pointing along the body of the module from pin 1. On SUN workstations, there is also an interactive program **place** (see Section 2) for positioning modules.

## 2.4 Duplicating Modules

Modules may duplicated using the copy command, **c**:

**c** *<source module>* *<target module>* *<CR>*

A group of consecutive target modules may be specified using the following syntax:

**c** *<source module>* *<target module>-<target module>* *<CR>*

The copy will not act destructively (i.e will not overwrite an existing module).

## 2.5 Deleting Modules

This may be achieved using the command:

**d** *<module number>* *<CR>*

The user is then asked to confirm the deletion. A range of modules may be specified in the same manner as for the copy command.

## 2.6 Listing Data

There are 3 listing commands available:

**l** <*CR*>

lists the signal names for each module together with its position and orientation if it has been specified. This is piped through the UNIX *more* command. So for example, */module n*<*CR*> will skip to the listing of module *n* and *q* will return directly to the command mode. The command:

**L** <*CR*>

sends a two column version of the listing to the line printer. A separate process produces the listing, so it can be some time after the command returns before the listing appears.**ncap** assumes that the output will appear on an old-fashioned 132 column line printer. If the output is to appear on a A4 laser printer, an alternative printing command can be defined by means of the environment variable *NCAP_PRINT*:

```
NCAP_PRINT='enscript -2r -Plw'
export NCAP_PRINT
ncap <design_name>
```

**s** <*CR*>

generates a short form of the listing containing the module numbers and titles. The listing can be restricted to a subset of modules e.g

**l** *200-300*

lists the signal names for modules 200 to 300 inclusive.

**l** *20*

lists all the modules after 20.

## 2.7 Other Commands

**e** <*CR*>

exits from **ncap** and writes everything back to the data file.

**q** <*CR*>

quits **ncap** without writing back to the data file. The user is asked to confirm this command.

**w** <*CR*>

writes to the data file without exiting **ncap**.

**h** <*CR*> or **?** <*CR*>

gives a brief list and description of all the available commands.

**x** <*CR*>

writes to the datafile and runs **netex** (see Section 3) without exiting from **ncap**.

**b** <*CR*>

allows the user to modify the board dimensions.

## 2.8 Ncap File Format

The format of the datafiles generated by **ncap** is shown below:

```
ncapture
version (default 0)
board type (not used - 0)
number of layers (1..n)
x size of board in thou
y size of board in thou
x start relative to (0,0)
y start relative to (0,0)

FOR each MODULE
        module number (0-499) (Viewdraw RefDes attribute)
        module type (device library device number.)
        number of pins
        variant (pitch) code (eg for a dilic, 0=300 thou pitch, 1=600 thou pitch, etc)
        description ( ≤ 50  characters)
        position entered (0=no 1=yes)
        x position
        y position
        orientation (0,1,2 or 3)
        signal names (one on each line for each pin)
```

The variant code cand be used for purposes other than that of distinguishing between dilics with different pitches. For example, it is used to differentiate single pads of different diameters (*dev9.1.n*)

The default values for the module types are described in Appendix C, page 47: Other module types can be user defined (see Section 5.4, page 17).

## 2.9 Notes

- If no *outlines* subdirectory (see Section 5.4, page 17) is found in the current directory, a warning message is displayed. Don't worry, the *outlines* directory is only required if when user-defined modules are being used.
- The board size and offets generated for the default boards are slightly different (for historical reasons) from those generated by the **vtoncap** utility (see Section 3, page 10).
- *ncap* files generated from a Viewlogic schematic (see Section 3, page 10) may generate signal names containg non-standard characters. Use of the **m** command to inspect modules containg these signal names will give rise to a warning message. This warning can safely be ignored.

# 3 Schematic Interface

## 3.1 Introduction

Schematics drawn using the Viewdraw package can be translated into **ncap** format by means of the program **vtoncap**. An alias file, situated in the directory pointed to by the OUTLINE_LIBS (see Section 5.4, page 17) environment variable should be created to match the package type attribute of Viewdraw to the device type identifiers used by the netlist extraction program, *netex*.

## 3.2 Generating ncap files from Viewdraw Diagrams

First copy the file *pcb/design_tools/lib/vdraw/pcb.cfg* to the working directory.

```
pcbfwd  <design>
vtoncap <design>
```

First the Viewdraw schematic is converted to a netlist by the **Viewlogic** utility *pcbfwd*. This creates files *design.net, design.pkg, design.dvf, design.fdc* and *design.ndc*. Only the first two files are significant. Subsequently the netlist and package information is converted into MUCS **ncap** format by **vtoncap**. Note that **pcbfwd** backannotates the viewlogic design file by allocating any unassigned package reference designator attributes (see below) and resolving any pin conflicts in devices with multiple component (slots) in a package.

## 3.3 Running vtoncap

The program prompts for a board template. A choice of Single, Double, Triple Eurocards or a user specified board is offered. If a Eurocard board is selected, the user may specify an standard size or extended size board. If a custom board is selected, the user is prompted for the x & y dimensions of the board in either mm, inches or thou; Further, an origin offset may be chosen; this may be required in order that components such as edge connector appear in board co-ordinates on grid. Single Eurocard boards will be rotated for more economical manufacture by the MUCS PCB unit.

## 3.4 Viewdraw Attributes

Any user-defined Viewdraw symbol representing a physical part to be placed on the Printed Circuit Board must have certain Viewdraw attributes defined.

**LEVEL=STD** This attribute tells the *pcbfwd* netlister not to search for any underlying schematic and that the symbol corresponds to a physical PCB component.

**REFDES** This is the package reference designator. It defines a unique name to reference the component on the board. *REFDES* consists of a prefix followed by a number or a "?". For example, REFDES=R10 might identify a resistor in the design.

A "?" following the prefix (e.g. REFDES=IC?) indicates that the *REFDES* attribute is unassigned. All symbol definitions should use unassigned REFDES attributes. Users can choose whether to set the REFDES attribute or leave it unassigned when instantiating a symbol on a schematic. The *pcbfwd* program will allocate unassigned REFDES attributes and back-annotate the schematic (see Section 3.7, page 11)

**SIGNAL** The SIGNAL attribute is used by Viewdraw to indicate implicit pin connections, typically power pins, which are not shown on the symbol drawing. The atribute value is the signal name terminated by a semicolon followed by a comma separated list of pin numbers:
SIGNAL=VDD;14
SIGNAL=GND;7,1

**DEVICE** This attribute is optional. If set, it will be included in the *ncap* device description field, otherwise the component name will be used.

**PKG_TYPE** This attribute specifies a name allowing PCB software tools to determine the package outline. Viewdraw defines a limited set of the attributes, for example PKG_TYPE=16PDIP is used for a standard 16 pin Dual-in-line package. Section 3.6 shows how user-defined PKG_TYPE attributes link to the MUCS system.

**PARTS** If a package has contains several identical components (for example, a 74LS00 contains 4 NAND gates), the *PARTS* attribute should be defined in the component symbol and should be set to the number of components in a package. The allocation of pin numbers is determined by the # attribute described below.

**#** The # attribute is the pin number attribute. Each pin on every Viewdraw symbol must have a pin number attribute. Normally, each pin on a symbol has a single pin number. However, a component which is replicated several times in a single package should have a list of pin numbers. For example, the output pin on the NAND gate symbol for a 74LS00 has the pin number attribute: #=3,6,8,11
The pin number attributes are attached to each pin in the symbol definition, not to the symbol itself. The number of items in the pin number list must correspond to number of parts declared by the PARTS attribute. When a symbol with a PARTS attribute $> 1$ is instantiated on a schematic, the next available set of pins from the pin number list is chosen. The chosen allocation may be over-ridden by the *change slot* command in Viewdraw

**NC** g This attribute identifies unconnected external package pins that are not displayed on the symbol. The attribute should be a comma separated list of pin numbers. A pin number should not appear in both a **NC** attribute and a **SIGNAL** attribute.

## 3.5 Power Supply Names & Global Signals

Viewdraw uses VDD & GND to denote power connections whereas the MUCS system uses VCC and GND. The program *vtoncap* translates VDD references to VCC. VCC may used on the Viewdraw schematic; however, users should be aware that VDD is by default defined to be a global signal, whereas VCC is not. Thus if a hierarchical schematic is used, the signal VCC will be be treated as a local signal lower down the hierarchy (it will actually be renamed to reflect the hierarchy). VCC may be given global scope by including the following line in the *viewdraw.ini* file:
**NETNAME VCC**

## 3.6 Alias File

The translation between Viewlogic Library Components and the MUCS design system library is by means of an alias file "v_alias". An example of the file is given below

```
# This is a comment, placed any where in the file
# The next entries are 14 and 16 pin Dilics
14PDIP    dev1.14.0
16PDIP    dev1.16.0
```

Comments in the alias file are introduced by # or " or ;
The first field is the Viewdraw attribute PKG_TYPE for the component. The last field is the device in the MUCS library to which the Viewdraw component corresponds. For more information see Section 5.4, page 17 and Appendix C, page 47.

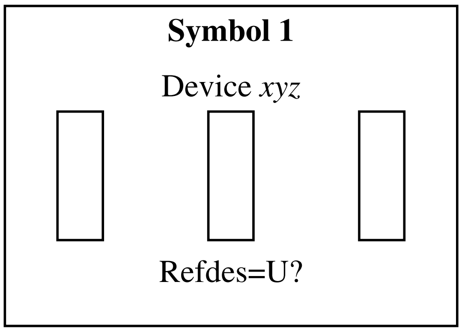
An alias file may exist in every directory specified by the *OUTLINE_LIBS* environment variable. Package types may be multiply defined; the earliest found will be the version used. The search order for "v_alias" files is

1. First directory specified by *OUTLINE_LIBS*
2. Subsequent directories specified by *OUTLINE_LIBS*
3. The current directory
4. The default library *design_tools/lib/outlines* in the *pcb* directory

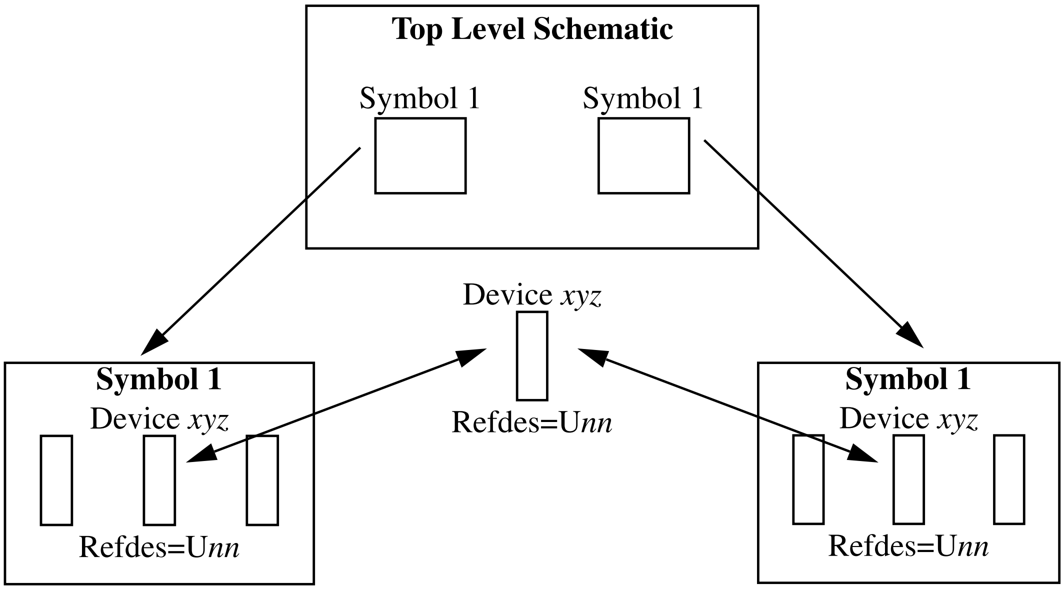
## 3.7 Setting Refdes in Hierarchical Schematics

### The Problem

Unless special action is taken, different instances of a physical device in a hierarchy will end up possessing the same reference designator (Refdes). This is because Viewdraw distinguishes between "instances" and "occurences". The problem is illustrated in Figure 1.




A level in the hierarchy, *symbol 1*, is defined containing a physical part, *device xyz*.. This component is then used several times in the top-level design:




Viewdraw uses a single instance to represent *device xyz* in both cases. Setting **Refdes**, either by *pcbfwd* or manually, in one will cause the other to display the same **Refdes**. Occurrence attributes (OATS) must be used to override component attributes.

Figure 1: Reference Designators in a Hierarchy

**The solution**

Occurence attributes in the schematic must be turned on. Select **Props** ⇒ **Modes** ⇒ **Oats** ⇒ **On**. For this change to be permanent, the *viewdraw.ini* file must have the a line starting: **—5.0 OATS**

## 3.8 Notes

- The default v_alias file maps chooses outlines **without** connections for a decoupling capacitor above pin 1 and the standard Vcc pin. You shoud redefine the file if this is not what is required.
- The Viewdraw REF_DES attribute is included in the *ncap* module description field.
- The placement of devices in an existing *ncap* file are respected are far as is possible
- The default board sizes and offets differ slightly from *ncap* (see Section 2, page 6) generated boards. Furthermore, *ncap* generated single Eurocard boards are not rotated.
- See pcb/examples/timu for an example of interfacing Viewdraw to the MUCS system.

# 4 Place - Interactive Device Placement

This section describes the use of a graphical *pcb* utility for interactively adding device placement information to an ncap datafile. This runs under XView using Xlib graphics calls for drawing. **Place** is invoked as follows:

**place** *[-display name][-fg colour][-bg colour][-hg colour] design_name*

where *design_name* is the **ncap** datafile. Any errors found are monitored on the standard error output. The options are:

**-display** *name*
: Allows a specific display to be specified. Defaults to the one being typed on.

**-fg** *colour*
: Specifies the required foreground colour - defaults to black.

**-bg** *colour*
: Specifies the required background colour - defaults to the background colour in use prior to **place** being called.

**-hg** *colour*
: Specifies the required highlight colour for indicating connections. Defaults to red.

The text in this section is also available in the help window of **place**.

## 4.1 The Display

The main display consists of two subwindows. The lower one consists of a display representing the board area. This is initially empty except for the board outline. Above the board area is a control panel which contains various mouse buttons, which may be activated with the left mouse button, and a number of textual items. It also contains a representation of the mouse buttons with their currently defined functions.

There are also two other independent windows which may be selectively displayed as required. The first of these is the help window - activated by the *help* button in the control panel or by pressing *h* or *?*. To scroll through the text use the scroll bar on the left hand side. To remove the window from the display, the *done* option has to be selected from the frame menu. The next time the help option is selected, the window will pop up unchanged and in its last position. The second pop-up window contains the devices which have not been positioned on the board and may be displayed using the *unplaced* button.

## 4.2 Moving Devices

Devices may be dragged from one position to another using the cursor. Point at a device outline using the cursor. It may be necessary to pop up the unplaced window if **place** is being used for the first time on a datafile. A description of the device and its position is displayed in the control panel. Pressing the left hand mouse button while pointing to a device causes the outline of the device to be replaced by a rectangle containing the device. The rectangle follows the cursor when it moves. The position of the device is dynamically updated in the control panel. Devices may be moved between the main display and the *unplaced* subwindow by simply dragging them to the appropriate window. The user may venture outside the windows if necessary but the device will not be displayed there. To place the device, simply release the mouse button. When a device is dragged from one window to another, it is not visible in the new window until the mouse button is released. This is due to a limitation of the window system. Note also that the devices are usually drawn smaller in the unplaced window than they appear on the board display.

## 4.3 Group Moves

It is also possible to move a group of devices at the same time. Pointing at a device and pressing the right mouse button selects the device (distinguished from unselected devices by its inverted label). Any number of devices may be selected in this manner. Moving a selected device (using the left mouse button) with the shift key pressed causes all selected devices to move together. To de-select a device, press the shift key and the right hand mouse button while pointing at the relevant device. All the devices in the current window may be de-selected by pressing the *z* key.

## 4.4 Rotating Devices

Pointing at device and pressing the central mouse button rotates the device 90 degrees clockwise. If the shift key is pressed simultaneously, the direction of rotation is reversed.

## 4.5 Changing The Grid

To simplify positioning of devices, movement is normally constrained in fixed increments to a grid. The size of this grid is displayed in the control window. To change the grid size, the cursor first has to moved into the control panel. Point the cursor at the grid size label point and click the left hand mouse button. The grid size may now be edited by simply typing the grid size followed by hitting the RETURN key. A message will be displayed confirming the new grid size.

## 4.6 Writing and Quitting

To write the new placement to the datafile, the *write* button in the control panel may be used. The user may edit the filename first before saving any changes. Before exiting, **place** will ask for confirmation but will not ensure that any changes have been written.

## 4.7 Showing Connections

Connectivity between devices may be displayed by means of the keyboard command *l* which toggles between its active and inactive states. When active, pointing at a device will highlight in a naive manner, the connections to that device.

## 4.8 Keyboard Commands

There are a number of keyboard commands that may be used whilst the cursor is in the main window or in the *unplaced* window. These are:

**q** quit

**w** write datafile

**h** display help window

**?** display help window

**z** de-select all devices in the window

**l** show connections

**r** redraw window

## 4.9 Default Positions for Edge Connectors

Eurocard indirect edge connectors may need rotating and must be aligned at their correct position on the board. Indirect edge connectors for Eurocard boards should be positioned at: (X=3600, Y=200), (X=8850, Y=200), (X=14100, Y=200)
Rotated single Eurocard boards, such has produced by **vtoncap**, (see Section 3, page 10) should have the connector positioned at (X=200, Y=500).

# 5 Netex - Creates the inter-connection specification

**Netex** checks an **ncap** datafile for errors and produces a set of boards design specification files that may be used for tracking and plotting. **netex** is invoked as follows:

**netex** *-[fslrxodq] design_name*

where *design_name* is the **ncap** datafile. Any errors found are monitored on the standard error output. Two directories are produced by **netex**, *design_name.ndir* which contains files which specify inter-connection data used for tracking and later board manufacture and *design_name.ddir* which holds UNIX plot format files describing the layout of pads and tracks on each layer of the board. These are described below.

**design_name.ndir**

| | |
|---|---|
| trackeript: | Tracker input file - the current design layout. |
| net_file: | Net lists identifying interconnected pins. |
| sig_no: | Net numbers for each signal name. |

The syntax of the trackeript file is outlined in Appendix A, page 38. This file will is modified. during the tracking phase before manufacture. Generation of this file may be suppressed by the -x option. In producing the specification, **netex** ignores any unplaced devices or isolated signal names which it regards as being unconnected.

