



***Intelligent Schematic  
Input System***

# **User Manual**

Issue 6.0 - November 2002

© Labcenter Electronics



# ICON REFERENCE CHART

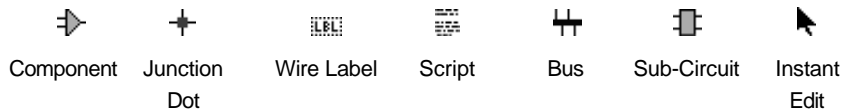
## File And Printing Commands



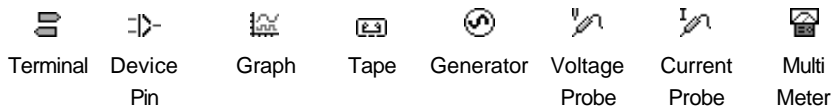
## Display commands



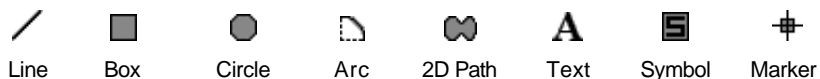
## Main Mode Icons



## Gadget Icons



## 2D Graphics



## Design Tools

  
Real Time Snap

  
Wire Autorouter

  
Search & Tag

  
Property Assignment Tool

  
New Sheet

  
Delete Sheet

  
Goto Sheet

  
Zoom to Child

  
Return to Parent

  
Bill of Material

  
Electrical Rules Check

  
Netlist to Ares

## Editing Commands

*These affect all currently tagged objects.*

  
Block Copy

  
Block Move

  
Block Delete

  
Block Rotate

  
Pick Device/Symbol

  
Make Device Decompose

  
Package Tool

  
Undo

  
Redo

  
Cut

  
Copy

  
Paste

## Rotate And Mirror Icons

  
Rotate Clockwise

  
Rotate Anti-clockwise

  
Flip X axis

  
Flip Y axis

# TABLE OF CONTENTS

<b>INTRODUCTION.....</b>	<b>1</b>
ABOUT ISIS.....	1
ISIS AND PCB DESIGN.....	2
ISIS AND SIMULATION.....	2
ISIS AND NETWORKS.....	4
HOW TO USE THIS DOCUMENTATION.....	4
<b>TUTORIAL .....</b>	<b>5</b>
INTRODUCTION .....	5
A GUIDED TOUR OF THE ISIS EDITOR.....	5
PICKING, PLACING AND WIRING UP COMPONENTS .....	7
LABELLING AND MOVING PART REFERENCES .....	9
BLOCK EDITING FUNCTIONS .....	10
PRACTICE MAKES PERFECT .....	10
ANNOTATING THE DIAGRAM.....	11
The Property Assignment Tool (PAT).....	11
The Automatic Annotator.....	12
CREATING NEW DEVICES.....	13
FINISHING TOUCHES .....	15
SAVING, PRINTING AND PLOTTING.....	16
MORE ABOUT CREATING DEVICES.....	16
Making a Multi-Element Device.....	16
The Visual Packaging Tool.....	18
Making Similar Devices.....	20
Replacing Components in a design.....	20
SYMBOLS AND THE SYMBOL LIBRARY .....	20
REPORT GENERATION.....	21
A LARGER DESIGN.....	21
<b>GENERAL CONCEPTS .....</b>	<b>23</b>
SCREEN LAYOUT .....	23
The Menu Bar.....	23
The Toolbars.....	23
The Editing Window .....	24
The Overview Window.....	25
The Object Selector.....	26

Co-ordinate Display .....	26
CO-ORDINATE SYSTEM .....	26
False Origin .....	27
The Dot Grid.....	27
Snapping to a Grid .....	27
Real Time Snap .....	27
FILING COMMANDS .....	28
Starting a New Design .....	28
Loading a Design .....	28
Saving the Design .....	29
Import / Export Section.....	29
Quitting ISIS.....	29
GENERAL EDITING FACILITIES .....	29
Object Placement.....	29
Tagging an Object .....	30
Deleting an Object .....	30
Dragging an Object .....	30
Dragging an Object Label.....	31
Resizing an Object.....	31
Reorienting an Object .....	31
Editing an Object .....	32
Editing an Object Label .....	33
Copying all Tagged Objects .....	33
Moving all Tagged Objects.....	34
Deleting all Tagged Objects.....	34
WIRING UP.....	34
Wire Placement.....	34
The Wire Auto-Router.....	35
Wire Repeat.....	35
Dragging Wires .....	36
THE AUTOMATIC ANNOTATOR .....	37
Value Annotation .....	37
MISCELLANEOUS .....	37
The Sheet Border.....	37
The Header Block.....	38
Send to Back / Bring to Front.....	40
<b>GRAPHICS AND TEXT STYLES .....</b>	<b>41</b>
INTRODUCTION .....	41
TUTORIAL.....	42
Editing Global Styles.....	42
Editing Local Styles.....	44

Working With The Template.....	45
TEMPLATES AND THE TEMPLATE MENU.....	46
<b>PROPERTIES .....</b>	<b>47</b>
INTRODUCTION .....	47
OBJECT PROPERTIES.....	47
System Properties .....	47
User Properties .....	48
Property Definitions (PROPDEFS) .....	48
SHEET PROPERTIES .....	49
Introduction.....	49
Defining Sheet Properties.....	49
Scope Rules for Sheet Properties .....	50
DESIGN PROPERTIES .....	51
PARAMETERIZED CIRCUITS.....	51
Introduction.....	52
An Example.....	52
Property Substitution .....	53
Property Expression Evaluation .....	54
The Rounding Functions E12 (), E24 ().....	55
THE PROPERTY ASSIGNMENT TOOL.....	55
The PAT Dialogue Form.....	56
PAT Actions .....	56
PAT Application Modes .....	58
The Search and Tag Commands .....	58
Examples .....	59
PROPERTY DEFINITIONS .....	60
Creating Property Definitions.....	60
Default Property Definitions.....	61
Old Designs.....	61
<b>OBJECT SPECIFICS.....</b>	<b>63</b>
COMPONENTS .....	63
Selecting Components from the Device Libraries .....	63
Placing Components.....	64
Replacing Components .....	65
Editing Components.....	65
Component Properties .....	66
Hidden Power Pins.....	66
DOTS.....	66
Placing Dots .....	67
Auto Dot Placement .....	67
Auto Dot Removal.....	67

WIRE LABELS.....	67
Placing And Editing Wire Labels .....	67
Deleting Wire Labels .....	68
Using a Wire Label to Assign a Net Name.....	69
Using a Wire Label to Assign a Net Property.....	69
Wire Label Properties.....	69
SCRIPTS .....	70
Placing and Editing Scripts.....	70
Script Block Types .....	71
Part Property Assignments (*FIELD).....	71
Sheet Global Net Property Assignments (*NETPROP).....	72
Sheet Property Definitions (*DEFINE) .....	72
Parameter Mapping Tables (*MAP ON varname).....	73
Model Definition Tables (*MODELS).....	73
Named Scripts (*SCRIPT scripttype scriptname).....	74
SPICE Model Script (*SPICE) .....	75
BUSES .....	75
Placing Buses.....	75
Bus Labels .....	76
Wire/Bus Junctions .....	77
Bus Properties.....	78
SUB-CIRCUITS.....	78
Placing Sub-Circuits .....	78
Editing Sub-Circuits .....	80
Sub-Circuit Properties.....	80
TERMINALS.....	80
Logical Terminals .....	80
Physical Terminals .....	81
Placing Terminals .....	81
Editing Terminals .....	82
Terminal Properties.....	83
PIN OBJECTS.....	83
Placing Pin Objects.....	83
Editing Pin Objects.....	84
Pin Object Properties .....	84
SIMULATOR GADGETS.....	85
2D GRAPHICS .....	85
Placing 2D Graphics .....	85
Resizing 2D Graphics .....	87
Editing 2D Graphics.....	88
MARKERS .....	88
Marker Types .....	88



Placing Markers.....	89
<b>LIBRARY FACILITIES .....</b>	<b>91</b>
GENERAL POINTS ABOUT LIBRARIES.....	91
Library Discipline .....	91
The Pick Command.....	91
SYMBOL LIBRARIES.....	92
Graphics Symbols .....	92
User Defined Terminals .....	93
User Defined Module Ports.....	94
User Defined Device Pins .....	95
Editing an Existing Symbol.....	96
Hierarchical Symbol Definitions .....	96
DEVICE LIBRARIES .....	96
Making a Device Element.....	97
The Make Device Command.....	100
The Visual Packaging Tool.....	105
Making a Single Element Device .....	108
Making a Multi-Element Homogenous Device.....	109
Making a Multi-Element Heterogeneous Device .....	110
Making a Device with Bus Pins.....	111
Property Definitions and Default Properties.....	112
Dealing with Power Pins.....	113
Editing an Existing Device.....	114
<b>MULTI-SHEET DESIGNS .....</b>	<b>117</b>
MULTI-SHEET FLAT DESIGNS.....	117
Introduction.....	117
Design Menu Commands.....	117
HIERARCHICAL DESIGNS.....	117
Introduction.....	117
Terminology .....	118
Sub-Circuits.....	119
Module-Components.....	120
External Modules .....	121
Moving About a Hierarchical Design .....	121
Design Global Annotation.....	122
Non-Physical Sheets .....	122
<b>NETLIST GENERATION .....</b>	<b>123</b>
INTRODUCTION .....	123
NET NAMES.....	123
DUPLICATE PIN NAMES .....	124

HIDDEN POWER PINS .....	124
SPECIAL NET NAME SYNTAXES .....	125
Global Nets .....	125
Inter-Element Connections for Multi-Element Parts .....	126
BUS CONNECTIVITY RULES .....	127
The Base Alignment Rule.....	127
Using Bus Labels to Change the Connectivity Rule .....	127
Using Bus Terminals to Interconnect Buses .....	128
Connections to Individual Bits.....	129
Tapping a Large Bus .....	131
General Comment & Warning .....	131
GENERATING A NETLIST FILE .....	132
Format .....	132
Logical/Physical/Transfer .....	132
Scope.....	132
Depth .....	132
Errors.....	133
NETLIST FORMATS.....	133
SDF .....	133
BOARDMAKER .....	133
EEDESIGNER .....	133
FUTURENET .....	133
MULTIWIRE .....	133
RACAL.....	134
SPICE.....	134
SPICE-AGE FOR DOS .....	134
TANGO .....	134
VALID .....	134
VUTRAX.....	134
<b>REPORT GENERATION .....</b>	<b>135</b>
BILL OF MATERIALS .....	135
Generating the Report .....	135
Bill of Materials Configuration .....	135
ASCII DATA IMPORT .....	136
The IF...END Command.....	136
The DATA...END Command.....	138
ELECTRICAL RULES CHECK.....	140
Generating the Report .....	140
ERC Error Messages .....	140
<b>HARD COPY GENERATION .....</b>	<b>143</b>
PRINTER OUTPUT .....	143

PLOTTER OUTPUT .....	143
Plotter Pen Colours .....	143
CLIPBOARD AND GRAPHICS FILE GENERATION .....	143
Bitmap Generation .....	144
Metafile Generation .....	144
DXF File Generation .....	144
EPS File Generation .....	144
<b>ISIS AND ARES .....</b>	<b>145</b>
INTRODUCTION .....	145
PACKAGING .....	145
Default Packaging .....	145
Manual Packaging .....	146
Automatic Packaging .....	146
Using the Bill of Materials to Help with Packaging .....	147
The Package Verifier .....	148
Packaging with ARES .....	148
NET PROPERTIES AND ROUTING STRATEGIES .....	148
FORWARD ANNOTATION - ENGINEERING CHANGES .....	149
Adding New Components .....	149
Removing Existing Components .....	150
Changing the connectivity .....	150
Re-Annotating Components, and Re-Packaging Gates .....	151
PIN-SWAP/GATE-SWAP .....	151
Specifying Pin-Swaps and Gate-Swaps for ISIS Library Parts .....	152
Specifying Pin-Swaps in Single Element Devices .....	152
Specifying Pin-Swaps in Multi-Element Devices .....	153
Specifying Gate-Swaps in Multi-Element Devices .....	153
Performing Manual Pin-Swaps and Gate-Swaps in ARES .....	154
The Gate-Swap Optimizer .....	155
RE-ANNOTATION .....	156
BACK-ANNOTATION WITH ISIS .....	157
Semi-Automatic Back-Annotation .....	157
Fully-Automatic Back-Annotation .....	157
<b>INDEX .....</b>	<b>159</b>



# INTRODUCTION

## ***ABOUT ISIS***

Many CAD users dismiss schematic capture as a necessary evil in the process of creating PCB layout but we have always disputed this point of view. With PCB layout now offering automation of both component placement and track routing, getting the design into the computer can often be the most time consuming element of the exercise. And if you use circuit simulation to develop your ideas, you are going to spend even more time working on the schematic.