**design_name.ddir**

| | |
|---|---|
| outlines: | outlines of the board and modules. |
| names: | strings to identify the modules corresponding to each outline. |
| pads.l*n*: | pads which exist on layer it n. |
| lines.l*n*: | tracks and vias which exist on layer *n*. |

These files are in UNIX plot format and may be displayed by the plotter driver *hp_plot*, by suitable laserwriters (*lpr -g*) and on the screen by the xplot command. These files can be suppressed by the -o and -d options described below. **Netex** creates the pad/lines files by internally calling the **unixplot** utility described later in this document in Section 9.1, page 36) . It will be used independently as required during the tracking phase of the design.
Various options can also be specified with **netex** which include:

| | |
|---|---|
| **-f** | force generation of a new *trackeript* file even if one already exists. the design. |
| **-s** | accept input from the standard input and not the **ncap** datafile. |
| **-l** | generate parts list on standard output. |
| **-r** | generate list of device types on standard output. |
| **-x** | suppress generation of *trackeript* file. |
| **-o** | suppress generation of device outline unix plot files. |
| **-d** | suppress generation of pad/layer unix plot files. |
| **-q** | quit if errors occur otherwise continue. |

**Note that if the *-f* option is specified, then the results of any previous routing attempts will be destroyed.**

## 5.1 Specifying Design Rules

When producing the tracker input file (*trackeript*), **netex** makes reference to a design rule file which contains details of various feature sizes and clearances to be used during tracking. If no other external action is taken, a default design rule file is used - but alternatively the user may create a file for the particular design. By convention, this should be called *design_name.rules*, where *design_name* is the name of the **ncap** file and should be placed in the same directory. The contents of the default file is shown below:

```
grid:       25;
clearance:  10 10 10;
track:      15;
via:        60 ;
margin:     100 50;
```

The design rule file consists of a number of keywords followed by a colon and values in **thou**. The first line specifies the grid size to be used by the router. The second line specifies the minimum clearances to be allowed for tracks, pads, and vias[1]. The third line specifies the track width to be used. The fourth line specifies the via size to be used. This size field may optionally be followed by the keyword **buried** which specifies that buried vias should be used in multi-layer boards as opposed to the default through-board vias. Do not specify buried vias unless you are really sure that that is what you want - Appendix F.3, page 54 discusses the use of buried vias. Finally the margins around the the board are specified to dictate firstly how close vias (and secondly tracks) may be placed to the edge during tracking. Each layer will finally have a border drawn round it for manufacturing purposes. This is 100 thou wide and for tracking layers is drawn around the outside of the edge. The edge is defined by the board limits in xy space specified at the start of the **ncap** datafile.

## 5.2 Choosing Design Rules

As noted above, different design rules can be specified. Additionally, selection of different device outlines (see Section 5.4, page 17) allows varying component pad sizes to be used. In general, the larger the feature size, the easier it is to manufacture the resulting PCB; the smaller the feature size and grid size, the more routable the board is – however tracking time becomes longer and the probability of producing defective boards becomes larger. Inspection of the detailed tables of clearances for various combinations of design rules, pad sizes and grid sizes given in Appendix G reveals that it is easy to choose inappropriate combinations that result in the worst of all worlds. The options listed below are suitable for manufacturing boards in the MUCS PCB facility. Note that for the Hi-density and Super Hi-density options, the *OUTLINE_LIBS* variable (see Section 5.4, page 17) should include the *pcb/design_tools/lib/outlines.20* directory to pick up the library components with the appropriate pad size.

| | | Normal Rules | | Hi-Density | | Super Hi-Density | |
|---|---|---|---|---|---|---|---|
| | | A | B | C | D | E | F |
| Grid | | 25 | 25 | 20 | 10 | 20 | 10 |
| Clearance | | 10 | 10 | 7.5 | 10 | 7.5 | 7.5 |
| Track Width | | 15 | 15 | 10 | 10 | 10 | 10 |
| Normal Components | Pad | 60 | 60 | 50 | 50 | 50 | 50 |
| | Drill (mm) | 1.0 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 |
| | Solder Mask | 65 | 65 | 55 | 55 | 55 | 55 |
| Edge Connector | Pad | 60 | 62.5 | 55 | 55 | 55 | 55 |
| | Drill (mm) | 1.0 | 1.0 | 0.9 | 0.9 | 0.9 | 0.9 |
| | Solder Mask | 65 | 65 | 60 | 60 | 60 | 60 |
| Via | Pad | 60 | 60 | 50 | 50 | 32 | 32 |
| | Drill (mm) | 1.0 | 0.8 | 0.8 | 0.8 | 0.35 | 0.35 |

Table 1: Recommended Design Rules (thou)

**A** The rules are suitable for use with the standard outlines library (see Section 5.4, page 17). The 1.0mm holes in the drilled in the component pads means that that assembly tends to be tedious because sockets are a loose fit and fall out of the board when it is turned upside down to solder the components.

**B** These rules are essentially the same as those in [A]. They are included to show what is required if a smaller drill size is to be used for normal components (so they don't fall out when the board is turned upside down for soldering!!). Indirect edge connectors do not fit a 0.8mm hole, so they must have a large drill size specified. Normally, the easiest way of specifying drill szies to a PCB manufacturer is to associate each different drill size with a different pad size. Thus the pad size for edge connector here is increased to 62.5 thou. Obviously, there are several ways of achieving the same effect.

**C** These rules are suitable for difficult to route boards. Two tracks can be squeezed between the pins of a standard IC. Two tracks can also run betwen the pins of an on-grid Eurocard connector. However, in a 6U (double height) or 9U (triple height board), the 2nd connector is positioned off-grid. For this connector, only 1 track will be allowed between the pins (see table 33, page 59. The reduced clearances (around the 55 thou edge

[1]The pad-track actual clearance are the maximum of the specified pad and track clearances. Similarly, the via-track clearance used is the maximum of the specified via and track clearances

connector pads) may reduce board yield, but note that the actual clearance around the 50 thou pads will be 10 thou.

**D** These rules are similar to [C], except the clearances have been have been increased to 10 thou everywhere. In order to get **any** tracks between the 55 thou pads on a edge connector, the grid size has to be reduced to 10 thou leading to longer tracking times. All edge connectors are on-grid and will have a maximum of 1 track bewteen pins. If 2 tracks between all edge connector are required, use a 10 thou grid and 7.5 thou clearances.

**E** The use of small vias gives more routing area on the board. However, the reduced clearances may give rise to problems during board manufacture.

**F** These rules are similar to [E]. Reducing the grid size to 10 thou allows closer diagonal packing of tracks and vias to other vias. The tracking time will be increased

Rule files matching the set of design rules described above can be found in *pcb/design_tools/lib/boards*

## 5.3 Board Layer Biasing

As with the design rules, **netex** references a file to determine the nature of each board layer. The default file is *design_rules/lib/boards/bias_n* in the *pcb* directory where *n* is the number of layers. For example the file for four layers may contain:

```
horizontal
"VCC"
"GND"
vertical
```

Each file has a line for each layer from 1 to *n*. The keywords *horizontal* and *vertical* specify a tracking layer bias. If no bias is required the keyword *unbiased* should be specified. *"VCC"* and *"GND"* specify a power layer. Again it is possible to override the default behaviour by creating a file *design_name.bias* with the desired format in the users current directory.

## 5.4 Specifying Different Device Outlines

A standard library of device exists which decribes the device shape, and the size and position of pads corresponding to pins on the device. It is possible to cause **ncap** and **netex** to use a personalised set of device footprints in addition to, or instead of, those used by default. This will be necessary if components are used which are not in the standard library or if an alternate library is used because smaller pads required for tighter design rules (see section 5.2, page 16). The default library is to be found in *pcb/design_tools/lib/outlines* (The current directory is also searched). A library suitable for routing with the higher-density design rules is to found in *pcb/design_tools/lib/outlines.20* and a further library containing less commonly used components is found in *pcb/design_tools/lib/outlines.xtra*. A description of the devices in the libraries can be found in Appendix C, page 47.

The libraries to be searched by **ncap** and **netex** are specified by the optional enivronment variable OUTLINE_LIBS. When the characteristics of a particular device are required, the set of directories specified by OUTLINE_LIBS[2] will be searched in sequence until the relevant device description file is located. So for example, issuing the commands to sh(1) or ksh(1)

```
OUTLINE_LIBS='/usr/fred/my_devs /usr/fred/my_devs2'
export OUTLINE_LIBS
```

will cause subsequent invocations of **netex** and **ncap** to search first */usr/fred/my_devs*, followed by */usr/fred/my_devs2*, then the current directory and finally the default directory *design_tools/lib/outlines* in the *pcb* directory. If a device is multiply defined, the first device file to be found in the search path will be used.

[2] if more than one directory is specified, the directories are *space* separated and thus the list will need to be quoted

## 5.5 Generating Customized Device Outlines

If you use devices not in the supplied libraries, you will need to know how to specifiy your own device outlines. Associated with each device type, e.g dilic, there is a file which describes global characteristics - such as its name - and provides a list of all the individual examples of this particular device type. This list is a tabulation of all the variations in number of pins and pin pitch for this device type used in the design. For each of these individual devices there is an additional file which describes the different pad shapes across the various board layers, pad positions and other information particular to that instance.

When creating a new device type the user must first give it a device reference number (*dev_numb*) which to avoid clashing with the default library should be greater than 1000. Then, having created a library directory, and referenced it via OUTLINE_LIBS, make a tabulation file within it, **tab.***dev_numb*, listing the different variations of this type used in the design. As an example, here is the contents of *tab.0* from the default library which corresponds to a dilic with integral pads for a decoupling capacitor which are in line with the dilic pins, 100 thou from pins 1 and 8.

```
dilic
pitch
 300 thou
 600 thou
 900 thou
!
 8.0
 10.0
 14.0
 16.0
 18.0
 20.0
 22.0
 24.0
 24.1
 28.1
 40.1
 64.2
```

The first line specifies the name of the device type. This is used, for example, as a prompt by **ncap**. It is assumed that different individual devices will be differentiated by the number of pins. They may be further distinguished by another characteristic which is the *pitch* in this case. The *pitch* option if used, should be specified on the second line followed by the various pitch values available. The *pitch* option is terminated by a line containing a single *!*. If the devices are not to be referenced in this way this line should immediately follow the device name. Finally there is a list of the individual devices until the end of file. These consist of the number of pins followed by a decimal point followed by the pitch option number starting from 0. (Even if there is no pitch option, it should be followed by a zero). For example in the above file the first such line corresponds to a dilic with 8 pins and a 300 thou pitch.

For each device then create a file **dev***dev_numb.pins.pitch_option* in the same directory. Here for example is the file **dev**0.8.0 describing an 8 pin dilic with pads for a 300 thou pitch decoupling capacitor 100 thou from pins 1 and 8.

```
paddef:
tracking  circle 60
power  annulus 80
end
pins
     0,0
     0,100
     0,200
     0,300
     -300,300
     -300,200
     -300,100
```

```
        -300,0
    end
pad "VCC" -300, -100
pad "GND" 0, -100
        -330,-30,30,-30
        30,-30,30,330
        30,330,-330,330
        -330,330,-330,-30
        -330,-130,30,-130
        30,-130,30,-70
        30,-70,-330,-70
        -330,-70,-330,-130
```

The first section describes a pad shape. This begins with the key-word **paddef** followed by a colon and an optional pad name. A pad name in this case is simply an arbitrary string identifier. On successive lines there is then a layer type (one of **tracking**, **component**, **solder**, **inner** or **power**),[3] a pad shape (**circle**, **square**, **rectangle**, or **annulus**) and the pad dimension(s) in thou. If the pad shape is a **rectangle** then the pad dimensions should be separated by a comma, with the width of the pad defined first, followed by the height of the pad. Each pad definition ends with the key-word **end**. There may be several pad defintions.

Following these definitions is the list of pin coordinates starting from pin 1. The convention has been adopted that the coordinates of pin 1 should be 0,0 and wherever possible this pin should be at the bottom of the device in the default orientation. Where just the coordinates appear on the line, the default pad type, i.e that defined without a name (as in the example above) is assumed. Alternatively the coordinates may be followed by a colon and a previously defined pad name e.g

```
paddef:P0
tracking square 60
power annulus 80
end
pins
 0,0:P0
```

or a fully expanded definition such as

```
 0,0:
tracking square 60
power annulus 80
end
```

The next two lines define the two pads which correspond to the decoupling capacitor associated with the dilic. **netex** arranges for these pads to connected to the correct power supply – it is not necessary, or indeed possible, to do this at the **ncap** or schematic level.

The last section of the file - following the end of the pin list - is a list of vectors which define the outline of the device. These should define one or more closed shapes which enclose all of the device and its pads. Note that since device has an associated decoupling capacitor, this is shown in the outline – this also serves to identify which end of the outline corresponds to pin 1. In general, avoid the use of anonymous rectangular boxes for specifying the outline; add a distinctive shape so that the orientation of the component can be easily determined from a plot of the layout or when using **place**. The specifications for devices dev1.n.*, for example, contain a marker on the edge of the device that runs from pin 1 to pin n.

A special utility for specifying pin-grid-array devices quickly is described in Section 7.8, page 26.

It would be useful to other users of this software, if designers who have created their own device libraries, would email them to `pcb@uk.man.cs` so that they can be incorporated into future releases of this package.

[3] The component layer is layer 1, the solder layer is the bottom layer. A surface-mount device will typically have pads on just the component layer. An "inner" layer refers to an inner layer on a multilayer board on which tracks will be placed whereas "power" refers to an inner layer on a multilayer board which is used as a power plane and on which no tracks will be placed. Thus tracking = component + solder + inner. Further details may be found in Appendix E.

# 6 Routing

## 6.1 Synopsis

**track [-adfFghi**<*num*> **mnostvx]** *design_name*

*track* is a shell script wrapper for driving the various routers. Normally, it runs *netex* prior to calling the specified routers(s). However, if the file *preplaced* is found in the directory *design.ndir*, this file will be used as input and *netex* is not run. See section 6.4, page 22 for more details. The options available are:

- `-a` Use a global tracker based on simulated-annealing.
- `-d` Produce plot files for failed connections and routed connections. By default only plot files for the failed connections are produced.
- `-f` Use a non-linear temperature decrement factor based on specific-heats for the annealed router.
- `-F` list the signal names of the failed connections.
- `-g` Use a greedy version of the global tracker.
- `-h` Use the hardware layer-pair tracker.
- `-i<num>` Specifes an optional iteration count for the annealed router. See the section on annealing for further details.
- `-m` Use a multilayer serial tracker for routing (the default).
- `-n` Use a modified layer-pair serial tracker.
- `-o` Use the original layer-pair serial tracker.
- `-p` preserve the output in a file *tracker-init* which can be used as an initial solution for the global routers in future invocations of *track*
- `-s` Set routing parameters suitable for routing switchboxes. In this mode vias are preferable to tracks going against the layer bias.
- `-t` Send monitoring output to the screen, rather than appending to a file.
- `-v` Report periodically on the progress of the tracker.
- `-x` Ignore any tracker-init file that may exist: the global routers will start afresh rather than using an initial solution

The syntax of the option specification follows that of *getopts[1]*, thus options may be cascaded or specified individually. The following are valid examples of calling the tracker.

**track -mv** *design_name*
: Calls the default tracker with reporting information

**track -v -m** *design_name*
: As above

**track -nai2 -v** *design_name*
: Use the layerpair tracker followed by the annealed global router with an iteration count of 2. The reporting flag (-v) applies to both phases of routing.

**track -a -n -vi 2** *design_name*
: As above

Only one serial router (`-nmo`) may be specified, but a serial router may be cascaded with the global routers as described later. The switchbox option (`-s`) is only available with the global and multilayer trackers. If none of the available routers are specified on the command line then the default router used is the multilayer tracker (the `-m` option).

After the routing is finished, `track` updates the plotter files created by **netex**. Unless the `-d` option is specified, the default action is to only produce a plot file for the failed connections.

## 6.2 Serial Routers

### 6.2.1 The Layer-Pair Router

The original software tracker written was developed as a basis for a hardware implementation of the tracker. It implements a *layer-pair* algorithm. The layer-pair algorithm uses a pair of layers with opposite biases, and then tries to route as many connections as possible on the pair of layers selected. Any failures are then routed on another layer pair and so on until all layers have been routed on, or there are no failed connections. The original software tracker can be specified using the `-o` option and the hardware equivalent, by the `-h` option. Currently the hardware router is not available for general use. Further details may be found in **[Spie83], [Spie85], [Spie87]**

### 6.2.2 The Modified Layer-Pair Router

This tracker is essentially the same as the previous tracker but with some bugs corrected and minor changes in the algorithm. The *layer-pair* structure is retained. The results from this tracker are usually better than that obtained from the original tracker, although improvements are minor. This tracker may also be specified by the `-n` option.

### 6.2.3 The True Multilayer Router

The third software tracker available is a true multilayer tracker in which all connections are routed on all the available layers simultaneously (as opposed to routing only on pairs of layers). This tracker may be specified by the `-m option`. For more than 2 routing layers, this router distributes tracks more evenly between the available routing layers than the *layer-pair* trackers, and the results are generally better. It is the default tracker.

## 6.3 Global Routers

### 6.3.1 Introduction

A global router has been implemented that uses the technique of simulated annealing to perform ripup and rerouting operations. This is a useful facility because it allows the modification of existing routes to make room for unrouted connections. A true multilayer routing algorithm is used to route all the connections. This tracker may be specified by the `-a` option. A separate version has been implemented that basically employs the same ripup and reroute operations but restricts the size of the search space in an effort to converge to a final solution more quickly. The latter choice can be specified using the `-g` option.

Critical to the annealing process is the way the initial configuration is determined. By default, the annealed tracker determines an appropriate starting solution internally. The user can overide this default by specifying another initial tracker on the command line. This initial tracker is then used to determine the starting configuration for the annealed router.

A characteristic of the global routers are their non-deterministic nature. **Rerouting a board with the same annealing parameters and board layout may not give the same results as previously obtained**. It may be possible therefore to obtain a better result by re-running the annealed router on the same data.

For performance reasons, buried-vias are not allowed when using the global routers.

### 6.3.2 Operation of the Global Routers

The global routers work by considering all the connections to be routed independently so that initially all connections tend cross each other. They then repeatedly rip up connections and reroute them in order to eliminate these conflicts. However, existing physical tracks defined in the *trackeript* file **are not ripped up**, thus sensitive preplaced tracks will be respected.