ISIS has been created with this in mind. It has evolved over twelve years research and development and has been proven by thousands of users worldwide. The strength of its architecture has allowed us to integrate first conventional graph based simulation and now – with PROTEUS VSM – interactive circuit simulation into the design environment. For the first time ever it is possible to draw a complete circuit for a micro-controller based system and then test it interactively, all from within the same piece of software. Meanwhile, ISIS retains a host of features aimed at the PCB designer, so that the same design can be exported for production with ARES or other PCB layout software.

For the educational user and engineering author, ISIS also excels at producing attractive schematics like you see in the magazines. It provides total control of drawing appearance in terms of line widths, fill styles, colours and fonts. In addition, a system of templates allows you to define a ‘house style’ and to copy the appearance of one drawing to another.

Other general features include:

- Runs on Windows 98/Me/2k/XP and later.
- Automatic wire routing and dot placement/removal.
- Powerful tools for selecting objects and assigning their properties.
- Total support for buses including component pins, inter-sheet terminals, module ports and wires.
- Bill of Materials and Electrical Rules Check reports.
- Netlist outputs to suit all popular PCB layout tools.

For the 'power user', ISIS incorporates a number of features which aid in the management of large designs. Indeed, a number of our customers have used it to produce designs containing many thousands of components.

- Hierarchical design with support for parameterized component values on sub-circuits.
- Design Global Annotation allowing multiple instances of a sub-circuit to have different component references.
- Automatic Annotation - the ability to number the components automatically.
- ASCII Data Import - this facility provides the means to automatically bring component stock codes and costs into ISIS design or library files where they can then be incorporated or even totalled up in the Bill of Materials report.

## ***ISIS AND PCB DESIGN***

Users of ARES, or indeed other PCB software will find some of the following PCB design specific features of interest:

- Sheet Global Net Properties which allow you to efficiently define a routing strategy for all the nets on a given sheet (e.g. a power supply needing POWER width tracks).
- Physical terminals which provide the means to have the pins on a connector scattered all over a design.
- Support for heterogeneous multi-element devices. For example, a relay device can have three elements called RELAY:A, RELAY:B and RELAY:C. RELAY:A is the coil whilst elements B and C are separate contacts. Each element can be placed individually wherever on the design is most convenient.
- Support for pin-swap and gate-swap. This includes both the ability to specify legal swaps in the ISIS library parts and the ability to back-annotate changes into a schematic.
- A visual packaging tool which shows the PCB footprint and its pin numbers alongside the list of pin names for the schematic part. This facilitates easy and error free assignment of pin numbers to pin names. In addition, multiple packagings may be created for a single schematic part.

A full chapter is provided on how to use ISIS and ARES together.

## ***ISIS AND SIMULATION***

ISIS provides the development environment for PROTEUS VSM, our revolutionary interactive system level simulator. This product combines mixed mode circuit simulation,

micro-processor models and interactive component models to allow the simulation of complete micro-controller based designs.

ISIS provides the means to enter the design in the first place, the architecture for real time interactive simulation and a system for managing the source and object code associated with each project. In addition, a number of graph objects can be placed on the schematic to enable conventional time, frequency and swept variable simulation to be performed.

Major features of PROTEUS VSM include:

- True Mixed Mode simulation based on Berkeley SPICE3F5 with extensions for digital simulation and true mixed mode operation.
- Support for both interactive and graph based simulation.
- CPU Models available for popular microcontrollers such as the PIC and 8051 series.
- Interactive peripheral models include LED and LCD displays, a universal matrix keypad, an RS232 terminal and a whole library of switches, pots, lamps, LEDs etc.
- Virtual Instruments include voltmeters, ammeters, a dual beam oscilloscope and a 24 channel logic analyser.
- On-screen graphing - the graphs are placed directly on the schematic just like any other object. Graphs can be maximised to a full screen mode for cursor based measurement and so forth.
- Graph Based Analysis types include transient, frequency, noise, distortion, AC and DC sweeps and fourier transform. An Audio graph allows playback of simulated waveforms.
- Direct support for analogue component models in SPICE format.
- Open architecture for 'plug in' component models coded in C++ or other languages. These can be electrical., graphical or a combination of the two.
- Digital simulator includes a BASIC-like programming language for modelling and test vector generation.
- A design created for simulation can also be used to generate a netlist for creating a PCB - there is no need to enter the design a second time.

Full details of all these features and much more are provided in the PROTEUS VSM manual.

## ***ISIS AND NETWORKS***

ISIS is fully network compatible, and offers the following features to help Network Managers:

- Library files can be set to Read Only. This prevents users from messing with symbols or devices that may be used by others.
- ISIS individual user configuration in the windows registry. Since the registry determines the location of library files, it follows that users can have individual USERDVC.LIB files in their personal or group directories.

## ***HOW TO USE THIS DOCUMENTATION***

This manual is intended to complement the information provided in the on-line help. Whereas the manual contains background information and tutorials, the help provides context sensitive information related to specific icons, commands and dialog forms. Help on most objects in the user interface can be obtained by pointing with the mouse and pressing F1.

ISIS is a vast and tremendously powerful piece of software and it is unreasonable to expect to master all of it at once. However, the basics of how to enter a straightforward circuit diagram and create your own components *are* extremely simple and the techniques required for these tasks can be mastered most quickly by following the tutorial given in *Tutorial* on page 5. We strongly recommend that you work through this as it will save you time in the long run.

With some of the more advanced aspects of the package, you are probably going to find some of the *concepts* are new, let alone the details of how ISIS handles them. Each area of the software has been given a chapter of its own, and we generally start by explaining the background theory before going into the operation and use of the relevant features. You will thus find it well worthwhile reading the introductory sections rather than jumping straight to the how-to bits.



## ***INTRODUCTION***

The aim of this tutorial is to take you through the process of entering a circuit of modest complexity in order to familiarise you with the techniques required to drive ISIS. The tutorial starts with the easiest topics such as placing and wiring up components, and then moves on to make use of the more sophisticated editing facilities, such as creating new library parts.

For those who want to see something quickly, ISISTUT.DSN contains the completed tutorial circuit. This and other sample designs are installed to the SAMPLES directory.

## ***A GUIDED TOUR OF THE ISIS EDITOR***

We shall assume at this point that you have installed the package, and that the current directory is some convenient work area on your hard disk.

To start the ISIS program, click on the *Start* button and select *Programs, Proteus 6 Professional*, and then the *ISIS 6 Professional* option. The ISIS schematic editor will then load and run. Along the top of the screen is the Menu Bar.

The largest area of the screen is called the *Editing Window*, and it acts as a window on the drawing - this is where you will place and wire-up components. The smaller area at the top right of the screen is called the *Overview Window*. In normal use the *Overview Window* displays, as its name suggests, an overview of the entire drawing - the blue box shows the edge of the current sheet and the green box the area of the sheet currently displayed in the *Editing Window*. However, when a new object is selected from the *Object Selector* the *Overview Window* is used to preview the selected object - this is discussed later.

You can adjust the area of the drawing displayed in the *Editing Window* in a number of ways:

- To simply 'pan' the *Editing Window* up, down, left or right, position the mouse pointer over the desired part of the *Editing Window* and press the F5 key.
- Hold the SHIFT key down and bump the mouse against the edges of the *Editing Window* to pan up, down, left or right. We call this *Shift Pan*.
- Should you want to move the *Editing Window* to a completely different part of the drawing, the quickest method is to simply point at the centre of the new area on the *Overview Window* and click left.
- You can also use the *Pan* icon on the toolbar.

To adjust the scale the drawing is displayed at in the *Editing Window* you can:

- Point with the mouse where you want to zoom in or out and press the F6 or F7 keys respectively.
- Press the F8 key to display the whole drawing.
- Hold the SHIFT key down and drag out a box around the area you want to zoom in to. We call this *Shift Zoom*.
- Use the *Zoom In*, *Zoom Out*, *Zoom Full* or *Zoom Area* icons on the toolbar.

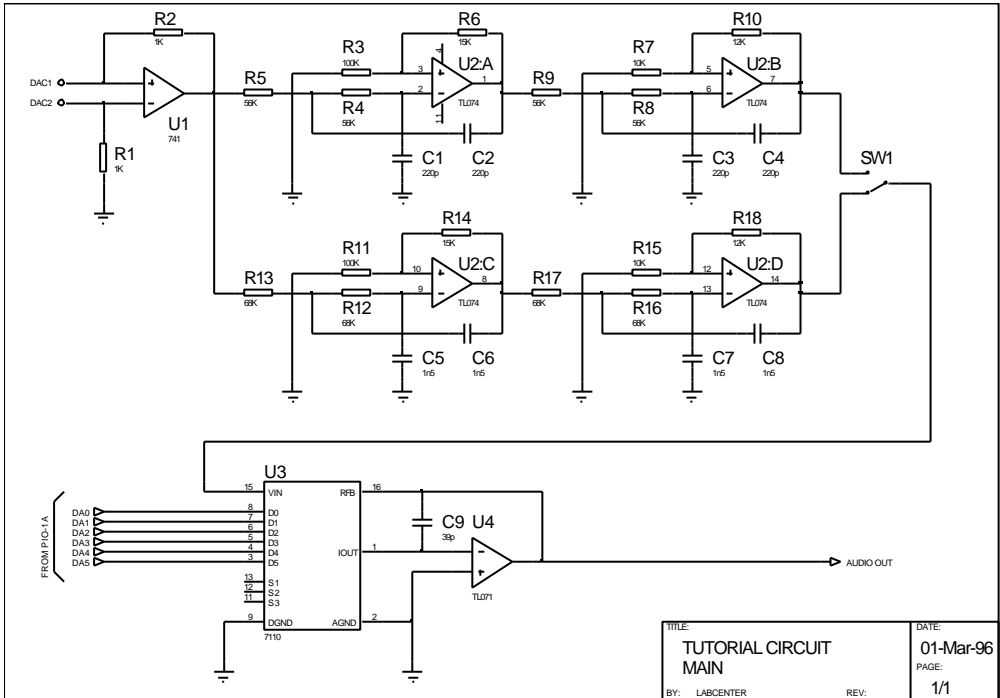
A grid of dots can be displayed in the *Editing Window* using the *Grid* command on the *View* menu, or by pressing 'G', or by clicking the *Grid* icon on the toolbar. The grid helps in lining up components and wires and is less intimidating than a blank screen. If you find it hard to see the grid dots, either adjust the contrast on your monitor slightly (by default the grid dots are displayed in light grey) or change their colour with the *Set Design Defaults* on the *Template* menu.

Below the *Overview Window* is the *Object Selector* which you use to select devices, symbols and other library objects.

At the bottom right of the screen is the co-ordinate display, which reads out the co-ordinates of the mouse pointer when appropriate. These co-ordinates are in 1 thou units and the origin is in the centre of the drawing.

## PICKING, PLACING AND WIRING UP COMPONENTS

The circuit we are going to draw is shown below. It may look quite a lot to do, but some parts of it are similar (the four op-amp filters to be precise) which will provide opportunity to use the block copying facilities.



*The Tutorial Circuit*

We shall start with the 741 buffer amplifier comprising U1, R1, R2. Begin by pointing at the **P** button at the top left of the *Object Selector* and clicking left. This causes the *Device Library Selector* dialogue form to appear and you can now select devices from the various device libraries. There are a number of selectors labelled *Objects*, *Libraries*, *Extensions* and a *Browser*, not all of which may be shown:

- The *Library* selector chooses which of the various device libraries (e.g. DEVICE, TTL, CMOS) you have installed is current.