One of the factors dictating the amount of CPU time the annealed tracker uses is the initial number of crossovers. One way of reducing the initial number of crossings, is to use an initial routing configuration obtained from another (faster) tracker. Thus any or all of the serial router, the greedy global router and the annealed router may be combined for example: `track -a -m` or `track -mag`. Note that the order in which the flags are specified is not important. When the routers are cascaded, the serial router is run first (if specified) followed by the greedy router (if specified) followed by the annealed router (if specified).

It is possible to repeatedly re-anneal a solution by specifying an iteration count to the global routers (e.g `-a -i3`) which will run the annealed tracker the specified number of times, each invocation using the solution from the previous iteration as a starting configuration. It is possible to combine an iteration count with an initial tracker to generate a good starting configuration e.g: `track -a -i3 -m`

It is also possible to run the global routers from a previously computed partial solution. An initial solution, computed from a previous routing attempt, should have been generated using the *-p* option in a previous run. See section 6.4, page 22 for more details of the file manipulation that is carried on behind the scenes by *track*

### 6.3.3 Forcing a solution

Normally a SIGINT signal (CTRL C or kill -2) terminates the tracker. However, if either of the global routers is in the conflict resolution phase, the signal is trapped and the current solution is written out. This is useful when the router is close to an optimal solution, but is spending a lot of time making little or no progress.

Note that SIGINT is ignored by processes running asynchronously, so that if `track` is run as a background job, it must first be brought to the foreground. Alternatively, since the `track` script spawns the individual trackers synchronously, `ps(1)` may be used to discover the *pid* of the router and a kill -2 signal sent to that process.

### 6.3.4 Choice of Annealing Parameters

There are three important parameters that have to be specified to each invocation of the annealed tracker. These include the starting temperature, the temperature decrement factor and the termination factor. These should be specified on consecutive lines in a file called *design_name.ap* in the directory from which **track** is called. If an error is found in reading the startup file or if the startup file does not exist, the default file is used, *design_tools/lib/anneal/default_parameters* in the *pcb* directory. The default startup file is shown below:

10.00 0.95 5

The 3 numbers refer to the default Initial Temperature, the default Temperature Decrement Factor and the default Termination Factor.

For further details on how to choose the various parameters, users are asked to refer to **[Moos92]** for further details. If the number of ripups seems to continually increase, the initial start temperature is too high. An extreme case is illustrated in the *madarch* example.

Correct choice of the annealing parameters is critical to optimum performance of the router. The temperature controls the percentage of uphill moves which are accepted. If the start temperature is too high, the solution will continually get worse. Eventually, it will improve as the temperature drops, but much computation time will have been wasted. If the start temperature is too low, the annealed router approximates to the greedy router, and the solution may converge to a local minimum which is not the best obtainable. When running the annealed router, the user should monitor the solution to ensure that correct behaviour is occurring.

The temperature decrement factor (between 0 and 1) determines how rapidly the temperature reduces during the course of a solution search – the smaller the number, the faster the temperature declines. The final parameter, the termination factor, governs how long the router tries to discover a better solution than the current one.

The greedy router does not use the temperature directly, it always accepts the best solution. However, changing the default termination parameter will alter the length of time that the router will search for a solution.

## 6.4 Tracker Files

The tracker-input-format allows both logical routes (netlists) and existing physical tracks to be specified. The *track* script arranges for a file, *trackeript* to be the standard input to the various routers and to update this file with the resilts of the routing session. Existing physical tracks in *trackeript* are regarded as inviolate and will not be ripped up by the global routers. However, physical tracks in a second input file *tracker-init* are regarded as temporary and are eligible for ripip. The files are manipulated by *track* thus:

- if a file *design.ndir/preplaced* is present, it is copied to *trackeript* thereby preserving any preplaced tracks. Thus part of a board, for example critical routes such as clock distribution nets can be pre-routed by using either the tracker or the pcb editor and then moving the resulting *trackeript* file to *preplaced*
- if *design.ndir/preplaced* is not found, *trackeript* is gnerated by running netex on the *ncap* file. This will result in a file containing **only** a logical netlist.
- The file *tracker-init*, if it exists, is used as an initial solution to the global routers unless the *-x* flag is specified. *tracker-init* will have been generated by a previous run of the router using the *-p* flag solution is specified by the *-r* flag.

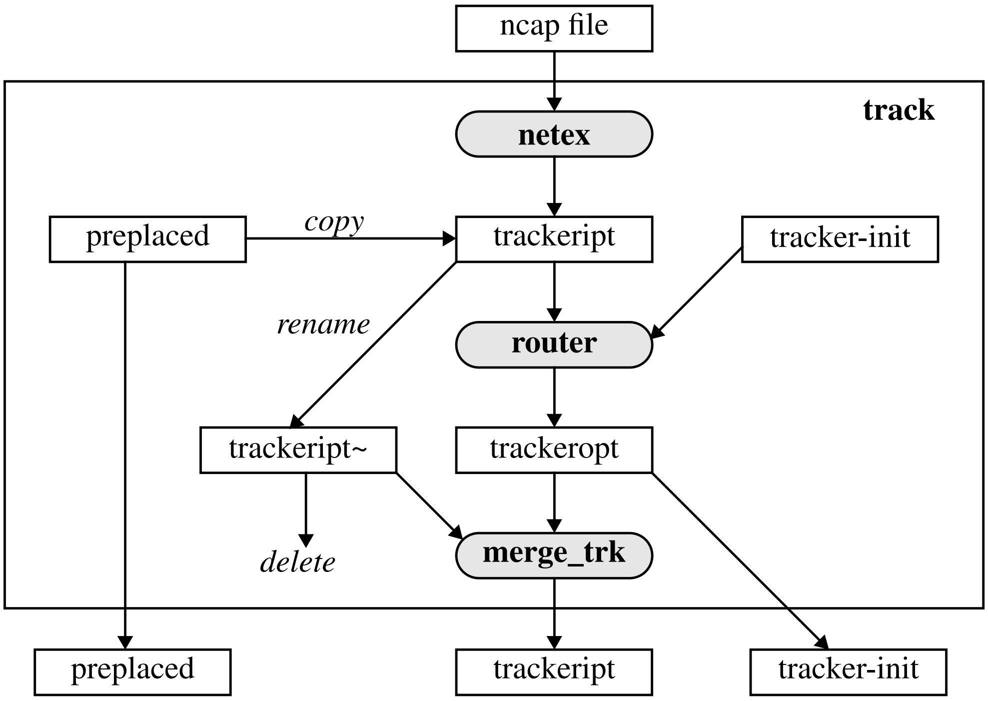



Figure 2: Files used by track

The relationship of the various filed used by *track* is shown in Figure 2.

If the various flavours of routers are cascaded, as described in the section on global routers, an intermediate solution is stored in the file *tracker-init*.

If the `-t` flag is specified, monitoring information is sent to the screen, otherwise it is appended in a file *design.mon*. Since the information is appended, this file will keep growing unless removed.

Various plot files are produced in the *design.ddir* directory. if the `-d` flag is specified, files for lines and pads for each layer are produced in Unix plot(5) format, otherwise only the failures are plotted.

It should be noted that for complex boards, the various tracker files will be large. Users may wish to delete any intermediate files produced after running `track`.

## 6.5 Choice Of Routers

The choice of router to use depends on several factors which are:

- The original tracker (`-o`) is obsolete and is only of interest to users of the hardware tracker. The modified layer-pair tracker fixes a number of minor bugs and usually produces slightly better results than the original layer-pair tracker.

- The annealed tracker produces better results than the other the trackers at the expense of CPU time. In difficult cases the annealed tracker may be up to 100 times slower than the quicker layer-pair and multi-layer routers. The extra computational effort invested may however be the difference between a design routed on 4 tracking layers and one routed on only 2 tracking layers.

- The user is recommended to use either of the layer-pair or multilayer trackers in the initial stages of the design process. This results in quicker track times though not necessarily the best solutions. If the solution is not acceptable, then the user can either alter the placement of various components using **place** and re-running **netex**, or use the annealed tracker to find a good solution for the given placement. If the annealed tracker fails to find a solution for a given design, this is usually a strong indication that more routing layers are needed.

- Experience with the annealed tracker indicates that boards that only have 2 tracking layers, or those boards using a 50 thou grid size, do not need an initial configuration to produce a quality solution quickly using annealing. For more complex boards (those needing more than 2 routing layers and less than 25 thou grid sizes) it is better to use the solution from the multilayer tracker as a starting configuration for the annealed router.

- Experiments with the annealed router seem to indicate that if a board is routable using the `-a` option on its own, then using the greedy option `-g`, also tends to converge to a zero failure solution, only quicker. The `-g` option is also useful in determining how routable a board is without spending too much time in the annealing process. The annealing option alone is most useful when the number of failures is to be reduced to an absolute minimum

- If the multilayer tracker produces a solution that has no failures, then other criteria such as vias and track-length may be optimised. Using the annealed tracker on the design at this point produces a solution with fewer vias and shorter overall track lengths. Another way of reducing the via count is to make the tracking grid size bigger. Note that this also affects the space available for routing (in a negative manner), so the user must be careful not to increase the grid size to much. Experience with the annealed tracker suggests that the increased CPU time used on boards for which a 100% routing is feasible is only a factor of 5. On the other hand, if the design is so complex that it is probably unroutable (by any means), the CPU time increase may be a factor of 50-100.

- The annealed tracker works well with a wide range of problems, from memory busses to random logic. It also gives comparable results to specialised switchbox and channel routers on the classical benchmarks for those problems.

- Because of the non-deterministic nature of the annealed tracker, if a solution with fewer than 5 failures is found, it may be worth rerunning the annealed router several times, perhaps with slightly different annealing parameters, to see if a zero-failure solution state can be discovered.

For an average treble eurocard the multilayer and layer-pair trackers usually require about 1 hour with the annealed tracker taking about 5-10 hours if the placement of the components is good. The CPU times mentioned are quoted for SUN Sparc1+ workstations and 6 layer boards. If the placement is poor or the number of layers specified is too few then the annealed tracker could take upto 3 days!.

Several examples illustrating many of the features discussed in this section are to be found in the examples supplied with the software.

## 6.6 Summary

The tracking software is complex and may seem intimidating at first. However, most users need only a few of the options. Here is a recommended procedure for first time users.

1. Place the board and use a serial router:
   `track` *design_name*

2. Replace and retrack if necessary

3. To remove the last few failures, use the global routers in conjuction with the serial router:
   `track -magvi2` *design_name*

# 7 Design alteration utility software

This section describes the utilities available for manipulation of the design at various stages of completion. Some of these are programs and others are scripts. Only a simple explanation of each is provided below. The first few allow the designer to alter his design tracker file (*trackeript*) to assist track layout during iterations of the tracking process. The later ones give assistance in device specification.

## 7.1 design-drc

This is a perl script which is a simple design rule checker for the *tracker-ipt* file which it expects on standard input.

## 7.2 design-select design_name

This perl script generates a tracker input file with selected nets only. The file *design_name.ndir/trackeript* is opened for input, *design_name.ndir/sig_no* is read to map names to numbers and *design_name.view* is read for a list of signal names to select.

## 7.3 design-filter

This perl script allows easy manipulation of the *trackeript* format. This file is copied from stdin to stdout with various changes as defined by the options:

| | |
|---|---|
| -gr *num* | set the grid size to *num* thou. |
| -pc *num* | set the pad clearance to *num* thou. |
| -tc *num* | set the track clearance to *num* thou. |
| -vc *num* | set the via clearance to *num* thou. |
| -vd *num* | set the via diameter to *num* thou. |
| -bv "y" or "n" | buried vias? y/n. |
| -tw *num* | set the track width to *num* thou. |
| -nt *num* | the number of tracks between 0.1" pins. |
| -sel *filename* | include selection records from *filename*. |
| -add *filename* | include route records from *filename*. |
| -ng *filename* | include nogo records from *filename*. |
| -rr | route rest of board after selection records. |
| -lb *num1 num2* | over-ride layer *num1* bias with *num2*. |
| -m | insert different cost param. record. |
| -ol | hack down to one layer board. |

The example "madarch" (Section H.4, page 63) illustrates the use of this facility.

## 7.4 design-swap layer1 layer2

Swaps tracks between layers. This script was written to allow the layers with the largest numbers of tracks to be placed on the outside of the PCB. Note that this swaps *only* track, layer and nogo records. Vias and pins are assumed to pass through all layers and pads are left where they were. The *trackeript* file is read from standard input and the altered version is written to standard output.

## 7.5 failed-signals design_name

This prints out the signal names in failed routes.

## 7.6 ncap-drc

This perl script is a simple design rule checker for an ncap file which it reads from standard input.

## 7.7 sel_gen [-m]

This is a perl script that allows a high level description of selection records for the *trackeript* file. Commands are read on the standard input at one per line. The results which are generated on standard output, contain the appropriate selection records to be placed at the head of the tracker file. There are four commands:

| | |
|---|---|
| ”route AddrBus00” | route this named signal. |
| ”route AddrBus00 AddrBus15” | route named signals in this range. |
| ”route_no 75” | route the route with this number. |
| ”route_no 80 90” | route the route numbers in this range. |
| ”inside 0 0 100 0 100 100 0 100” | route inside this polygon. |
| ”outside 0 0 100 0 100 100 0 100” | route outside this polygon. |

A polygon may have any number of co-ordinates. The ”#” comment convention is supported. Calling options are:

| | |
|---|---|
| -m *filename* | numbers to names mapping file. |

## 7.8 pgatodev - Create Pin Grid Array device files

Usage: **pgatodev** [-d] [-s *sdir*] [-h *header-filename*] -f *pga-filename*

Reads a graphical description of the Printed Circuit Board (PCB) footprint of a Pin Grid Array (PGA) chip and outputs a device description file for use with **ncap and netex**. The input *pga-filename* specifies the footprint in terms of filled or empty polygons, lines and points on an 0.1” grid. The device description is produced on the standard output. Note that the device is assumed to being viewed from above..

### 7.8.1 Options

**-d**

Causes the output file representing the device footprint to be in Unix plot format rather than the normal device description for use with **ncap and netex**. This appears on the standard output and can be viewed via the xplot command available in X-windows.

**-s** *[lrtb] [lrbt] [rltb] [rlbt] [tblr] [tbrl] [btlr] [btrl]*

Used to define the ordering of the pins in the device file. For example, if *sdir* is ‘tblr’ the PGA outline is scanned from top to bottom, progressing in a left to right direction. Thus the first pin to appear in the device file would be the one in the top left corner and the last pin would be the one in the bottom right corner of the outline. Defaults to ‘lrtb’.

**-h** *header-filename*

By default the device file starts with:

```
paddef:
tracking circle 50
power annulus 80
end
```

This may be overwritten by specifying an alternative header filename.

### 7.8.2 Input file format

The input file consists of a sequence of commands that set the fill mode and describe the shapes:

**fill**
: Set the current fill mode to ”fill”. In ”fill” mode, polygons, lines and points describe *pin* positions in the footprint.

**vcc_fill**
: As above, except that the pins being described are power pins.

**gnd_fill**
: As above, except that the pins being described are ground pins.

**clear**
: Set the current fill mode to ”clear”. In ”clear” mode, polygons, lines and points describe *empty* positions in the footprint.

**polygon***startpoint [point] endpoint* from *fillpoint*
: Define a polygon starting at *startpoint* with any number of vertices up to *endpoint* (*startpoint* and *endpoint* must be the same!). Polygon edges are restricted to horizontal, vertical or 45 degree angles. The polygon is filled with the current fill value using a flood fill algorithm starting from *fillpoint* which *must* be inside the defined polygon.

**lines** [*startpoint endpoint*]
: Define a line from *startpoint* to *endpoint*. Lines must be horizontal, vertical or at 45 degrees. Points on the line are filled with the current fill value.

**points** *apoint*
: Fill *apoint* with the current fill value.

Points are two numbers separated by the following characters: ”.()@,*space*” Furthermore, any of these characters may be used to enclose points, for example: (3,5),(6.7),6@3. The origin is interpreted as the bottom left of the device.

### 7.8.3 Tracker device files

**pgatodev** output appears on standard output as either a Unix plot representation of the required footprint where ”O” represents a pin (when the -d option is specified) or as the device file for the PGA in it's default orientation. The former should be given the filename *pga-filename*.**foot** and the latter the .**0** (pitch 0) version of the pga-filename.