- The Objects selector displays all the parts in the currently selected library according to the settings in the *Extensions* selector, if it is shown (see below). Click left once on a part to browse it or double-click a part to 'pick' it in to the design.
- The *Browser* displays the last selected part in the *Parts* selector as a means of browsing the contents of the library.

We need two devices initially - OPAMP for the 741 op-amp and RES for the feedback resistors. Both these are in the DEVICE library, so, if it is not already selected, start by selecting the DEVICE library in the *Library* selector. Then, double-click on the OPAMP and RES parts from the *Parts* selector to select each part. The devices you have picked should appear in the *Object Selector*.

Whenever you select a device in the *Devices* selector or use the *Rotation* or *Mirror* icons to orient the device prior to placement, the selected device is shown previewed in the *Overview Window* with the orientation the selected device will have if placed. As you click left or right on the *Rotation* and/or *Mirror* icons, the device is redrawn to preview the new orientation. The preview remains until the device is placed or until the another command or action is performed.

Ensure the OPAMP device is selected and then move the mouse pointer into the middle of the *Editing Window*. Press and hold down the left mouse button. An outline of the op-amp will appear which you can move around by dragging the mouse. When you release the button, the component will be *placed* and is drawn in full. Place the op-amp in the middle of the *Editing Window*. Now select the RES device and place one resistor just above the op-amp as in the diagram and click left once on the anti-clockwise *Rotation* icon; the preview of the resistor shows it rotated through 90°. Finally, place the second (vertical) resistor, R1 as before.

Unless you are fairly skilful, you are unlikely to have got the components oriented and positioned entirely to your satisfaction at this first attempt, so we will now look at how to move things around. In ISIS, objects are selected for further editing by 'tagging' them. Try pointing at the op-amp and clicking right. This tags the object, causing it to be highlighted. Now, still keeping the pointer over it, hold the left button down and drag it around. This is one of the ways to move objects. Release the left button, and click right on the op-amp a second time. Clicking right on a tagged object deletes it. Select *Undo* on the *Edit* Menu (or press 'U') to recover it. Tag it again, and click first left and then right on the *Rotation* icon whilst watching the op-amp itself. The rotation of the last object you tagged can be adjusted in this way; the *Mirror* icon can similarly be used to reflect the last object tagged. Armed with the above knowledge, you should now be able to adjust the three components you have placed such that they match the diagram. When you have finished editing, point at a free space in the *Editing Window* (i.e. somewhere where there is no object) and click right to untag all tagged objects.

We can now move on to place some wires. Start by pointing at the tip of the upper end of R1 and clicking left. ISIS senses that you are pointing at a component pin and deduces that you wish to connect a wire from it. To signify this, it displays a green line which goes from the pin to the pointer. Now point at the tip of the inverting input of the op-amp and click left again. ISIS takes this as the other end for the wire and invokes the *Wire Auto Router (WAR)* to choose a route for the wire. Now do the same thing for each end of R2, following the diagram. Try tagging objects and moving them around whilst observing how the WAR re-routes the wires accordingly.

If you do not like a route that the *Wire Auto Router* has chosen, you can edit it manually. To do this, tag the wire (by pointing at it and clicking right) and then try dragging it first at the corners and then in the middle of straight runs. If you want to manually route a wire you can do so by simply clicking left on the first pin, clicking left at each point along the required route where you want a corner, and then finish by clicking left on the second pin. After a while, you will get a feel for when the WAR will cope sensibly and when you will need to take over.

To complete this first section of the drawing, you need to place the two generic and one ground terminals and wire them up. To do this, select the *Terminal* icon; the *Object Selector* changes to a list of the terminal types available. Select the *Ground* terminal, ensure its preview shows it correctly oriented, and place it just under R1. Now select the *Default* terminal from the selector and place two terminals as in the diagram. Finally wire the ground terminal to the bottom of R1 and the two default terminals to the corners of the wires going into the op-amp. ISIS will place the junction dots where required, sensing automatically that three wires are meeting at these points.

## **LABELLING AND MOVING PART REFERENCES**

ISIS has a very powerful feature called *Real Time Annotation* which can be found on the *Tools Menu* and is enabled by default. Full information can be found on page 11 but basically, when enabled, this feature annotates components as you place them on the schematic. If you zoom in on any resistor you have placed you will see that ISIS has labelled it with both the default value (RES) and a unique reference. To edit/input part references and values click left on the *Instant Edit* icon and then click left on the object you wish to edit. Do the resistors first, entering R1, 1k and R2, 1k as appropriate. Now do the op-amp and the two terminals. To move the 'U1' and the '741' labels to correspond with the diagram, press F2 to reduce the snapping grid to 50th (it starts off at 100th) and then tag the op-amp. Now point at the label 'U1' and with the left button depressed, drag it to its correct position under the op-amp. Then do the same with the '741' label.

When you have finished positioning the labels, put the snap back to 100th by pressing F3. Although with the *Real Time Snap* feature ISIS is able to locate pins and wires not on the

current snap grid, working consistently with the same snap grid will keep drawings looking neat and tidy.

### ***BLOCK EDITING FUNCTIONS***

You may have noticed that the section of circuit you have drawn so far is currently located in the middle of the sheet, whereas it should be in the top left hand corner. To move it there, first tag all the objects you have placed by dragging a box round them using the right mouse button: point at a position above and to the left of all the objects; then press and hold down the right button and drag the mouse pointer to a position below and to the right of the objects. The selected area is shown by a cyan *tag-box* and (as the initial right click automatically untags any previously tagged objects) all and only those objects wholly within the *tag-box* will be tagged after the operation.

Now click left on the *Block Move* icon. A box will appear round all the tagged objects, and you can now begin to move this up towards the top left hand corner of the sheet. The sheet border appears in dark blue so you can now re-position the buffer circuit up at the top left of the drawing. Click left to effect the move, or else you can abort it by clicking right. You should also note how, when you moved the pointer off the *Editing Window* to the top or left, ISIS automatically panned the *Editing Window* for you. If you want to pan like this at other times (i.e. when *not* placing or dragging an object), you can use the *Shift Pan* feature.

The group of objects you have moved will remain tagged, so you might as well experiment with the *Copy* and *Delete* icons which similarly operate on the currently tagged objects. The effect of these icons can be cancelled by *immediately* following their use by pressing the 'U' key for Undo.

### ***PRACTICE MAKES PERFECT***

You should be getting the hang of things now, so get some more practice in by drawing the next section of circuitry centred around the op-amp U2:A. You will need to get a capacitor (CAP). A quick method of picking devices whose names you know is to use the *Pick Device/Symbol* command. Press the 'P' key (for *Pick Device/Symbol*) and then type in the name - CAP. Use the various editing techniques that have been covered so far to get everything in the right place. Move the part reference and value fields to the correct positions, but do not annotate the parts yet - we are going to use the *Automatic Annotator* to do this.

When you have done one op-amp filter to your satisfaction, use a *tag-box* and the *Block Copy* icon to make three copies - four filters in all - as there are in the diagram. You may find it useful to use the zoom commands on the *View* menu (or their associated short-cut keys) so as

to be able to see the whole sheet whilst doing this. When you have the four filters in position, wire them together, and place a SW-SPDT device (SW1) on the drawing.

## **ANNOTATING THE DIAGRAM**

ISIS provides you with four possible approaches to annotating (naming) components:

- **Manual Annotation** - This is the method you have already used to label the first op-amp and resistors. Any object can be edited either by selecting the *Instant Edit* icon and then clicking left on it, or by clicking right then left on it in the normal placement mode. Whichever way you do it, a dialogue box then appears which you can use to enter the relevant properties such as Reference, Value and so forth.
- **The *Property Assignment Tool* (PAT)** - This tool can generate fixed or incrementing sequences and assign the resulting text to either all objects, all tagged objects (either on all sheets or the current sheet) or to the objects you subsequently click left on. Using the PAT is faster than manual annotation, though slower than using the *Automatic Annotator*. However, it does leave you in control of which names are allocated to which parts.
- **The *Automatic Annotator*** - Using the *Automatic Annotator* leads to the whole design being annotated in a matter of seconds. The tool is aware of multi-element parts like the 7400 TTL NAND gate package and will allocate gates appropriately. However, the whole process is non-interactive so you get far less control over the names that are allocated than with the other two methods.
- **Real Time Annotation** – This feature, when enabled, will annotate components as you place them on the design, obviating any need for you to place references and values in your design. As with the *Automatic Annotator*, however, it makes the whole process non-interactive and offers no user control over the annotation process. *Real Time Annotation* can be toggled on and off through the *Real Time Annotation* command on the *Tools* Menu or via the CTRL + N shortcut key.

In practice you can use a mix of all four methods, and in any order you choose. The *Automatic Annotator* can be set to leave alone any existing annotation so that it is possible to fix the references of certain parts and then let ISIS annotate the rest by itself. As the *Real Time Annotation* is enabled by default, we shall leave it on and use the other three methods to edit the existing annotation of the design.

### ***The Property Assignment Tool (PAT)***

Let us suppose, for the sake of argument, that you wished to pre-annotate all the resistors using the PAT. Given that you have already manually annotated R1 and R2, you need to generate the sequence R3, R4, R5 etc. To do this, select the *Property Assignment Tool* option

on the *Tools* menu. Enter REF=R# in the *String* field, then move the cursor to the next field (the *Count* field) and key in the value 3. Ensure the *On Click* button is selected and then click left on the *OK* button or press ENTER. The hash-character ('#') in the *String* field text will be replaced with the current *Count* field value each time the PAT assigns a string to an object and then the *Count* field value is incremented.

ISIS automatically selects the *Instant Edit* icon so that you can annotate the required objects by clicking left on them. Point at resistor R3 and click left. The PAT supplies the R3 text and the part is redrawn. Now do the same for the resistor below it, R4 and see how the PAT's *Count* field value increments each time you use it. You can now annotate the rest of the resistor references with some panache. When you are done with this, cancel the PAT, by calling up its dialogue form (use the 'A' keyboard shortcut for speed) and then either clicking on the *CANCEL* button or by pressing ESC.

The PAT can also be used to assign the same *String* to several tagged objects, for example the part values of resistors or capacitors that all have the same value. Consider the capacitors C1 to C4 which all have the value 220p. To assign this value, first ensure that only the capacitors are tagged by first clicking right on a free area of the *Edit Window* to untag all objects, then clicking right on each capacitor. Now invoke the PAT and enter VAL=220p in the *String* field, select the *Local Tagged* button and click OK. That's it - you do not need to cancel the PAT as it is not in its 'On Click' assignment mode.

Try this on your own for the rest of the diagram until you are clear about how the PAT works - although a little tricky at first, it is an extremely powerful tool and can eliminate a great deal of tedious editing. Do not forget that, when used in its *On Click* mode, you need to cancel the tool when finished.

### ***The Automatic Annotator***

ISIS features an automatic annotator which will choose component references for you. It can be made to annotate all the components, or just the ones that haven't yet been annotated - i.e. those with a '?' in their reference.

Since you have already annotated some of the parts, we will run the *Automatic Annotator* in 'Incremental' mode. To do this, invoke the *Global Annotator* command on the *Tools* menu, click on the *Incremental* button, and then click on OK. After a short time, the diagram will be re-drawn showing the new annotation. Since the OPAMP device is not a multi-element part like a true TL074, the annotator annotates them as U2 to U5 which is not what is wanted. To correct this, edit each one in turn and key in the required reference. We will see how to create and use a proper TL074 later on.

Even with the automatic annotator, you still have to set the component values manually, but try this for speed - instead of moving around the drawing to edit each component in turn, simply key 'E' for *Edit Component* (on the *Edit* menu) and key in a component's reference.



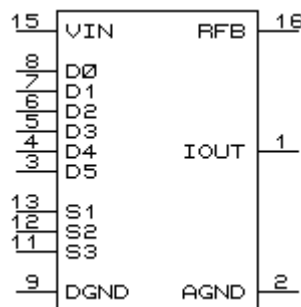
This automatically locates the desired part and brings up its *Edit...* dialogue form. You should also try out the using the *Property Assignment Tool* as described in the above section.