The following is an example definition of an 17x17 PGA with 223 pins:

```
fill polygon 0@0 16@0 16@16 0@16 0@0 from 8@8
clear polygon 4@4 12@4 12@12 4@12 4@4 from 8@8
fill
lines 4@6 4@10
lines 12.6 12.10
lines 7,4 9,4 7,12 9,12
clear points (16,16) (0,0)
fill points (0@0)
```

which produces an xplot footprint as follows when the *-s rltb* option is specified:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 | $O^{1}$ | $O^{2}$ | $O^{3}$ | $O^{4}$ | $O^{5}$ | $O^{6}$ | $O^{7}$ | $O^{8}$ | $O^{9}$ | $O^{10}$ | $O^{11}$ | $O^{12}$ | $O^{13}$ | $O^{14}$ | $O^{15}$ | $O^{16}$ | |
| 15 | $O^{17}$ | $O^{18}$ | $O^{19}$ | $O^{20}$ | $O^{21}$ | $O^{22}$ | $O^{23}$ | $O^{24}$ | $O^{25}$ | $O^{26}$ | $O^{27}$ | $O^{28}$ | $O^{29}$ | $O^{30}$ | $O^{31}$ | $O^{32}$ | $O^{33}$ |
| 14 | $O^{34}$ | $O^{35}$ | $O^{36}$ | $O^{37}$ | $O^{38}$ | $O^{39}$ | $O^{40}$ | $O^{41}$ | $O^{42}$ | $O^{43}$ | $O^{44}$ | $O^{45}$ | $O^{46}$ | $O^{47}$ | $O^{48}$ | $O^{49}$ | $O^{50}$ |
| 13 | $O^{51}$ | $O^{52}$ | $O^{53}$ | $O^{54}$ | $O^{55}$ | $O^{56}$ | $O^{57}$ | $O^{58}$ | $O^{59}$ | $O^{60}$ | $O^{61}$ | $O^{62}$ | $O^{63}$ | $O^{64}$ | $O^{65}$ | $O^{66}$ | $O^{67}$ |
| 12 | $O^{68}$ | $O^{69}$ | $O^{70}$ | $O^{71}$ | | | | $O^{72}$ | $O^{73}$ | $O^{74}$ | | | | $O^{75}$ | $O^{76}$ | $O^{77}$ | $O^{78}$ |
| 11 | $O^{79}$ | $O^{80}$ | $O^{81}$ | $O^{82}$ | | | | | | | | | | $O^{83}$ | $O^{84}$ | $O^{85}$ | $O^{86}$ |
| 10 | $O^{87}$ | $O^{88}$ | $O^{89}$ | $O^{90}$ | $O^{91}$ | | | | | | | | $O^{92}$ | $O^{93}$ | $O^{94}$ | $O^{95}$ | $O^{96}$ |
| 9 | $O^{97}$ | $O^{98}$ | $O^{99}$ | $O^{100}$ | $O^{101}$ | | | | | | | | $O^{102}$ | $O^{103}$ | $O^{104}$ | $O^{105}$ | $O^{106}$ |
| 8 | $O^{107}$ | $O^{108}$ | $O^{109}$ | $O^{110}$ | $O^{111}$ | | | | | | | | $O^{112}$ | $O^{113}$ | $O^{114}$ | $O^{115}$ | $O^{116}$ |
| 7 | $O^{117}$ | $O^{118}$ | $O^{119}$ | $O^{120}$ | $O^{121}$ | | | | | | | | $O^{122}$ | $O^{123}$ | $O^{124}$ | $O^{125}$ | $O^{126}$ |
| 6 | $O^{127}$ | $O^{128}$ | $O^{129}$ | $O^{130}$ | $O^{131}$ | | | | | | | | $O^{132}$ | $O^{133}$ | $O^{134}$ | $O^{135}$ | $O^{136}$ |
| 5 | $O^{137}$ | $O^{138}$ | $O^{139}$ | $O^{140}$ | | | | | | | | | | $O^{141}$ | $O^{142}$ | $O^{143}$ | $O^{144}$ |
| 4 | $O^{145}$ | $O^{146}$ | $O^{147}$ | $O^{148}$ | | | | $O^{149}$ | $O^{150}$ | $O^{151}$ | | | | $O^{152}$ | $O^{153}$ | $O^{154}$ | $O^{155}$ |
| 3 | $O^{156}$ | $O^{157}$ | $O^{158}$ | $O^{159}$ | $O^{160}$ | $O^{161}$ | $O^{162}$ | $O^{163}$ | $O^{164}$ | $O^{165}$ | $O^{166}$ | $O^{167}$ | $O^{168}$ | $O^{169}$ | $O^{170}$ | $O^{171}$ | $O^{172}$ |
| 2 | $O^{173}$ | $O^{174}$ | $O^{175}$ | $O^{176}$ | $O^{177}$ | $O^{178}$ | $O^{179}$ | $O^{180}$ | $O^{181}$ | $O^{182}$ | $O^{183}$ | $O^{184}$ | $O^{185}$ | $O^{186}$ | $O^{187}$ | $O^{188}$ | $O^{189}$ |
| 1 | $O^{190}$ | $O^{191}$ | $O^{192}$ | $O^{193}$ | $O^{194}$ | $O^{195}$ | $O^{196}$ | $O^{197}$ | $O^{198}$ | $O^{199}$ | $O^{200}$ | $O^{201}$ | $O^{202}$ | $O^{203}$ | $O^{204}$ | $O^{205}$ | $O^{206}$ |
| 0 | $O^{207}$ | $O^{208}$ | $O^{209}$ | $O^{210}$ | $O^{211}$ | $O^{212}$ | $O^{213}$ | $O^{214}$ | $O^{215}$ | $O^{216}$ | $O^{217}$ | $O^{218}$ | $O^{219}$ | $O^{220}$ | $O^{221}$ | $O^{222}$ | $O^{223}$ |

# 8 Pcbed - Printed Circuit Board Editor

**Pcbed** is a monochrome graphics interface PCB editor driven via Sunview which allows a design to be edited or created from scratch. It is invoked as follows:

**pcbed** *design_name*

where *design_name* is optional and if specified can be an **ncap** datafile. A *design_name.ndir* sub-directory containing a *trackeript* file is expected when *design_name* is specified. When no *design_name* is specified, the user is starting a new design from scratch and the necessary file structure will be created automatically, resources permitting.

All board layers can be selectably viewed and edited through two conceptual layers, the Foreground and the Background. By default the Foregound is initially layer 1 and the Background layer 2, and each can subsequently be set to any desired layer number by the user. A keyhole viewing technique is used to display a small rectangular section of the full board at a suitable magnification. An adjacent window is maintained highlighting the proportions of this section relative to the full board size. The keyhole may be scaled up and down to allow more detail or greater viewing range as desired. It also has both manual and automatic movement control over the full range of the board. Pins, tracks and nogo areas may be created, selected, moved and deleted mainly under mouse control with some quick-key equivalents. No device library is available so **pcbed** is more readily applicable to incomplete or partially tracked designs.

## 8.1 The Display

The screen display is illustrated in Figure 2 and consists of a number of subwindows which are now described clockwise from the bottom left.

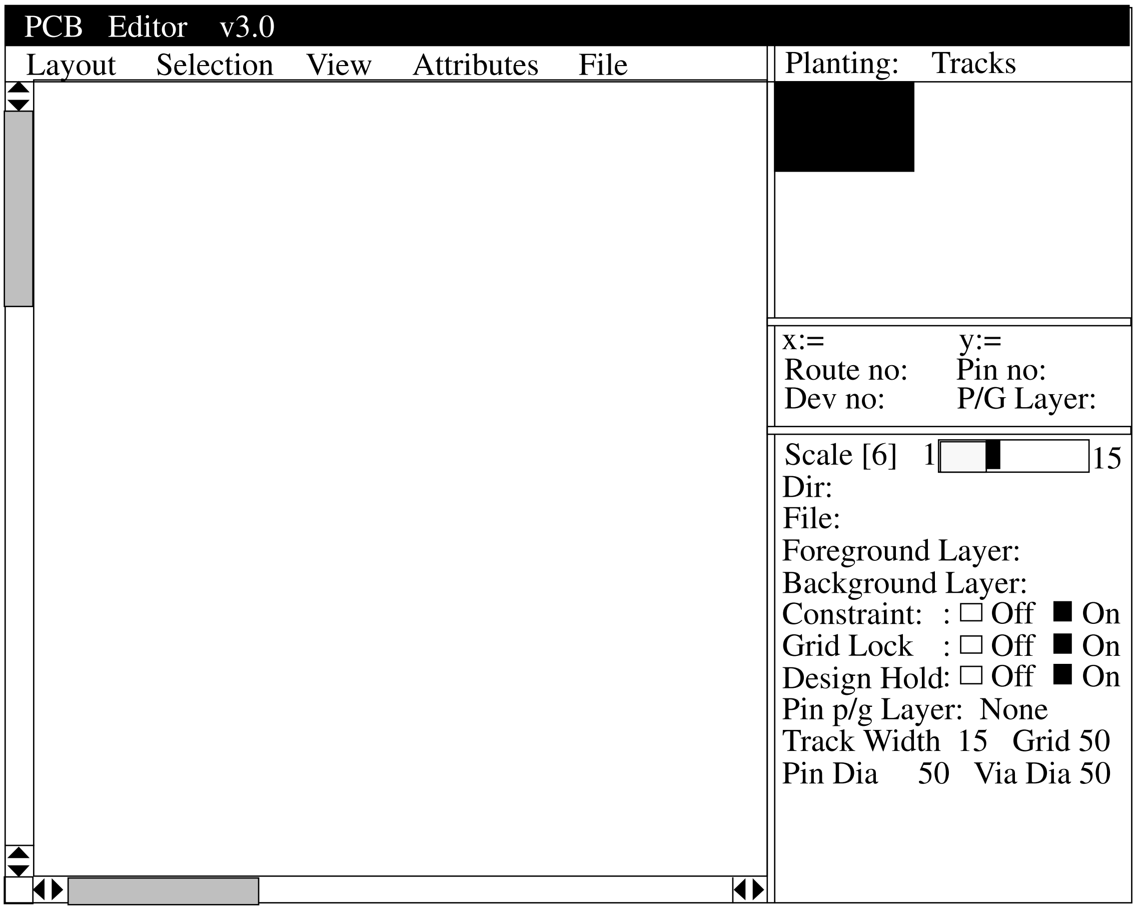



FIGURE 2 : THE SCREEN DISPLAY

- The large, approximately square, canvas is the main editing area. It is bordered by standard Sunview scrollbars which allow it to be moved around the board.
- The shallow panel area above it provides five different *pull-down* menus, **Layout, Selection, View, Attributes** and **File**. Each appears in a fixed position when a mouse button is pressed with the cursor in the vicinity of it's heading. Most of the pull-down items display a quick-key equivalent in brackets which will achieve the same effect as selecting the item using the mouse. Quick keys only work if pressed while the cursor is somewhere within the main editing canvas. Some of the **Attributes** menu items cause *pull-right* menus to be displayed which allow selection of, for example, track widths from a list of all available sizes. **Attributes** also contains a *Board Specification* item which displays a pop-up window containing board size and layer biasing selections.
- The small panel to the right of this monitors the currently selected **Layout** mode.
- Below this, a small canvas provides a scaled-down view of the entire board (for clarity only the pins are shown), with the keyhole area (the main editing window display) shown inverted. The keyhole may be moved about the board from this canvas by pressing down the left mouse button with the cursor over the keyhole and dragging the mouse to a new position, before releasing the button.
- The panel below is for output only and provides constant feedback of the cursor's position, in board coordinates, while within the main editing canvas. The other fields only monitor data when the cursor is stationary over a track or pad in the main canvas. Information monitored is the **Route No.** to which the object belongs, and in the case of a pad, the **Device No.** it is associated with if any, it's **Pin No.** within the device and the **P/G Layer** (Power or Ground layer) if any, to which it is connected.
- The last panel, in the bottom right, monitors the main canvas editing environment. This is displayed as a set of editing environment variables whose current values change dynamically in line with alterations made on the main canvas. They may also be directly changed by the user. These include **Scale**, **Dir**ectory pathname, **File**name, **Foregound** and **Background** layers, default selections of object dimensions, the current **Pin p/g Layer** for laying down new pins, and **Constraint**, **Grid Lock** and **Design Hold** modes all of which are described later.

## 8.2 Editing

Editing is achieved mainly via the mouse but also the keyboard. The functions controlled by the three mouse buttons are modified by the SHIFT key and extended by the SPACE and DELETE keys. Note that for these functions to take effect the cursor must be in the main editing window. Figure 3 illustrates their general funtionality which is expanded below. Other keys are quick keys associated with the pull-down menu items and are dealt with later.

### 8.2.1 LEFT BUTTON – For object selection & movement

- *PIN & TRACK Modes* - Press to select the object under the cursor and deselect all others. This will visibly mark it. If the **SHIFT** key is held down at the same time, the currently selected set is augmented. If the button is dragged (held down and moved) after initial selection of an object, the object will move until the button is released. If **SHIFT** is used in conjunction with this, the complete selected set will move. Pressing when there is no object under the cursor will merely clear the selected set unless the **SHIFT** key is used. Dragging the cursor in these circumstances will produce a rubber rectangle which, on release, will cause all objects lying fully within it's limits to be set selected. If the rubber rectangle moves beyond the limits of the editing 'keyhole' the 'keyhole' will move across automatically to display the next portion of the board. If, by any means, a pin is selected which belongs to a device, then all other pins belonging to the device will also be selected.
- *NOGO Mode* - Press on an existing nogo corner to extend or alter a nogo area by dragging. Can also be used in conjunction the **MIDDLE** button in laying down new nogos of complex shapes (see above). This facility also forces the editing keyhole to follow it around the board.

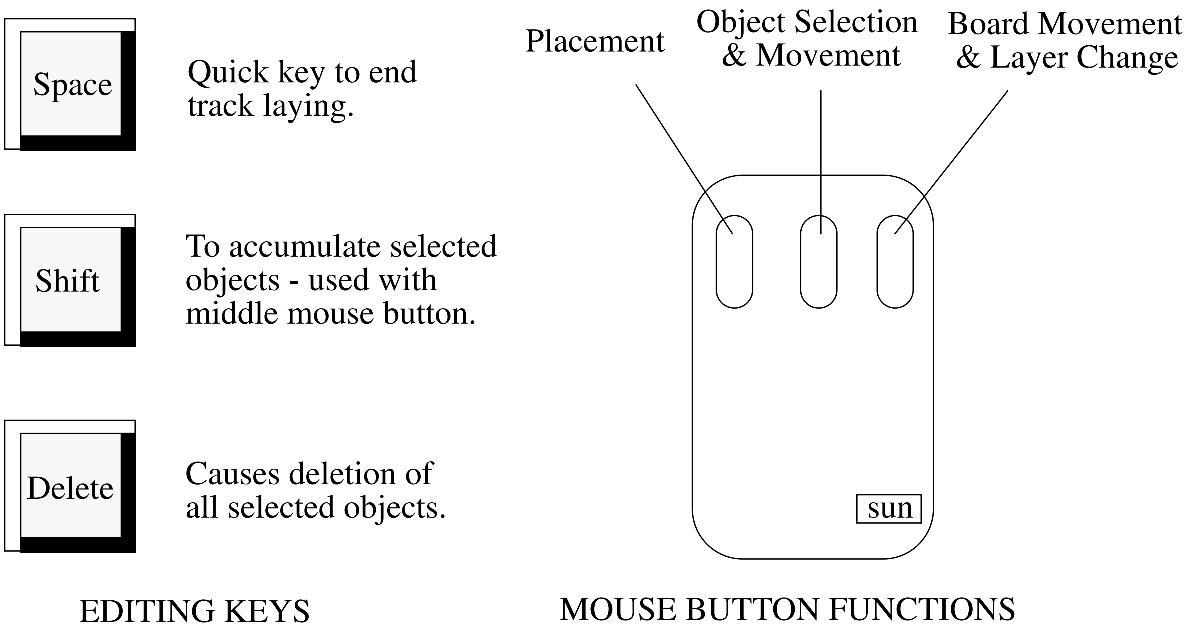



FIGURE 3 : EDITING FUNCTIONS

#### 8.2.2 MIDDLE BUTTON – For placement of new objects.

- *PIN Mode* - Plants a pin at the cursor. A pin cannot be layed on top of or too close to another. Any attempt is monitored by an error prompt box.
- *TRACK Mode* - Starts a rubber band track at the cursor. If already rubber banding, plants a corner first. A *Double-click* ends the track (also achieved by *SPACE*). Laying down tracks forces the editing keyhole to move automatically over the board as the cursor dictates.
- *NOGO Mode* - Creates a nogo area (initially rectangular) when the button is pressed and dragged, then realeased. (If the left button is pressed during dragging, a new nogo corner will be planted causing a change of shape.) NB. These actions merely 'black out' the nogo area making it obvious to the user if he attempts to place objects within it; no error is monitored if he does this.

#### 8.2.3 RIGHT BUTTON – Layer change & Movement.

- *TRACK Mode* - When laying tracks, clicking (press and release without movement) will cause a via to be planted and the display will automatically swop to the Background layer, where track laying can continue. When not laying tracks, clicking only causes the display swop. Dragging (press and move) intrinsically causes the board display to move around underneath the keyhole, although this is represented by the keyhole moving in the corresponding opposite direction on the small canvas.
- *PIN & NOGO Modes* - Clicking causes the display to swop Foreground and Background layers. Dragging causes the board display to move around underneath the keyhole as described above.

## 8.3 The Pull Down Editing Menus

These are illustrated in the sections below. Quick key equivalents appear alongside most items and only operate when the cursor is within the editing window.

#### 8.3.1 The Layout Menu

Figure 4 illustrates the possible selections. The current selection is monitored in the top right hand panel of the display and the default is *Tracks* as shown in Figure2. The description of the editing functions given earlier demonstrated how the mouse button functions are affected by this mode.

| Layout | |
|---|---|
| Track | (T) |
| Pin | (P) |
| Nogo | (N) |

FIGURE 4 : LAYOUT MENU

**Track & Pin layout strategies**

Some further explanation of the strategies used in track and pin placement is required. When a new track section is being layed down it may be required to join to existing tracks and pins in a number of ways. These connections may be attempts to join together separate routes. This would most likely be the case when producing a new design where initially there are no routes and they are being set up in the editor from scratch.

Designs from a *trackeript* file with pre-defined routes which may already be partially tracked, require protection from any mistaken attempts to alter their route integrity. The **Design_Hold** switch in the editing environment monitoring window is set by default to cause a warning prompt to appear when this occurs allowing the new track segment to be abandoned. The switch can be manually toggled to avoid this checking at the user's discretion.

When routes are deliberately connected together they amalgamate into the route which was used to initiate the connection. Nominally route zero is allocated to all pins which belong to no route and to tracks which do not connect to any pin. A new route is created when an unconnected track is joined to a pin with no route (or vice versa).

Pins which are layed over a track will cause the track to be joined to the pin centre. NB. This may cause the track to be diverted.

### 8.3.2 The Selection Menu

| Selection | |
|---|---|
| Extend by route | (R) |
| Extend between pins | (B) |
| Create Group | (G) |
| Duplicate | (D) |
| Spin 90' | (S) |
| Ungroup | (U) |

FIGURE 5 : SELECTION MENU

Figure 5 illustrates that the selection functions extend beyond the facilities made available through the mouse buttons and editing keys.

- *Extend by route* extends the current selection of objects according to the routes to which they belong. This is a useful method for identification of partially completed routes.

- *Extend between pins* causes all tracks connected to the currently selected tracks to be selected up to the nearest pin or dead-end. This is a useful way of identifying how pins are connected.

- *Create Group* allows the currently selected set of pins to be associated together as a device. This allows for convenient identification of devices for movement or duplication. The number allocated is merely the next numerically available following the last pre-defined device.

- *Duplicate* expects a device to have been pre-selected and produces a highlighted rectangle of it's shape which can be moved to a new location on the board. Error checking ensures that no unwanted connections occur.

- *Spin 90'* - Not yet implemented but should turn the pre-selected device through 90'.

- *Ungroup* - The pre-selected device is converted to a set of unassociated individual pins.

#### 8.3.3 The View Menu

Pulls down a set of options for split viewing of the main editing window.

| View | |
|---|---|
| Single View | |
| Split Horizontally | (H) |
| Split Vertically | (V) |

FIGURE 6 : VIEWING MENU

#### 8.3.4 The Attributes Menu

| Attributes | |
|---|---|
| Board Specification | |
| Track Width | => |
| Pin Diameter | => |
| Via Diameter | => |
| Grid Spacing | => |
| Pin p/g Plane | => |
| Set Attributes | (A) |
| Restore Defaults | (Z) |

FIGURE 7 : ATTRIBUTES MENU

- Board Specification - This displays a pop-up window as shown below. The top left of this is a button selection panel of standard eurocard sizes which when selected cause dimensions in thousandths/inch to be shown in the *Width* and *Height* fields to the right. These fields may also be set independently to any dimension required. Below is the layer bias table which displays 16 possible layers. By default only layers 1 and 2 are in use and are set to Horizontal and Vertical bias respectively. The biasing options are available as a 'checkmark cycle item list' which is shown when the right hand mouse button is pressed down with the cursor over the current selection for any layer. Otherwise only the current option for each layer is displayed. This can also be altered by clicking on it with the left mouse button which will cause the options to be cycled. The 'OK' field at the foot of the window causes all changes to be implemented and the pop-up to be removed. At this time the keyhole display window will automatically alter to indicate the size of the keyhole relative to the newly specified board size. Normally these settings are built into the design and are unlikely to be changed thereafter.

- *Track Width, Pin Diameter, Via Diameter & Grid Spacing* all have *pull-right* menus of the same form, namely a list of all the possible sizes.

- *Pin p/g Plane* - This allows a plane (layer) number to be specified as the power or ground layer to which all subsequently defined pins are connected. The *pull-right* menu lists all layer numbers which have a layer bias of *Power/Ground* as well as the *none* option. This is an editing environment parameter which is permanently displayed in the bottom right hand window of the display where it can also be altered.

- *Set Attributes* - This allows the user to set values (in thousandths of an inch) for the current track width, pin diameter, via diameter and grid spacing by selecting from pullright menus for each item.

- *Restore Defaults* - This allows the above Attributes to be reset to the values defined by the board file.

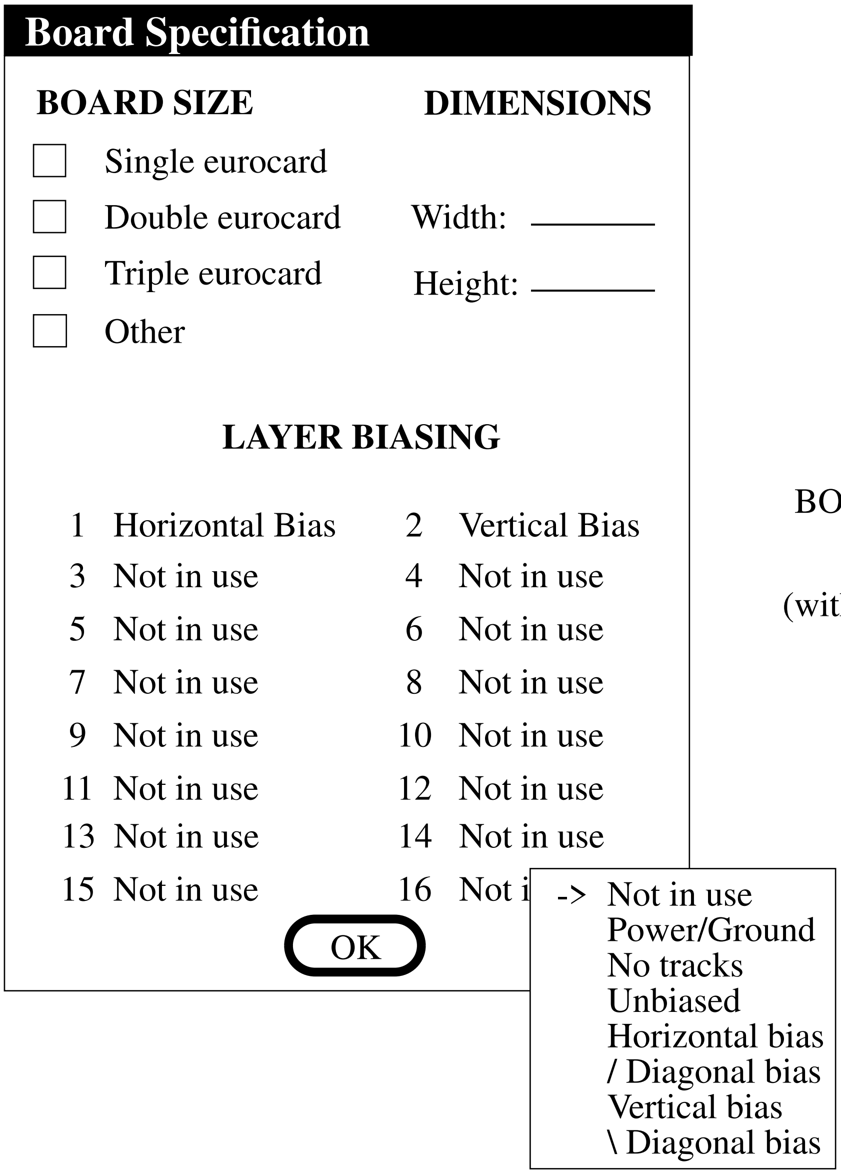



FIGURE 8 :
THE
BOARD SPECIFICATION
POP-UP WINDOW
(with Layer bias choice list)

### 8.4 The File Menu

The options available here are self explanatory. They use as their filename what is assumed to be the **ncap** format file specified in the environment window. The *Write* options will set up a **.ndir** directory if none exists.

### 8.5 The Environment variables

- **Scale** – This is a slider panel which provides both numerical and graphical indication of the current scale ratio in the range 1-15. Changing the slider value directly, using the mouse, will cause a zoom to the new scale.
- **Dir:** – This holds the first part of the pathname to used when files are accessed. Initially it is the current directory but may be altered at any time by the user.
- **File:** – This is the *ncap* filename which may again be altered by the user. It can contain further directory references.
- **Foreground** – The foreground layer can be changed here cyclically by clicking on it with the left mouse button or selected from a pull-down choice menu by the right mouse button. It is also flipped to the background

| File | |
|---|---|
| Open File | (O) |
| Write File | (W) |
| Write & Exit | (X) |
| Quit | (Q) |

FIGURE 9 : FILE MENU

layer indirectly by clicking the right mouse button when the cursor is in the main editing window.

- **Background** – Similar to **Foreground**
- **Constraint** – This is a two button off/on choice variable which when set constrains track segments to horizontal, vertical and 45’ layout options. By default it is set. Unsetting allows tracks to be layed out at any angle.
- **Grid Lock** – This is two button off/on choice variable which by default constrains the cursor to operate at grid points.
- **Design Hold** – This is a two button off/on choice variable which by default imposes a warning check on any attempt to connect together separate routes. Designs which are constructed from scratch would turn this off to allow routes to built up complete freedom of method.
- **Pin p/g Layer** – This variable defines which layer is the power or ground layer, if any, to which new pins are to be connected. It may be set cyclically by the clicking on it with the left mouse button or selected from a pull-down choice menu by the right mouse button. It can also be altered from the **Attributes** menu described above.
- **Track Width, Pin Dia, Via Dia & Grid** – These hold values in thousandths of an inch. They may be altered here directly or through the **Attributes** menu described earlier.

# 9 Display and Manufacture

This section describes the utilities available for display of the board design to date on a Sun monitor, a laser printer or a Hewlett packard plotter. It further describes the gerber utility which converts the current design into a set of gerber format files suitable for photoplotting and subsequent board manufacture.

## 9.1 Unixplot - Generating Unix format plot files

This utility generates a set of files in a sub-directory *design_name.ddir* which are in UNIX plot format. These may subsequently be displayed on a monitor via X-Windows using the xplot command or on a laser printer via the lpr -g option, or on a Hewlett Packard plotter via the hp_plot utility described later. It is invoked as:

**unixplot** *[-O] design_name*

If the directory does not already exist it is created, otherwise it is overwritten. UNIX plot format files are created for each board layer. These have filenames *pads.l<layer no>* and *lines.l<layer no>*. A *trackeript* file must exist in the *design_name.ndir* directory. This utility is called automatically by **netex**, which results in the *design_name.ddir* directory with a full set of plot files being available to the designer prior to tracking. Subsequent tracking iterations to the design can then be viewed by calling this utility independently.

Unixplot automatically invokes an optimiser to minimise latency time between drawn objects. This can be suppressed by use of the *[-O]* option.

## 9.2 Plot_fail - Generates Unix format plot file of untracked routes

This utility uses the tracker output file *trackeropt* to generate a *failures* file in the sub-directory *design_name.ddir* in UNIX plot format. As with the other files in this sub-directory it may subsequently be displayed on a monitor via X-Windows using the xplot command, on a laser printer via the lpr -g option, or on a Hewlett Packard plotter via the hp_plot utility described later. This shows the connections which the router has failed to make as a set of straight lines joining the unconnected pin pairs in question. It is invoked as:

**plot_fail** *design_name*

A *trackeropt* file must exist in the *design_name.ndir* directory. Only one *failures* is produced indicating unconnected pin pairs irrespective of layers. Finally monitoring is given of the percentage of failures.

## 9.3 Gerber - Generating Gerber Format for Photoplotting

This utility generates a set of files in **gerber** format which are the initial data for the manufacturing process. It is invoked as:

**gerber** *design_name*

It produces a directory *design_name.pdir* with 2 gerber files for each board layer. These have filenames *pads.l<layer no>* and *lines.l<layer no>*. A *trackeript* file must exist in the *design_name.ndir* directory. Tracker coordinates are multiplied by 10 to give gerber coordinates in tens of a thou.

Where pad, annulus, rectangle or square sizes have no exact equivalent aperture on the photo plotter, an attempt will be made to allocate a suitable alternative with appropriate monitoring being given. Pads, squares and rectangles will be allocated the next smaller available aperture, but annuli will be given the next larger. If no larger annuli exists a pad of equal or greater size will be allocated. When no suitable aperture can be allocated the program will abort with an error message.

**Gerber** draws a border around each board layer. On power layers the border lines go inside the actual edges of the board thus avoiding the possibility of layers shorting when the edges are trimmed. On the other layers the inside of the line marks the actual edge of the board.

## 9.4 Hp_plot - Generating Hewlett Packard Plotter Output

This utility drives an HP Plotter from files in Unix plot format producing drawings controlled by the options listed below. It is invoked by:

**hp_plot** *-[A2,A3,A4,a2,a3,a4,2,Ii,Rr,Ss,Vv,Cc,Oo] [f1 f2 ..]*

The files to be plotted may be specified by name (f1 f2 ..) or may be read from standard input when no names are specified. These files will be the UNIX plot format files held in the *design_name.ddir* directory. The options listed below may be specified:

| | |
|---|---|
| **-A2,A3,A4,** | |
| **-a2,a3,a4** | Allows page size A2,A3 or A4 to be forced by software. |
| **-2** | Allows second page of A2 output to be loaded. |
| **-I,i** | Interactive mode - User sets scale points and selects pen number. |
| **-R,r** | Forces rectangular (assymmetric) scaling. |
| **-S,s** <**scale**> | Scales by scale argument. |
| **-V,v** <**speed**> | Specifies pen velocity. |
| **-C,c x y** | Generates characters of size x, y. |
| **-O,o** | Suppresses circles being drawn as octagons. |

# A Tracker Input Format

An outline of the tracker input format is presented here. First, pseudo code to be used as a basis for reading the tracker file is given. This is then followed by a detailed description of each type of record with its associated fields. We apologise for the Fortran-like syntax: the fromat was not devised by MUCS.

```
    read JOB HEADER RECORD
    skip SPARE PARAMETER RECORD
    read COST PARAMETER RECORD

    do 10  i = 1, (HIGHEST LAYER - LOWEST LAYER) + 1
        read LAYER RECORD
10  continue

    if (NUMBER OF SELECTION RECORDS  >  0)  then
        do 20  i = 1,NUMBER OF SELECTION RECORDS
            read SELECTION RECORD
    if (SELECTION CODE  ≠  0)  then
                read NOGO RECORD
        do 15  j = 1,NUMBER OF SIDES OF ENCLOSED POLYGON - 1
            read CORNER RECORD
15              continue
    endif
20     continue
    endif


    read an indeterminate number of unconnected PIN RECORDS

    do 30  i = 1,NUMBER OF ROUTES
        read ROUTE HEADER RECORD
        do 40  j = 1,NUMBER OF PINS IN ROUTE
            read PIN RECORD
    do 32  k = 1,NUMBER OF LAYER SIZE RECORDS
                read LAYER SIZE RECORD
32          continue
40      continue
        if (NUMBER OF TRACKENDS - NUMBER OF PINS  >  0) then
            do 50  k = 1,NUMBER OF TRACKENDS - NUMBER OF PINS
                read FREE END RECORD or FLOAT END RECORD or VIA RECORD
                if (RECORD TYPE = VIA RECORD)
               do 48  l = 1,NUMBER OF LAYER SIZE RECORDS
                        read LAYER SIZE RECORD
48                  continue
           endif
50          continue
        endif
        if (NUMBER OF TRACKS IN ROUTE  >  0)  then
            do 60  l = 1,NUMBER OF TRACKS IN ROUTE
        read TRACK RECORD
                if (NUMBER OF SEGMENTS.gt.1)  then
  do 70  m = 1,NUMBER OF SEGMENTS - 1
                       read CORNER RECORD
70                 continue
        endif
60          continue
        endif
30  continue

75  read next record
    if (RECORD TYPE = NOGO RECORD)  then
        do 80  n = 1,NUMBER OF SIDES - 1
            read CORNER RECORD
80      continue
        goto 75
    endif

    return
    end
```

Tables describing the various record types now follow. For each of the record types a list of field descriptions is given. Each field description is divided into 3 parts:

- The field identifier or range of values that the field may have,
- A functional description of the field parameter,
- The type assumed by the field identifier or range.

| *Value* | *Description* | *Type* |
|---|---|---|
| **j** | Denotes job header record | Character |
| **0...n** | Minimum X coordinate of board in thou | Integer |
| **0...n** | Minimum Y coordinate of board in thou | Integer |
| **1...n** | Maximum X coordinate of board in thou | Integer |
| **1...n** | Maximum Y coordinate of board in thou | Integer |
| **1...n** | Lowest layer to be tracked | Integer |
| **1...n** | Highest layer to be tracked | Integer |
| **1...n** | Number of route records | Integer |
| **n.n** | Grid size in thou | Float |
| **1...n** | Pad clearance in thou | Float |
| **1...n** | Track clearance in thou | Float |
| **1...n** | Via clearance in thou | Float |
| **0...n** | Via symbol number (circular assumed) | Integer |
| **n.n** | Via size in thou | Float |
| **y,n** | Buried vias allowed | character |
| **0...n** | Default track image number (Ignored) | Integer |
| **n.n** | Default track width | Float |
| **1...n** | Maximum number of tracks between 0.1” pins (ignored) | Integer |
| **0...50** | Number of selection records<br>If 0 then attempt all tracks. If selection is by<br>window this figure should not exceed 1 | Integer |
| **0,1** | If 0 then stop routing after selected routes<br>If 1 then route rest of the board after selected routes | Integer |
| **0...10** | Type of router to use (ignored) | Integer |

Table 2: Job Header Record

Note that the list is ordered in the same manner as the fields are specified in the tracker input file. *All fields belonging to a particular record type must be specified on a single line.*

## A.1 Job Header Record

The job header record global board information such as the dimensions of the board, default track widths, number of layers, default clearances, number of net records, number of selection records etc.

## A.2 Spare Parameter Record

This record is spare and is ignored by all the routers.

## A.3 Cost Parameter Record

All the routers use a cost based Lee’s maze routing procedure in determining connection paths. Associated with the search process are several cost parameters which may be used to guide the search paths. There are a default set of costs, determined experimentally, that have been found to give reasonably good results that are hardwired to the trackers but the Cost parameter record may be used to overide these defaults. Further information on how to tune these parameters can be found in Appendix E.

## A.4 Layer Records

Layer records list for each layer of the board the preferred direction of tracks for that particular layer. Having wires predominantly in one direction on any given layer has been found to reduce the probability of wire blockage and thus improve the overall quality of wire layout. Cost based Lee routers achieve this by penalising any expansions against the grain (bias) of the layer. In the current implementations of the trackers, only -1,1 or 3 are used.

| *Value* | *Description* | *Type* |
|---|---|---|
| **k** | Denotes spare parameter record | Character |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |
| **0** | Spare | Integer |

Table 3: Spare Parameter Record

| *Value* | *Description* | *Type* |
|---|---|---|
| **m** | Denotes cost parameters record | Character |
| **0...1** | Cost record validity flag<br>0 = Ignore record<br>1 = Use record | Integer |
| **0...n** | Maximum number of source pins per expansion | Integer |
| **0...n** | Maximum spread for a line of pads | Integer |
| **0...15** | Penalty for inter layer expansions | Integer |
| **0...15** | Penalty for wrong direction expansions | Integer |
| **0...15** | Penalty for expanding away from target area | Integer |
| **0...15** | Gain for expanding towards the target area | Integer |
| **0...15** | Penalty for wrong way expansions around blocked cells | Integer |
| **0...15** | Maximum cost allocated to a cell | Integer |
| **0...15** | Gain for expanding parallel to other tracks | Integer |

Table 4: Cost Parameter Record

| *Value* | *Description* | *Type* |
|---|---|---|
| **l** | Denotes layer record | Character |
| **1...n** | Layer number | Integer |
| **-2...4** | Directional bias:<br>-2) Power/Ground layer<br>-1) No tracks on this layer<br>0) Tracks in any direction<br>1) Horizontal tracks preferred<br>2) / diagonal preferred<br>3) Vertical tracks preferred<br>4) \ diagonal preferred | Integer |

Table 5: Layer Direction Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **s** | Denotes selection record | Character |
| **0...3** | Selection codes available include:<br>0 indicates selection of routes by number, next two fields represent a contiguous block of one or more routes to be attempted. Total number of blocks should be equal to the number of selection records read.<br>1 instructs the tracker to only attempt those untracked pinpairs which lie WHOLLY WITHIN the polygon whose outline is specified in the nogo/corner records to follow. The next two fields will be ignored<br>2 instructs the tracker to only attempt those untracked pinpairs which lie WHOLLY OUTSIDE of the polygon whose outline is specified in the nogo/corner records to follow. The next two fields will be ignored<br>3 instructs the tracker to route only those pin pairs where at least one pin is part of an edge connector | Integer |
| **1...n** | First route in block to be attempted | Integer |
| **1...n** | Last route in block to be attempted | Integer |

Table 6: Selection Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **n** | Denotes nogo record | Character |
| **0...n** | Layer number - This will be ignored when the record is being used to define routing windows. The window is assumed to apply to all layers.<br>If layer is 0, then nogo record applies to all layers. | Integer |
| **-1,0,1** | -1) No vias allowed<br>0) No vias OR tracks allowed<br>1) No tracks allowed | Integer |
| **1...n** | X coordinate of start/end point in thou | Integer |
| **1...n** | Y coordinate of start/end point in thou | Integer |
| **3...n** | Number of sides of enclosed polygon | Integer |

Table 7: Nogo Records

## A.5 Selection Records

All the routers available in this software package incorporate a serial routing algorithm in which all nets are routed one by one. This means that the final result is dependent on the particular order in which the nets are routed. While every effort is taken by the routers to order the nets in an intelligent manner, the optimality of the ordering is by no means guaranteed. Selection records are used to give the user some control in the order in which nets are routed. These records may be used to specify critical nets to route first or instead identify critical areas of the boards that need to be tracked first e.g in the region of edge connectors. The tracker input file probably needs to be edited manually by the user in order to specify this particular record. Owing to the non-deterministic nature of annealed tracking, selection records will be ignored with this particular mode of tracking.