## CREATING NEW DEVICES

The next section of the circuit employs a 7110 digital attenuator, and this provides an opportunity to learn how to make new devices in ISIS.

In ISIS new devices are created directly on the drawing - there is no separate device editor mode, let alone a separate program. The new device is created by placing a collection of 2D graphics and pins, annotating the pins, and then finally tagging them all and invoking the *Make Device* command.

You will find it helpful when creating new devices to sketch out on paper how you want the device to look, and to establish roughly how big it needs to be by considering how many pins there will be down each side and so on. In this case you can use the diagram opposite as a guide. The first thing we need to do is to locate a free area of your design where the new device can be created - click the left mouse button on the lower-right region of the *Overview Window* to position the *Editing Window* on that area of the design.



Begin by drawing the device body of the new device. Select the *Box* icon. You will see that the *Object Selector* on the right displays a list of *Graphics Styles*. A graphics style determines how the graphic we are about to draw will appear in terms of line colour, line thickness, fill style, fill colour, etc. Each style listed is a different set of such attributes and define the way different parts of the schematic appear.

ISIS supports a powerful graphics style system of local and global styles and the ability of local styles to ‘follow’ or ‘track’ global styles that allows you to easily and flexibly customise the appearance of your schematic. See the section *Graphics And Text Styles* on page 41 for a complete explanation of how styles work and how they are used.

As we are drawing the body of a component, select the *COMPONENT* graphics style and then place the mouse pointer over the *Editing Window*, press and hold down the left mouse button and drag out a rectangle. Don't worry about getting the size exactly right - you can always resize the rectangle later. You will see that, as a result of choosing the *COMPONENT* graphics style, the rectangle appears in the same colour, fill, etc. as all the other components on the schematic.

The next thing to do is to place the pins for the new device. To do this, first select the the *Device Pin* icon. The *Object Selector* lists the types of available pins (note that you can also

create your own pin objects in ISIS, though we will not cover that in this tutorial). Select the *Default* pin type from the selector; the *Overview Window* provides a preview of the pin with the pin's name and number represented by the strings NAME and 99 and its base and end indicated by an *Origin* marker and cross respectively - the cross represents the end to which you will eventually connect a wire. Use the *Rotation* and *Mirror* icons to orient the pin preview ready to place the left-hand pins and then click the left mouse button in the *Editing Window* on the left edge of the rectangle where you want each pin's base to appear. Place pins for the VIN, D0..D5, S1..3 and DGND pins. Note that you can use the DOWN key to move the mouse pointer down one grid square and the ENTER key as a substitute for the left mouse button - it is sometimes quicker to use these keys instead of the mouse. Now click left on the *Mirror* icon and then place the three right-hand pins: RFB, IOU and AGND. To finish, place two pins, one on the top edge and one on the bottom edge of the rectangle, adjusting the *Rotation* and *Mirror* icons before placing them in order that they point outwards from the device body; these pins will be the VDD and VBB power pins and will eventually be hidden (this is why they are not shown in the figure).

At this stage, you can reposition the pins or resize the rectangle as required. To move a pin, tag it with the right mouse button and then drag it with the left button; to re-orient it, use the *Rotation* and *Mirror* icons. To adjust the size of the device body rectangle, tag it with the right mouse button, click and hold down the left mouse button on one of the eight 'drag handles' (the small white boxes at the corners and mid-points of the rectangle's edges) and drag the handle to the new position. If you adjust its width, you will also need to draw a *tag-box* (with the right mouse button) around the pins and then use the *Move* icon to re-position them.

So, having arranged the device body rectangle and pins as required, we now need to annotate the pins with names and numbers, and to assign them an electrical type. The electrical type (input, power, pull-up, etc.) is used by the *Electrical Rules Check* to ensure that only pins with the correct electrical type are inter-connected.

We will first assign names, electrical types and visibility. To do this, we have to tag each pin by clicking right on it and then edit it by clicking left on the tagged pin; the pin displays its *Edit Pin* dialogue form.

Edit each pin in turn, as follows:

- Enter the pin's name in the *Name* field. Leave the *Number* field empty as we will assign the pin numbers with the *Property Assignment Tool*.
- Select the appropriate electrical type for the pin - *Output* for the IOU pin, *Power* for the VDD, VBB, AGND and DGND pins, and *Input* for the remainder.,
- Select whether the pin is to be hidden by unchecking its *Draw body* checkbox - the VDD and VBB pins are both standard power pins and can be hidden. The AGND and

DGND pins are non-standard and so need to remain visible in order that they can be wired up as appropriate to the design the device is being used in.

- Select the OK button when finished.

To assign the pin numbers, we will use the *Property Assignment Tool*. To initialise the PAT, select the *Property Assignment Tool* command from the *Tools* Menu, and enter NUM=# in the *String* field and the value 1 in the *Count* field. Select the *On Click* button, and then close the dialogue form with the OK button. Now carefully click on each pin in order of its number (IOUT, AGND, etc.). As you click on each pin, it is assigned a pin number by the PAT. When done, don't forget to cancel the PAT by bringing up its dialogue form and selecting the CANCEL button.

All we do now is actually make the device. To do this, tag all the pins and the body rectangle - the easiest way is to drag out a tag-box with the right mouse button around the whole area being careful not to miss out the two hidden power pins. Finally, select the *Make Device* command from the *Library* menu to display the *Make Device* dialogue form. Key in the name 7110 in the *Name* field and the letter U in the *Prefix* field. Then press the *Next* button until the list of writable device libraries is displayed, select an appropriate library and then click the OK button to save the new device.

## FINISHING TOUCHES

Now that you have defined a 7110 you can place, wire up and annotate the remainder of the diagram - use the *Automatic Annotator* in Incremental mode to annotate the new parts without disturbing the existing annotation.

The labelling and bracket around the six input terminals DA0-DA5 is done with 2D graphics. ISIS provides facilities for placing lines, boxes, circles, arcs and text on your drawings; all of which offered as icons on the *Mode Selector* toolbar.