## A.6 Nogo Records

Nogo records are used to isolate specific areas and layers of the board which need to have special restrictions placed on them. The different restrictions that can be placed are listed in the table below.

| *Value* | *Description* | *Type* |
|---|---|---|
| **c** | Denotes corner record | Character |
| **1...n** | X coordinate of corner in thou | Integer |
| **1...n** | Y coordinate of corner in thou | Integer |

Table 8: Corner Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **r** | Denotes route header record | Character |
| **1...n** | Route number | Integer |
| **2...n** | Number of pins in route | Integer |
| **2...n** | Number of trackends in route | Integer |
| **0...n** | Number of tracks in route | Integer |
| **0...n** | Default track image number (ignored) | Integer |
| **n.n** | Trackwidth to be used for this route | Float |
| **1...n** | Spare | Integer |
| **1...n** | Spare | Integer |

Table 9: Route Records

## A.7 Corner Records

Corner records form the basis of all tracks and shapes specifications. The number of corner records is usually defined in the appropriate header records (see Nogo Records and Track Records). When specifying nogo areas, it is assumed tha the relevant area is enclosed. When used to describe a track, corner records identify the vertices of the track. Note that the end point of a track is also a corner point.

## A.8 Route Records

The route header record is used to specify a logical net. The net may either be unrouted or already partially or fully routed. Pin, via and track records describe the physical position of elements in the net.

## A.9 Pin Records

There are 2 logical categories of pins, unconnected and connected pins. Unconnected pins do not belong to any net. They may be, for example, pins connected to a power plane. Connected pins belong to a single net record and can only be connected to another pin belonging to the same net. Both categories of pins can be described using the same record and may be distinguished by the character in the first field of the record. Each pin record has associated with it a number of layer records which convey the physical characteristics of the pad such as its shape, size and extent of the corresponding physical characteristic on the board (in other words the layers to which the physical information applies). Initially, unrouted pins within a particular route number, possess a unique link-label. Pins, physically connected however posses the same link label.

## A.10 Via Records

Via records are used to specify vias, special pads which allow connections to cross to other layers. Like pin records, each via has a number of associated physical layer records as shown in the Via Records table.

## A.11 Free End/ Floating End Records

Floating ends and free ends are special pad records used to convey teeing information. These records are used internally by post-track software. The current implementations of the trackers ignore these records. It is included here for completeness sake.

| *Value* | *Description* | *Type* |
|---|---|---|
| **p,u** | Denotes pin record - p(route), u(unconnected) | Character |
| **0...n** | Route number, 0=unconnected otherwise 1....n | Integer |
| **1...n** | X coordinate of pad centre in thou | Integer |
| **1...n** | Y coordinate of pad centre in thou | Integer |
| **1...n** | Start layer number | Integer |
| **2...n** | Finish layer number | Integer |
| **0...n** | Link label - Any pad or via in the same ROUTE with the same link label value is connected by track to other pads of the same link value. Ignored if the pin is unconnected | Integer |
| **1,2,3,4** | Type of device<br>1 - ic, 2 - edge con, 3 - discrete, 4 - other | Integer |
| **1...n** | Pin number in device type | Integer |
| **1...n** | Number of layer size records | Integer |
| **1...n** | Component/Device number | Integer |
| **x,y,z** | Orientation of pins 1 and 2 in this instance | Integer |
| **1...n** | Spare | Integer |
| **1...n** | Spare | Integer |

Table 10: Pin Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **v** | Denotes via record | Character |
| **0...n** | Via symbol number | Integer |
| **1...n** | X coordinate of pad centre in thou | Integer |
| **1...n** | Y coordinate of pad centre in thou | Integer |
| **1...n** | Start layer number | Integer |
| **2...n** | Finish layer number | Integer |
| **1...n** | Link label - Any pad or via in the same ROUTE with the same link label value is connected by track to other pads of the same link value. | Integer |
| **1...n** | number of layer size records | Integer |

Table 11: Via Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **q** | Denotes a layer size record | Character |
| **c,s,r,n** | Pad Type: c - circle, s - square<br>r - rectangle, n - annulus | Character |
| **0,45,90,135** | Angle of rotation of pad. Circle is always 0, Square is 0 or 45, Rectangle is 0,45,90,135 | Integer |
| **n.n** | Diameter of circle, side of square, length of rectangle in thou | Float |
| **n.n** | Breadth of rectangle, otherwise 0 thou | Float |
| **0...n** | Start layer. 0 if applies to all layers | Integer |
| **2...n** | Finish layer | Integer |

Table 12: Layer Size Record

| *Value* | *Description* | *Type* |
|---|---|---|
| **f,g** | Free end(f), Floating end(g) | Character |
| **1...n** | X coordinate of endpoint in thou | Integer |
| **1...n** | Y coordinate of endpoint in thou | Integer |
| **1...n** | Link label - If any of free, float, pin and via ends ends in this route has same link label then there is a track between them | Integer |
| **1...n** | Layer number | Integer |

Table 13: Free End/ Floating End Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **t** | Denotes track record | Character |
| **0...n** | Track image number (ignored) | Integer |
| **n.n** | trackwidth in thou | Float |
| **1...n** | X coordinate of start point in thou | Integer |
| **1...n** | Y coordinate of start point in thou | Integer |
| **1...n** | X coordinate of end point in thou | Integer |
| **1...n** | Y coordinate of end point in thou | Integer |
| **1...n** | Layer number | Integer |
| **1...n** | Number of corner records | Integer |

Table 14: Track Records

## A.12 Track Records

Track records are used to describe the layout of a connection on the board. Each track starts from either a pin or a via and ends on either a pin,via or another track. Note that the X,Y coordinates of the endpoint refer to the‘ logical endpoint of the track. Thus, if a track tees onto another track for example, the latter track is recursively followed to its endpoint which becomes the endpoint of the former track. The endpoint of a track is always described by a corner record as well as appearing in the track record.

# B Tracker Output Format

The basic layout of the tracker output file is illustrated below:

```
Job Header record (same as for trackeript)

       +-------------------------- 1...n
       |  Route
       |       Pin
       |       Pin
       |      +------------------- 1...n
       |      |     Via            0...1
       |      |     Track
       |      |          Corner
       |      |          Corner    1...n
       |      +-------------------
       |   Failures                0...n
       +--------------------------
```

Some additional points include :

- A track may start on a pin, existing via or tee out of an existing track.
- There is only a single pin pair per route record.
- Routes and the associated pin pair sets may appear in any order.
- Failure records may be output with each route or collectively at the end of the file.

The information conveyed by the various records is essentially the same as the corresponding records in the tracker input format. Users are asked to refer to Appendix A for relevant details.

| *Value* | *Description* | *Type* |
|---|---|---|
| **C** | Denotes corner record | Character |
| **1...n** | X coordinate of corner in thou | Integer |
| **1...n** | Y coordinate of corner in thou | Integer |

Table 15: Corner records

| *Value* | *Description* | *Type* |
|---|---|---|
| **P** | Denotes pin record | Character |
| **1...n** | X coordinate of pad centre in thou | Integer |
| **1...n** | Y coordinate of pad centre in thou | Integer |
| **1...n** | Spare | Integer |
| **1...n** | Spare | Integer |

Table 16: Pin Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **R** | Denotes route header record | Character |
| **1...n** | Route number | Integer |
| **1...n** | Spare | Integer |
| **1...n** | Spare | Integer |

Table 17: Route Header Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **T** | Denotes track record | Character |
| **0...n** | Track image number | Integer |
| **1...n** | Layer number | Integer |
| **1...n** | Number of corner records | Integer |

Table 18: Track Records

| *Value* | *Description* | *Type* |
|---|---|---|
| **V** | Denotes via record | Character |
| **0...n** | Via symbol number | Integer |
| **1...n** | X coordinate of pad centre in thou | Integer |
| **1...n** | Y coordinate of pad centre in thou | Integer |
| **1...n** | Start layer number | Integer |
| **2...n** | Finish layer number | Integer |
| **y,n** | y if via lies on existing track other wise n | Integer |

Table 19: Via records

| *Value* | *Description* | *Type* |
|---|---|---|
| **X** | Denotes failed connection record | Character |
| **1...n** | X coordinate of start | Integer |
| **1...n** | Y coordinate of start | Integer |
| **1...n** | X coordinate of finish | Integer |
| **1...n** | Y coordinate of finish | Integer |
| **1...n** | route number | Integer |
| **n** | Spare | Integer |

Table 20: Failed Connection Records

# C PCB Outlines

## C.1 Standard device library

This is the standard library to be found in pcb/design_tools/lib/outlines. It is suitable for tracking with "normal" design rules (See Section 5.2, page 16)

| *File* | *Description* | *Pins* | *Pitch* | *Pad Size* |
|---|---|---|---|---|
| dev0.8.0 | dilic with capacitor | 8 | 300 | 60 |
| dev0.10.0 | dilic with capacitor | 10 | 300 | 60 |
| dev0.14.0 | dilic with capacitor | 14 | 300 | 60 |
| dev0.16.0 | dilic with capacitor | 16 | 300 | 60 |
| dev0.18.0 | dilic with capacitor | 18 | 300 | 60 |
| dev0.20.0 | dilic with capacitor | 20 | 300 | 60 |
| dev0.22.0 | dilic with capacitor | 22 | 300 | 60 |
| dev0.24.0 | dilic with capacitor | 24 | 300 | 60 |
| dev0.24.1 | dilic with capacitor | 24 | 600 | 60 |
| dev0.28.1 | dilic with capacitor | 28 | 600 | 60 |
| dev0.32.1 | dilic with capacitor | 32 | 600 | 60 |
| dev0.40.1 | dilic with capacitor | 40 | 600 | 60 |
| dev0.48.1 | dilic with capacitor | 48 | 600 | 60 |
| dev0.64.2 | dilic with capacitor | 64 | 900 | 60 |
| dev1.6.0 | dilic | 6 | 300 | 60 |
| dev1.8.0 | dilic | 8 | 300 | 60 |
| dev1.10.0 | dilic | 10 | 300 | 60 |
| dev1.14.0 | dilic | 14 | 300 | 60 |
| dev1.16.0 | dilic | 16 | 300 | 60 |
| dev1.18.0 | dilic | 18 | 300 | 60 |
| dev1.20.0 | dilic | 20 | 300 | 60 |
| dev1.22.0 | dilic | 22 | 300 | 60 |
| dev1.24.0 | dilic | 24 | 300 | 60 |
| dev1.24.1 | dilic | 24 | 600 | 60 |
| dev1.28.1 | dilic | 28 | 600 | 60 |
| dev1.40.1 | dilic | 40 | 600 | 60 |
| dev1.48.1 | dilic | 48 | 600 | 60 |
| dev1.64.2 | dilic | 64 | 900 | 60 |
| dev2.2.0 | SIL pack | 2 | - | 60 |
| dev2.3.0 | SIL pack | 3 | - | 60 |
| dev2.4.0 | SIL pack | 4 | - | 60 |
| dev2.5.0 | SIL pack | 5 | - | 60 |
| dev2.6.0 | SIL pack | 6 | - | 60 |
| dev2.7.0 | SIL pack | 7 | - | 60 |
| dev2.8.0 | SIL pack | 8 | - | 60 |
| dev2.9.0 | SIL pack | 9 | - | 60 |
| dev2.10.0 | SIL pack | 10 | - | 60 |
| dev2.12.0 | SIL pack | 12 | - | 60 |
| dev2.13.0 | SIL pack | 13 | - | 60 |
| dev2.15.0 | SIL pack | 15 | - | 60 |
| dev3.64.0 | 64 way dil connector | 64 | - | 60 |
| dev4.96.0 | 96 way til connector | 96 | - | 60 |
| dev5.26.0 | DIL Header | 26 | - | 60 |
| dev5.50.0 | DIL Header | 50 | - | 60 |

Table 21: Standard Library: Devices 0 - 5

A 64 way indirect edge connector (dev3) has 2 rows of 32 pins, conventionally labelled 1a to 32a and 1c to 32c. The pin numbering in the MUCS system is 1 to 64, i.e. pin 33 corresponds to pin 1c. Similarly, a 96 way indirect connector (dev4) is conventionally labelled 1a to 32a, 1b to 32b, 1c to 32c. The pin numbering in the MUCS system is from 1 to 96, i.e. pin 33 corresponds to 1b and pin 65 corresponds to pin 1c.

| *File* | *Description* | *Pins* | *Pitch* | *Pad Size* |
|---|---|---|---|---|
| dev6.*.* | Not in current use | | | |
| dev7.*.* | Not in current use | | | |
| dev8.2.0 | 2 pin component | 2 | 100 | 60 |
| dev8.2.1 | 2 pin component | 2 | 200 | 60 |
| dev8.2.2 | 2 pin component | 2 | 300 | 60 |
| dev8.2.3 | 2 pin component | 2 | 400 | 60 |
| dev8.2.4 | 2 pin component | 2 | 500 | 60 |
| dev9.1.0 | Single Pad | 1 | - | 60 |
| dev9.1.1 | Single Pad | 1 | - | 75 |
| dev9.1.2 | Single Pad | 1 | - | 80 |
| dev9.1.3 | Single Pad | 1 | - | 100 |
| dev9.1.4 | Single Pad | 1 | - | 125 |
| dev9.1.5 | Single Pad | 1 | - | 150 |
| dev9.1.6 | Single Pad | 1 | - | 200 |
| dev10.4.0 | Xtal Oscillator | 4 | 300 | 60 |
| dev11.0.0 | Preconnected Electrolytic Decoupler | 0 | 500 | 65 |
| dev20.30.0 | simm | 30 | - | 60 |
| dev30.50.0 | IDC Connector | - | - | 60 |
| dev99.1.0 | Power Pad - internal use only | 1 | - | 60 |
| dev200.169.0 | pin grid array:AMD 32Bit ALU | 169 | - | 60 |
| dev201.175.0 | pin grid array:Xilink 3090 | 175 | - | 60 |
| dev202.144.0 | pin grid array:LSI Logic FPU | 144 | - | 60 |
| dev203.120.0 | pin grid array:AMD AMC register | 120 | - | 60 |

Table 22: Standard Library: Devices 6-1000

Note that device 11, a preconnected electrolytic capacitor is shown as having zero pins. This is because the device specification connects the pin at pin 1 (0,0) to Vcc and pin 2 at (0,500) to Gnd. Consequently, there are no signal pins unconnected.

## C.2 outlines.20 device library

This library contains the same components as the standard library. The only difference is that the pad sizes are optimised for auto-routing on a 20 thou grid.

| *File* | *Description* | *Pins* | *Pitch* | *Pad Size* |
|---|---|---|---|---|
| dev0.8.0 | dilic with capacitor | 8 | 300 | 50 |
| dev0.10.0 | dilic with capacitor | 10 | 300 | 50 |
| dev0.14.0 | dilic with capacitor | 14 | 300 | 50 |
| dev0.16.0 | dilic with capacitor | 16 | 300 | 50 |
| dev0.18.0 | dilic with capacitor | 18 | 300 | 50 |
| dev0.20.0 | dilic with capacitor | 20 | 300 | 50 |
| dev0.22.0 | dilic with capacitor | 22 | 300 | 50 |
| dev0.24.0 | dilic with capacitor | 24 | 300 | 50 |
| dev0.24.1 | dilic with capacitor | 24 | 600 | 50 |
| dev0.28.1 | dilic with capacitor | 28 | 600 | 50 |
| dev0.32.1 | dilic with capacitor | 32 | 600 | 50 |
| dev0.40.1 | dilic with capacitor | 40 | 600 | 50 |
| dev0.48.1 | dilic with capacitor | 48 | 600 | 50 |
| dev0.64.2 | dilic with capacitor | 64 | 900 | 50 |
| dev1.6.0 | dilic | 6 | 300 | 50 |
| dev1.8.0 | dilic | 8 | 300 | 50 |
| dev1.10.0 | dilic | 10 | 300 | 50 |
| dev1.14.0 | dilic | 14 | 300 | 50 |
| dev1.16.0 | dilic | 16 | 300 | 50 |
| dev1.18.0 | dilic | 18 | 300 | 50 |
| dev1.20.0 | dilic | 20 | 300 | 50 |
| dev1.22.0 | dilic | 22 | 300 | 50 |
| dev1.24.0 | dilic | 24 | 300 | 50 |
| dev1.24.1 | dilic | 24 | 600 | 50 |
| dev1.28.1 | dilic | 28 | 600 | 50 |
| dev1.40.1 | dilic | 40 | 600 | 50 |
| dev1.48.1 | dilic | 48 | 600 | 50 |
| dev1.64.2 | dilic | 64 | 900 | 50 |
| dev2.2.0 | SIL pack | 2 | - | 50 |
| dev2.3.0 | SIL pack | 3 | - | 50 |
| dev2.4.0 | SIL pack | 4 | - | 50 |
| dev2.5.0 | SIL pack | 5 | - | 50 |
| dev2.6.0 | SIL pack | 6 | - | 50 |
| dev2.7.0 | SIL pack | 7 | - | 50 |
| dev2.8.0 | SIL pack | 8 | - | 50 |
| dev2.9.0 | SIL pack | 9 | - | 50 |
| dev2.10.0 | SIL pack | 10 | - | 50 |
| dev2.12.0 | SIL pack | 12 | - | 50 |
| dev2.13.0 | SIL pack | 13 | - | 50 |
| dev2.15.0 | SIL pack | 15 | - | 50 |
| dev3.64.0 | 64 way dil connector | 64 | - | 55 |
| dev4.96.0 | 96 way til connector | 96 | - | 55 |
| dev5.26.0 | DIL Header | 50 | - | 50 |
| dev5.50.0 | DIL Header | 50 | - | 50 |

Table 23: outlines.20 Library: Devices 0 - 5

A 64 way indirect edge connector (dev3) has 2 rows of 32 pins, conventionally labelled 1a to 32a and 1c to 32c. The pin numbering in the MUCS system is 1 to 64, i.e. pin 33 corresponds to pin 1c. Similarly, a 96 way indirect connector (dev4) is conventionally labelled 1a to 32a, 1b to 32b, 1c to 32c. The pin numbering in the MUCS system is from 1 to 96, i.e. pin 33 corresponds to 1b and pin 65 corresponds to pin 1c.

| *File* | *Description* | *Pins* | *Pitch* | *Pad Size* |
|---|---|---|---|---|
| dev6.*.* | Not in current use | | | |
| dev7.*.* | Not in current use | | | |
| dev8.2.0 | 2 pin component | 2 | 100 | 50 |
| dev8.2.1 | 2 pin component | 2 | 200 | 50 |
| dev8.2.2 | 2 pin component | 2 | 300 | 50 |
| dev8.2.3 | 2 pin component | 2 | 400 | 50 |
| dev8.2.4 | 2 pin component | 2 | 500 | 50 |
| dev9.1.0 | Single Pad | 1 | - | 55 |
| dev9.1.1 | Single Pad | 1 | - | 75 |
| dev9.1.2 | Single Pad | 1 | - | 80 |
| dev9.1.3 | Single Pad | 1 | - | 100 |
| dev9.1.4 | Single Pad | 1 | - | 125 |
| dev9.1.5 | Single Pad | 1 | - | 150 |
| dev9.1.6 | Single Pad | 1 | - | 200 |
| dev10.4.0 | Xtal Oscillator | 4 | 300 | 50 |
| dev11.0.0 | Preconnected Electrolytic Decoupler | 0 | 500 | 65 |
| dev20.30.0 | simm | 30 | - | 50 |
| dev30.50.0 | IDC Connector | - | - | 50 |
| dev99.1.0 | Power Pad - internal use only | 1 | - | 60 |
| dev200.169.0 | pin grid array:AMD 32Bit ALU | 169 | - | 50 |
| dev201.175.0 | pin grid array:Xilink 3090 | 175 | - | 50 |
| dev202.144.0 | pin grid array:LSI Logic FPU | 144 | - | 50 |
| dev203.120.0 | pin grid array:AMD AMC register | 120 | - | 50 |

Table 24: outlines.20 library: Devices 6-1000

Note that device 11, a preconnected electrolytic capacitor is shown as having no pin. This is because the device specification connects the pin at pin 1 (0,0) to Vcc and pin 2 at (0,500) to Gnd. There are thus no signal pins unconnected.

## C.3 outlines.xtra library

This library contains various custom odds and sods!

| *File* | *Description* | *Pins* | *Pitch* | *Pad Size* |
|---|---|---|---|---|
| dev401.3.0 | DSPDT (Single pole double throw switch) | 3 | - | 50 |
| dev402.4.0 | not in current use | 4 | - | 60 |
| dev403.4.0 | Vertical strenghtening bar | 4 | - | 150 |
| dev404.4.0 | Horizontal strengthening bar | 4 | - | 150 |
| dev405.2.0 | Fuse | 2 | - | 80 |

Table 25: outlines.xtra library

# D Other Programs Used In PCB Production

## D.1 SUN Network

**emma:** Sends EMMA format photoplot files to paper tape input on EMMA 85 photoplotter.

**etherdrill:** Sends drill data from EMMA photoplot files OR Excel drill file to MICRONIC 1 drill.

**emmadraw,nemmadraw:** Converts data from BOTTOM DRAWER format to EMMA photoplot format, and adds border with reference marks.

**gemma,ngemma:** Converts data from GERBER photoplot format to EMMA photoplot format, and adds border with reference marks.

**gerbmap:** Applies scale factors and coordinate shifts to GERBER photoplot files, removes some redundant commands befor using **gemma.**

**lineopt,padopt:** Optimisers for photoplot files to reduce wasted movement and image changes.

## D.2 PC Compatibles - PROTEL programs

**TRAXEDIT:** PCB layout and routing.

**TRAXPLOT:** Converts TRAXEDIT output for plotting.

**SCHEDIT:** Schematic design.

**SCHPLOT:** Converts SCHEDIT output for plotting.

**TRAXSTAR:** Supplementary autorouter for TRAXEDIT.

**GVIEW:** Gerber photoplot file viewer and editor

# E Tuning Search Costs

The default costs are shown below (the maximum cost that can be specified is 15). Advice on how to tune these costs will be released later. Some care is needed in interpreting the costs. The routers uses “staged” searches; the default order of staging is:

primary direction before secondary direction before through vias.

Consequently, the actual cost of a search in the secondary direction can never be less than that in the primary direction even if a cost of 15, say, is assigned to the primary direction, and a cost of 2 is assigned to the secondary direction. The real costs can be calculated from the specified costs as follows:

primary cost = specified primary-directioncost
secondary cost = primary cost + specified secondary-direction cost + 1
via cost = secondary cost + specified via cost + 1

The greedy and annealed routers allow a switchbox option in which the staging of the secondary-direction searches and the through-via searches are reversed; in other words it is more favourable to take a via rather than a wrong direction move. This is suited to technologies where the size of a via is comaprable with the track width.

There are several other parameters defined below. They may all be specified in the Cost Parameter Record (page 39) of the trackeript file. The program **design-filter** (Section 7.3, page 25 may be used to change this record if you are nervous about editing the file directly. The *madarch* example (page 63) supplied with the software illustrates the use of this facility.

| | Normal Costs | Switchbox Costs |
|---|---|---|
| Maximum number of source points | 2 | 2 |
| Maximum spread for a line of pads | 500 | 500 |
| Penalty for via expansions | 6 | 8 |
| Penalty for wrong direction exapansions | 8 | 6 |
| Penalty for expanding away from target | 4 | 0 |
| Gain for expanding towards target | 1 | 0 |
| Penalty for wrong way expansions around blocked cells | 0 | 15 |
| Maximum cost allocated to a cell | 7 | 7 |
| Gain for expanding parallel to other tracks | 2 | 0 |

Table 26: Default Costs

# F Printed Circuit Board Production

This note explains the printed circuit production process in the Dept of Computer Science, University of Manchester (hereafter referred to as MUCS). An understanding of the process may be helpful in deciding how to design a printed circuit board.

## F.1 Board Types and Layer Assigments

The cost of a board depends on its size and on the number of layers. Increasing the number of layers will ease the routing task, aid power distribution and decrease the area, although the cost/unit area will increase.

The simplest board type is a *single layer* board which as its names suggests has copper on only one side of the board. A *double sided* board has copper on both sides of the board, but has no connections between the two layers. Typically, one side of the board is used for inter-connections and the other side is used as a ground plane. The cost of the extra layer is only about 20%. A common board type is a *double sided pth* board, which has connections between the two layers of copper achieved by means of plated-through-holes. It is thus feasible to put inter-connections on both sides of the board, greatly increasing the ease of routing but the cost is increased by a further 50%. Power distribution is often difficult, especially for high performance circuits, with two layer boards because the $V_{cc}$ and GND needs to be supplied using long tracks with associated inductance problems. The solution is to use a *4-layer* board. In this case, two outer layers are used for the signal interconnections, whilst the two inner layers distribute $V_{cc}$ and GND. It is not usual to place any inter-connections on the power layers because doing so would destroy the low inductance properties of the copper planes. The cost of a 4-layer board is approximately double that of a two layer board. However, a greater packing density should be achievable since the inter-connection layers are free of any power distribution tracks. For more complex boards, thicker sandwiches can be manufactured with more interconnection layers with a corresponding increase in cost.

The top layer of a board is normally referred to as the component side and is the layer on which components are placed. Tracks (inter-connections) may also be present on this layer. The bottom layer is usually referred to as the solder layer and is usually reserved for tracking. Occasionally, to achieve the greatest packing density, surface-mount devices may be placed on this layer. In this case, users of MUCS design software will need two device descriptions in OUTLINE_LIBS (see Section 3 – Netex), for each SMT used in this manner: one description for the device placed on the normal component layer and a second for the device placed on the solder layer. Inner layers may either be Power or Ground layers, or in the case of boards with 6 or more layers, they may be tracking layers.

## F.2 Basic Process

The basic material for all PCBs produced by MUCS is epoxy-glass laminate in various thicknesses, clad on one or both sides with copper foil. The required circuit configuration is produced by chemically etching away the areas of copper not needed.To achieve this, the areas to be retained are protected by an etch resist which is defined by a photo-imaging process. The whole surface is coated with a polymer sheet which can be hardened by exposure to ultra-violet radiation. This layer is exposed through a high-contrast negative of the required circuit pattern, hardening the areas over the final circuit configuration. The unexposed areas of the polymer are dissolved away by a developing process, leaving the unwanted copper accessible to the etchant which dissolves it away in the next process stage. Finally, the hardened etch-resist layer is stripped away, and a protective coating of tin applied to the finished copper circuit.

If connections are required between the two sides of a double-sided board,they can be made by depositing metal on the walls of drilled holes between the two sides. This process, known as through-hole plating, normally involves an initial chemically-deposited seeding layer of palladium or copper, which is built up to a suitable thickness to carry the required currents and withstand soldering stresses by electroplating more copper onto the seeded surface. All the required holes are drilled and through-plated before the final definition and etching of the circuit pattern.

Circuit boards of any required thickness and number of circuit layers can be produced by stacking double- and/or single-sided boards in register, and bonding them together using epoxy resin impregnated glass cloth, which acts as electrical insulation between adjacent metal layers as well as providing the bonding adhesive. Clearly the internal circuit layers must be etched before the stack is bonded together. Following bonding, the multi-layer board is drilled, through-plated, and the outer layers defined and etched exactly as for a double-sided board. The through-plated holes then serve as a means of interconnection between all the circuit layers, whether buried or external.

## F.3 Buried Vias

A variation on this straightforward multi-layer construction which may be useful in saving board area for dense tracking is to use buried vias. These are plated-through holes which go only between two adjacent layers rather than through the entire board thickness. In this case each relevant pair of layers is produced as if it were an individual through- plated double-sided board before the stack of layers can be bonded together, *so the production cost is considerably increased.* As before, after bonding the whole stack is drilled with those holes which run through the whole board thickness (which obviously must not intersect any of the buried vias on any internal layer pair), and then through-plated and etched in the usual manner.

As the name implies, buried via layers must be actually buried within the structure of the complete multi-layer board. This is because during the bonding process, molten adhesive is forced through the holes, and would contaminate the outer surfaces, interfering with subsequent processing, if the buried via layers with pre-drilled and plated holes were actually on the outside of the stack being bonded. Our normal structure for buried via boards therefore is to add a single copper layer to each outer face of the assembled stack, which can contain as many pairs of buried via layers as required. *These outer layers have pads only for the holes which go through the full thickness of the board*, e.g. for component mounting, but may of course also be used for carrying tracks, or even as power or ground planes.

## F.4 Drawing Pads and Lines

As stated earlier, for each layer of circuitry, a negative photo- tool is required through which to expose the photo-imagable etch resist coating. For tracking layers, this is a mostly opaque film with clear areas representing tracks and pads. For power or ground planes, the film is mostly clear, with opaque sections representing metal to be removed, usually to isolate areas where a through-hole which is NOT to be connected to that layer passes through. These films are generated in our case by an EMMA85 vector photoplotter, which uses a set of 48 apertures, and can expose pads by static flashing whilst stationary, or tracks by moving with the appropriate aperture continuously illuminated. Either negative or direct-reversal film may be used so that a working negative can always be produced in a single step by drawing just the features required, whether they represent copper to be retained (track layers) or absence of copper (power/ground planes). It is strongly recommended that designs are organised so that they use pad sizes and track widths which correspond directly to available apertures on our photoplotter.

Most design software will support "painting" with a smaller aperture to create features of odd shapes and sizes, but this can lead to poor definition, unnecessarily protracted plotting times, and various other problems, (for example we will not be able to test your boards for track and pad continuity). It is rarely really necessary to use painting if sufficient forethought is used.

For designs to be produced by us, it is essential that pads on tracking layers are plotted as SOLID pads with no centre hole. If not, in the etching process the etchant enters the hole and removes the through-plating. To define isolations on power/ground planes, we prefer the use of an annulus rather than a solid pad. This retains a circle of copper over the position where the hole is drilled, which leads to more consistent drilled hole quality than if all the copper were removed and the resulting void (in a multi-layer board) filled with adhesive resin. For the same reason, all tracking layers should contain a full complement of copper pads matching the outer layers and the drill pattern, even if no tracks connect with an individual pad.

## F.5 Drill Information

The holes in the boards are drilled using a numerically controlled drill capable of automatically selecting a range of different size drill bits. Normally, the information required to drill the board is derived from the pads on the outer layers. By associating drill size with pad code, we can drill different size holes as required in the board.

This method of deriving drilling data is only valid when the pads are plotted by flashing not by painting. If it is *really* necessary for you to have non-circular pads produced by painting, then you must provide us with a NC drill file from your design package in EXCELLON format together with information on what drill size you require for each number in the NC drill file. In this case, we may have to make an extra charge for the non-standard operations involved.

If more than one drill size is required, we can work out which pads are to be drilled with which drill size by a mapping of drill size on to pad size. sizes.

A table of apertures available on our EMMA85 photoplotter is appended. This information should allow the photoplot file generation utility of any reasonable PCB design package to be configured to produce output to run on our plotter.

We have an EMMA85 vector photoplotter, and processing facilities for film sizes up to 24 x 20 inches. We can accept files for photoplotting in either EMMA or Gerber format. For further information on current pricing, turn-round and facilities, email: `pcb@uk.ac.man.cs`

## F.6 Summary of Guidelines

- Avoid buried vias if possible – they are expensive
- Vias contacting the outer layers should *only* be through-board.
- Outer layer pads should match each other.
- Use pad sizes and track widths which correspond to the sizes available from EMMA.
- Don't "paint" pads or tracks
- Use solid pads on tracking layers and annuli on the power/ground planes

## F.7 EMMA85 Image set

| EMMA code | Gerber code | Size (x 0.001”) | Shape | Use for L(line), F(flash),E(either) N(not recommended) |
|---|---|---|---|---|
| I 00 | D 10 | 5 | Circle | E N |
| I 01 | D 11 | 8 | C | E |
| I 02 | D 12 | 10 | C | E |
| I 03 | D 13 | 12 | C | E |
| I 04 | D 14 | 15 | C | E |
| I 05 | D 15 | 20 | C | E |
| I 06 | D 16 | 25 | C | E |
| I 07 | D 17 | 30 | C | E |
| I 08 | D 18 | 32 | C | E |
| I 09 | D 19 | 40 | C | E |
| I 10 | D 20 | 50 | C | E |
| I 11 | D 21 | 55 | C | E |
| I 12 | D 22 | 60 | C | E |
| I 13 | D 23 | 62.5 | C | E |
| I 14 | D 24 | 65 | C | E |
| I 15 | D 25 | 70 | C | E |
| I 16 | D 26 | 75 | C | E |
| I 17 | D 27 | 80 | C | E |
| I 18 | D 28 | 85 | C | E |
| I 19 | D 29 | 90 | C | E |
| I 20 | D 30 | 100 | C | E |
| I 21 | D 31 | 110 | C | E |
| I 22 | D 32 | 125 | C | E |
| I 23 | D 33 | 150 | C | E |
| I 24 | D 34 | 200 | C | E |
| I 25 | D 35 | 10 | Square | F |
| I 26 | D 36 | 20 | S | F |
| I 27 | D 37 | 30 | S | F |
| I 28 | D 38 | 40 | S | F |
| I 29 | D 39 | 50 | S | F |
| I 30 | D 40 | 60 | S | F |
| I 31 | D 41 | 75 | S | F |
| I 32 | D 42 | 100 | S | F |
| I 33 | D 43 | 125 | S | F |
| I 34 | D 44 | 150 | S | F |
| I 35 | D 45 | 100 / 80 | Annulus | F |
| I 36 | D 46 | 85 / 60 | A | F |
| I 37 | D 47 | 75 / 50 | A | F |
| I 38 | D 48 | 25 X 40 | Rectangle | F |
| I 39 | D 49 | 40 X 25 | R | F |
| I 40 | D 50 | 125 / 13 | Annulus | F N |
| I 41 | D 51 | 100 / 13 | A | F N |
| I 42 | D 52 | 75 / 13 | A | F N |
| I 43 | D 53 | 62.5 / 13 | A | F N |
| I 44 | D 54 | 50 / 13 | A | F N |
| I 45 | D 55 | 59 / 13 | A | F N |
| I 46 | D 56 | 40 / 20 | A | F N |
| I 47 | D 57 | 32 / 16 | A | F N |

Straight lines can be drawn between any two points (i.e. at any angle), hence only circular images should be used for line drawing. True circles or arcs can only be plotted from EMMA code.

# G Interaction Between Design Rules and Grid Size

The auto-router is a gridded router. Because the router places tracks in the centre of the grid cells, the best choice of design rules is not always obvious. For example, it should be possible to get a 10 thou track between two 60 thou pads which are 100 thou apart (the standard distance bewteen pins on an dilic) with a track to pad clearance of 15 thou. However, because of the gridded nature of the tracker, in fact **no** tracks will be placed between such pins. The following tables contain a wealth of such information for those interested in exploring non standard design-rules. A distillation of these tables is contained in Section 5.2, page 16: Choosing Design Rules.

If you intend to use MUCS for the fabrication of your printed circuit boards, please remember that the facility is not able to reliably manufacture boards with lines widths $< 10$ thou or *actual* clearances $< 7.5$ thou. The smallest via size recommended is 30 thou. A commercial manufacturer may be able to accommodate tighter design rules, but the cost is likely to be prohibitive.

Table 27 details the closest distance that a track can pass to a pad of a given size and the actual clearances obtained for nominal clearances. The spacing is measure from centre of the pad to the centre of the track and is therefore a measure of the packing density. The clearance is measured from the perimeter of the pad to the edge of the track. The figures are listed in the form spacing/clearance.

| Design Rules | | | Grid Size | | | | |
|---|---|---|---|---|---|---|---|
| pad size | track width | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 10 | 5 | 25/5 | 30/10 | 25/5 | 40/20 | 25/5 |
| 32 | 10 | 7.5 | 30/9 | 30/9 | 37.5/16.5 | 40/19 | 50/29 |
| 40 | 10 | 5 | 30/5 | 30/5 | 37.5/12.5 | 40/15 | 50/25 |
| 40 | 10 | 7.5 | 35/10 | 40/15 | 37.5/12.5 | 40/15 | 50/25 |
| 40 | 10 | 10 | 35/10 | 40/15 | 37.5/12.5 | 40/15 | 50/25 |
| 50 | 10 | 7.5 | 40/10 | 40/10 | 37.5/7.5 | 40/10 | 50/20 |
| 50 | 10 | 10 | 40/10 | 40/10 | 50/20 | 40/10 | 50/20 |
| 55 | 10 | 7.5 | 40/7.5 | 40/7.5 | 50/17.5 | 40/7.5 | 50/17.5 |
| 55 | 10 | 10 | 45/12.5 | 50/17.5 | 50/17.5 | 60/27.5 | 50/17.5 |
| 55 | 15 | 10 | 45/10 | 50/15 | 50/15 | 60/25 | 50/15 |
| 60 | 10 | 10 | 45/10 | 50/15 | 50/15 | 60/25 | 50/15 |
| 60 | 15 | 10 | 50/12.5 | 50/12.5 | 50/12.5 | 60/22.5 | 50/12.5 |
| 62.5 | 15 | 10 | 50/11.25 | 50/11.25 | 50/11.25 | 60/21.25 | 50/11.25 |

Table 27: Closest Orthogonal Pad or Via to track Spacings & Actual Clearances

Table 28 details the spacings and actual clearances when planting two pads side by side. It is necessary not

| Design Rules | | Grid Size | | | | |
|---|---|---|---|---|---|---|
| pad size | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 5 | 35/5 | 40/10 | 37.5/7.5 | 40/10 | 50/20 |
| 32 | 7.5 | 40/8 | 40/8 | 50/18 | 40/8 | 50/18 |
| 32 | 10 | 45/13 | 50/18 | 50/18 | 60/28 | 50/18 |
| 40 | 7.5 | 50/10 | 50/10 | 50/10 | 60/20 | 50/10 |
| 40 | 10 | 50/10 | 50/10 | 50/10 | 60/20 | 50/10 |
| 50 | 7.5 | 60/10 | 60/10 | 62.5/12.5 | 60/10 | 75/25 |
| 50 | 10 | 60/10 | 60/10 | 62.5/12.5 | 60/10 | 75/25 |
| 55 | 7.5 | 65/10 | 70/15 | 62.5/12.5 | 80/25 | 75/20 |
| 55 | 10 | 65/10 | 70/15 | 75/20 | 80/25 | 75/20 |
| 60 | 10 | 70/10 | 70/10 | 75/15 | 80/20 | 75/15 |

Table 28: Closest Via-to-Via Spacings & Actual Clearances

just to take into account the spacings orthogonally, but also to consider how close pads and tracks may be placed together in the diagonal directions i.e. NW, N,E SE or SW from the centre of a feature. Tables 29 and 30 tabulate this information; the spacings are **not** measured diagonally, but rather orthogonally even though they refer to features occupying diagonally separated grid cells. Another parameter of importance is the number of tracks that

| Design Rules | | | Grid Size | | | | |
|---|---|---|---|---|---|---|---|
| pad size | track width | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 10 | 5 | 20 | 20 | 25 | 20 | 25 |
| 32 | 10 | 7.5 | 25 | 30 | 25 | 40 | 25 |
| 40 | 10 | 5 | 25 | 30 | 25 | 40 | 25 |
| 40 | 10 | 7.5 | 25 | 30 | 25 | 40 | 25 |
| 40 | 10 | 10 | 30 | 30 | 37.5 | 40 | 50 |
| 50 | 10 | 7.5 | 30 | 30 | 37.5 | 40 | 50 |
| 50 | 10 | 10 | 30 | 30 | 37.5 | 40 | 50 |
| 55 | 10 | 7.5 | 30 | 30 | 37.5 | 40 | 50 |
| 55 | 10 | 10 | | | | | |
| 55 | 15 | 10 | 35 | 40 | 37.5 | 40 | 50 |
| 60 | 10 | 10 | 35 | 40 | 37.5 | 40 | 50 |
| 60 | 15 | 10 | 40 | 40 | 37.5 | 40 | 50 |
| 62.5 | 15 | 10 | 40 | 40 | 37.5 | 40 | 50 |

Table 29: Closest Diagonal Pad or Via to track Spacings

| Design Rules | | Grid Size | | | | |
|---|---|---|---|---|---|---|
| pad size | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 5 | 25 | 30 | 25 | 40 | 25 |
| 32 | 7.5 | 30 | 30 | 37.5 | 40 | 50 |
| 32 | 10 | 30 | 30 | 37.5 | 40 | 50 |
| 40 | 7.5 | 35 | 40 | 37.5 | 40 | 50 |
| 40 | 10 | 40 | 40 | 37.5 | 40 | 50 |
| 50 | 7.5 | 45 | 50 | 50 | 60 | 50 |
| 50 | 10 | 45 | 50 | 50 | 60 | 50 |
| 55 | 7.5 | 45 | 50 | 50 | 60 | 50 |
| 55 | 10 | 50 | 50 | 50 | 60 | 50 |
| 60 | 10 | 50 | 50 | 50 | 60 | 50 |

Table 30: Closest Diagonal Via-Via Spacings

will fit between pads of a given size arranged on a fixed pitch - for example between the pins of a dilic or the pins of an edge connector. Most components come on a 100 thou pitch, but there is an increasing trend to high-density components with a pitch of 75 thou.

Tables 31 and 32 detail the number of tracks that can be placed between pads arranged on a standard pitch. Notice that reducing the grid size from 25 thou to 20 thou actually makes the routablility worse if an inappropriate pad size is used!

Care should be exercised when using double or treble Eurocards. The correct position for the 2nd connector (top for a 6U board, middle for a 9U board) is not centred on a 20 thou grid. This affects the number of tracks that can run between the pins. Table 33 compares the number of tracks that can be run between the pins of a typical edge connector in the cases when the connector is on the 20 thou grid or off the 20 thou grid.

| Design Rules | | | Grid Size | | | | |
|---|---|---|---|---|---|---|---|
| pad size | track width | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 10 | 5 | 4 | 3 | 3 | 2 | 3 |
| 32 | 10 | 7.5 | 3 | 3 | 2 | 2 | 1 |
| 40 | 10 | 5 | 3 | 3 | 2 | 2 | 1 |
| 40 | 10 | 7.5 | 2 | 2 | 2 | 2 | 1 |
| 40 | 10 | 10 | 2 | 2 | 2 | 2 | 1 |
| 50 | 10 | 7.5 | 2 | 2 | 2 | 2 | 1 |
| 50 | 10 | 10 | 2 | 2 | 1 | 2 | 1 |
| 55 | 10 | 7.5 | 2 | 2 | 1 | 2 | 1 |
| 55 | 10 | 10 | 1 | 1 | 1 | **0** | 1 |
| 60 | 10 | 7.5 | 1 | 1 | 1 | **0** | 1 |
| 60 | 10 | 10 | 1 | 1 | 1 | **0** | 1 |

Table 31: Number of tracks between pads 100 thou apart

| Design Rules | | | Grid Size | | | | |
|---|---|---|---|---|---|---|---|
| pad size | track width | nominal clearance | 5 | 10 | 12.5 | 20 | 25 |
| 30 | 10 | 5 | 2 | 2 | 2 | 1 | 2 |
| 32 | 10 | 7.5 | 1 | 1 | 1 | 1 | 0 |
| 40 | 10 | 5 | 2 | 1 | 1 | 1 | 0 |
| 40 | 10 | 7.5 | 1 | 1 | 1 | 1 | 0 |
| 40 | 10 | 10 | 1 | 1 | 1 | 1 | 0 |
| 50 | 10 | 7.5 | 0 | 0 | 1 | 0 | 0 |

Table 32: Number of tracks between pads 75 thou apart

| grid | clearance | on 20-thou grid | off 20-thou grid |
|---|---|---|---|
| 20 | 7.5 | 2 | 1 |
| 20 | 10 | 0 | 1 |
| 10 | 7.5 | 2 | 2 |
| 10 | 10 | 1 | 1 |

Table 33: No. of 10 thou tracks between 55 thou pads, on and off a 20 thou grid

# H Examples

## H.1 Introduction

The examples included here contain a number of real boards that have been designed using the MUCS system. They are:

**mainboard** This board, although large (14.4in x 15in) and containing over 175 components, is the easiest to route. The board is preplaced.

**memboard**

**madarch**

**timu**

This appendix contains copies of the README files in *doc/examples/**. The actual files may be more up to date than the listing included here.

## H.2 Mainboard

```
This board is a real design created using the MUCS PCB system. It is
actually a microcoded processor and thus contains a fair amount of
random logic and wide data buses. The system is laid out on an
extended 9U card. The board is the easiest to route of the 3 examples
supplied with the system.

The directory initially contains:
README - this document
mainboard   - an ncap description of the board
outlines - A directory containing oulines particular to this board
layout          - A directory containing postscript plots of a routed
                  version of this board.

THIS EXAMPLE USES DEVICES FROM AN OBSOLETE VERSION OF THE DEVICE
OUTLINE LIBRARY CONTAINED IN THE OUTLINES DIRECTORY.

Copy the directory to somewhere convenient in your user filestore and
change directory there.

Try running "ncap" and typing "l" to get a listing of
the board on the screen. You can view the layout of the board by:

place mainboard


To get an idea of how the routers perform first extract the netlists
using netex:
netex mainboard.

Note the directories mainboard.ddir and mainboard..ndir have been
created.

A picture of the board can be obtained from:
cd mainboard.ddir
cat outlines names  | lpr -g -P<postscript printer>

Alternatively, the utility "xplot" cat be used:
cat outlines names  | xplot.
```

```
Change back to the "mainboard" directory:
cd ..

Now try running the default tracker;
track -tdvF mainboard

The "-F" option to "track" reports the netnames of those tracks wich
could not be routed. The -t options sends the reporting information to
the screen rather than accumulating it in a file "mainboard.mon"

The tracked board can be viewed:
cd mainboard.ddir
cat pads.l1 lines.l1 | lpr -g -P<postscript printer>
cat pads.l4 lines.l4 | lpr -g -P<postscript printer>
cd ..
(Alternatively, the xplot utility may used to view the board on the
screen).


You may wish to run some of the other routers available through the
command lines options. Typical execution times for a SUN Sparcstation
5IPC are given below. (times on a Sun IPC are ~x3.3 longer.

Note that the global routers are non-deterministic and thus there may
be some variation in the execution times and the results. The extreme
vaulues found over several runs are quoted below.

RESULTS: 25 thou grid size (Default Desing Rules)
-n:      5min 25 secs   6 fails
-m:      5min 15 secs   6 fails (this is the default router)

-mg:     33min 25sec    1 fails
-g:      47min 33sec    3 fails


-ma:     33min 25sec    1 fails
-a:      29min 52sec    0 fails

Because of the non-deterministic nature of the rip-up routers,
different solutions may be obtained from those presented above.


 ********************

Try running the board with a 50 thou grid size. First install the
design-rule file:

cp $MUCS_PCB_DIR/design_tools/lib/boards/rules_50 mainboard.rules
netex -f


RESULTS: 50 thou grid size:
-m:      1min 32sec     25 fails
-mg:     6min 35sec     5 fails
-g:     10min 05sec     2 fails
-ma:    10min 27sec     1 fails
```

```
-a       6min 23sec     0 fails
```

The output of routers can be re-annealed by saving the solution state using the "-p" option:

```
track -p mainboard                # preserve solution
track -gp mainboard               # try again (2 fails)
track -a mainboard                # and again (0 fails)
rm mainboard.ndir/tracker-init    # remove tmp solution
```

The re-annealing took approx 6min.

Postscript plots of the board tracked on a 50thou grid can be found in the directory "layout". They were produced from the "plot" format files using the utility "psplot"

Note that attempting to track the board using the tighter design rules found in rules_C, rules_D, rules_E, rules_F will result worse results because the outlines used in this example have 60 thou pads.

## H.3 Memboard

This example is  harder to route than "mainboard". Again it is a real design on an extended 9U Eurocard board. Typical timings and behaviours are given below. Although the chip density may not look high, this board exhibits a high degree of interconnectivity as can be seen by viewing the sample plots in the memboard.ddir directory.

THIS EXAMPLE USES DEVICES FROM AN OBSOLETE VERSION OF THE DEVICE OUTLINE LIBRARY CONTAINED IN THE OUTLINES DIRECTORY.

Note some warning messages are produced when running "netex". These may safely be ignored.

```
RESULTS: - 25thou grid
option hr:min:sec fails
-o:     0:11:50         39
-n:     0:11:6          26
-m:     0:12:02         26
-g:     4:16:32         12
-a:     4:25:02          1
-ma:    6:33:30          6     Note (1):
-af:    2:48:42          7     Note (2):
-r:     0:53:54          5     from previous solution
```

Notes:
1) This illustrates the behaviour when the standard annealing parameters are not correct. The solution diverges from the intermediate one found by the standard tracker, taking a long time to come back to equilibrium.

2) The use of the -f option causes the temperature to decline more rapidly to a state where fewer uphill moves are accepted. The tracking time is thus much shorter, but a poorer result may be obtained.

## H.4 Madarch

This is a particularly difficult board. You need at least a 16 Mybte machine to run the annealed router happily on. (The serial routers run OK in much less), but give too many failures to be useable.

The example illustrates the use of some of the more obscure features of the tracker input format, particularly the use of selection records to pre-route important tracks. Being familar with the formats of the various records in the trackeript file, particularly the job header record, layer direction, cost parameter and selection records would help in understanding this example. The record formats are detailed in Appendix A of the supplied documentation.

The board is a 6 layer (4 tracking + 2 Power) 9U Eurocard with several Pin-Grid Array Devices. The design rules, defined in madarch.rules, are based on a 20 thou grid with 10 thou tracks and 50 thou pads/vias in order to get 2 tracks between pads on a 0.1inch pitch. Several specialised device outlines are needed; they are to be found in the outlines subdirectory. Most of these devices are now incorporated in the standard part library, although the part numbers have been changed.

The board is routed using shell script (track-madarch). The basic strategy is to first pre-route the sensitive clock-tracks with the cost penalties changed from the standard values in order to minimise any wrong way-movement as much as possible. The rest of the board is board is then routed.

Since this is a real design example, there are a few idosyncratic commands in the script which need explaining, so a walk-through follows:

After defining some convenient shell variables, (including OUTLINE_LIBS) and arranging for the shell to trap the hangup quit and terminate signals so that the temporary file that gets created is removed, the script invokes "sed" to change SIG_VCC and SIG_GND to VCC and GND. The reason for this is that the board was orginally specified using cuttable segments for VCC and GND connections, but it was felt that these consumed valuable board space, so the power and ground connections were changed to direct connections to the IC pins.

Next, "netex" is run to generate the trackeript file in madarch.ndir as normal.

The "perl" script "sel_gen" is then run to extract from the netlist the records pertaining to the sensitive clock tracks. The netlist information is held in the file "madarch.ndir/sig_no" which is produced by netex. The nets to be run first are listed in "madarch.sel1" are converted to selection records in tracker input format in the file madarch.sr1.

The selected nets next have to be incorporated into the trackeript file. At the same time the cost penalties are changed from the default to encourage as straight tracking as possible. Actually, the only cost which is changed is the wrong-direction penalty which is increased from 8 -> 15 (max). The maximum cost allocated to a cell at initialisation is also reduced to 3; this minimises the impact of pads on the chosen route. OK, this is a black-art, but all will be revealed in future releases of the manual/software.

At the same time the layer biasing field is changed to -1 on layers 5 and 6 to prohibit tracking on those layers. This is to ensure that the clock tracks keep to layers 1 & 2 to control their impedance and keep them as direct as possible. Finally a no-go area is defined to restrict tracking within a particular area. (Normally, the margin keyword in the design.rules file can be used to keep tracks away from the edge of the board, but in this case, a non rectangular border was required for fixing holes in the corners). All these changes are made using the call to "design-filter" (Note that this program changes the trackeript file header to specify that only the routes in the selection records should be attempted).

The board is then tracked; because of the selection records ONLY the clock tracks are routed.

The next step is is to remove the trackeript˜ file. The reason for this is explained after the rest of the script has been run through.

The next stage of the tracking process is  to route the rest of the nets. Firstly the layer-biases are redfined back to the default, and the no-go area defined using a call to "design-filter". The selection records have disappeared from the trackeript file because they were successfully routed in the previous stage, and the cost penalties get reset to their default by "design-filter" (no cost penalties having been specified when calling "design-filter").

The board is now routed again using a cascade of the default serial router, followed by the greedy router, followed by the annealed router. The annealing parameters are changed from the default by the file madarch.ap in order to get a convergent behaviour. If the default parameters are used, the initial good solution arrived at by the greedy router gets gradually worse for many iterations before slowly converging again. (Try it)

The final action is to add a track in the power layer (thereby splitting it into 2 sections) so that it is fused. The file madarch.pre contains this information and was hacked by hand.

It is important to understand why the trackeript˜ file is removed prior to running "track" again. Running "netex -f" at the beginning of the script created a new trackeript file and removed any previous trackeript˜ file. The first invocation of "track", to route the clock signals, created a trackeript˜ file as a copy of the original trackeript file. It thus contained no physical routes, only logical netlists. "track", after attempting to route a board, merges trackeript˜ and the output file (trackeropt) produced by the routing process to create a new version of the trackeript file (which then contains the newly routed clock signals). However, "track" will not change trackeript˜ if one already exists, and the trackeropt file only contains information on newly routed nets (& failures). So, if trackeript˜ had not been removed, the second call to "track" would have not have updated trackeript˜. Thus the new trackeript file, being a merging of the orginal trackeript˜ and the new routes in the new trackeropt file, would lose the information about the clock tracks that had been placed in the first phase. By removing trackeript˜ before calling "track" the second time, a new trackeript˜ is created which does contain the pre-routed clock tracks. Thus when in the newly tracked nets are merged with the trackeript˜ file, the resulting trackeript contains ALL the routed nets, both from this pass and the previous pass. OK!!

The first phase of (-m) routing gives 186 failures out of 1813 (90%)
The greedy router (-g) gets down to 2 failures
The annealing phase (-a) gets down to 1 failure.
The total time spent tracking is 25hrs on an SUN IPC.

It is possible to achieve zero failures, try reannealing the solution.

# I References


**[Spie83]** Spiers T.D *An investigation of Lee's Algorithm with regard to its implementation by hardware,*
M.Sc. dissertation, Manchester, 1983.

**[Spie85]** Spiers T.D *A Hardware Assisted Routing System,*
Phd dissertation, Manchester, 1985.

**[Spie87]** Spiers T. D. and Edwards D.A. *A High Performance Routing Engine,*
24th ACM/IEEE Design Automation Conference, Miami 1987.

**[Moos92]** Moosa Z. *Routing Algorithms,*
Phd dissertation, To be published