The bracket is made from three lines - place these by selecting the *Line* icon and then clicking at the start and end of each line. Then place the text FROM PIO-1A as shown by selecting the *Text* icon, setting the *Rotation* icon to 90° and then clicking left at the point where to want the bottom of the 'F' character to appear. You can of course tag and drag 2D graphics objects around to get things just how you want.

Finally, you need to place a sheet border and a header block. To do the former, select the *Box* icon, zoom out till you can see the whole sheet outline (dark blue) and then place a graphics box over it. It is important to realise that the dark blue sheet outline does not appear on hard copy - if you want a bounding box you must place one as a graphics object.

The header block is worthy of more discussion. It is, in fact, no different from other symbols such as you might use for your company logo (see section §*Symbols And The Symbol*

*Library*] for more on symbols). A default header block called HEADER is provided but you can re-define this to suit your own requirements - see *The Header Block* on page 38.

To actually place the header, select the *Symbol* icon and then click left on the **P** button of the *Object Selector* to display the *Symbol Library Selector* dialogue form. Picking symbols from symbol libraries is similar to picking devices from device libraries except that there is no *Prefix* selector. Select the HEADER object from the SYSTEM symbol library and close down the dialogue form. With HEADER now the current symbol, point somewhere towards the bottom left of the drawing, press the left mouse button, and drag the header into position.

Some of the fields in the header block will fill in automatically; others such as the Design Title, Sheet Title, Author and Revision need to be entered using the *Edit Design Properties* and *Edit Sheet Properties* commands on the *Design* menu. Note that the *Sheet Name* field on the *Edit Sheet Properties* dialogue form is different from the *Sheet Title* - the *Sheet Name* is a short label for the sheet that is used in hierarchical design. The *Sheet Title* is a full description of the circuitry on that sheet and it is this that will appear in the header block.

You will need to zoom in on the header to see the full effects of your editing.

## **SAVING, PRINTING AND PLOTTING**

You can save your work at any time by means of the *Save* command on the *File* menu, and now is as good a time as any! The *Save As* option allows you to save it with a different filename from the one you loaded it with.

To print the schematic, first select the correct device to print to using the *Printer Setup* command on the *File* menu. This activates the Windows common dialogue for printer device selection and configuration. The details are thus dependent on your particular version of Windows and your printer driver - consult Windows and printer driver documentation for details. When you have selected the correct printer, close the dialogue form and select the *Print* option on the *File* menu to print your design. Printing can be aborted by pressing ESC, although it may be a short time before everything stops as both ISIS and possibly your printer/plotter have to empty their buffers.

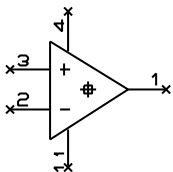
Further details regarding printer and graphics output are given under *Hard Copy Generation* on page 143.

*If you have the demo version, please note that you can only print un-modified sample designs. To try this now, use the Load command on the File menu to load a sample design.*

## **MORE ABOUT CREATING DEVICES**

### ***Making a Multi-Element Device***

We shall now define a proper library part for the TL074 quad op-amp. As there are four separate op-amps to a single TL074 package our tutorial will be showing you how to create *multi-element devices* using the *Visual Packaging Tool*.



The illustration on the left shows the new op-amp device before it is made. The op-amp is made from some 2D graphics, five pins and an origin marker. We will look at two ways to construct the op-amp graphics. The easiest approach uses the pre-defined OPAMP symbol. Proceed as follows:

- Click on the *Symbol* icon and then click the **P** button at the top left of the *Symbols Object Selector*. This will launch the *Symbol Library Selector* dialogue form.
- Double-click on OPAMP in the *System Library* and close the dialogue form using the *Windows Minimise* button on the right of the title bar.
- Position the mouse pointer in an empty area of the *Editing Window* and use the left mouse button to place the op-amp. The op-amp automatically appears in the *COMPONENT* graphics style as this style was used to create the symbol.

Now place the pins around the component body. This is the same process as for creating the 7110 attenuator earlier:

- Select the *Device Pin* icon to obtain a list of available pin types and select the *Default* type.
- Use the *Rotate* and *Mirror* icons to orient the pins before placing them on the design.
- Once all the pins are in the correct positions, edit each pin in turn by tagging it with the right mouse button and then clicking left on it. Use the resulting *Edit Pin* dialogue form to annotate the pin with the correct electrical type and pin name. We have to give the pins names so that we can reference them in the *Packaging Tool* however, we don't want the name to be displayed (as the op-amp pins' uses are implicit from the graphics) so ensure that the *Draw Name* check-box is not checked. Note, there is no need to specify pin numbers as these will entered using the *Packaging Tool*.

The power pins have the names V+ and V- and have the electrical type of *Power*; if you place them just in from the left edge of the op-amp, you will find they just touch the sloping sides of the OPAMP graphic whilst keeping their pin ends (marked by an 'X') on a grid-square. If in a similar situation, they didn't touch, you could 'extend' the base of the pin by placing short lines in 2D Graphic Mode and with the mouse snap off. The input pins have the names +IP and -IP and the electrical type *Input*. The output pin has the name OP and the electrical type *Output*.

The final stage is to place an *Origin* marker. Select the *Marker* icon to display a list of system marker symbols in the *Object Selector*. Select the *Origin* marker in the *System Library* and then place the marker symbol at the centre of the op-amp graphics. The *Origin* marker is displayed as a rectangle with cross-hairs and it indicates to ISIS how the new device should appear around the mouse pointer when the device is dragged or placed in a design.

We have now completed making the device. Tag the constituent parts - the op-amp symbol, pins and the *Origin* marker - by dragging out a *tag-box* around them using the right mouse button, and then invoke the *Make Device* command on the *Library* menu. Then proceed as follows:

- Enter the *Device Name* as TL074 and the Prefix as 'U'
- Click *Next* button display the *Packaging* page and click *Add/Edit* to launch the packaging tool itself.

### ***The Visual Packaging Tool***

The visual packaging tool provides a graphical environment in which to assign one or more PCB footprints to a schematic part. For each packaging, a table of pin numbers to pin names is created such that different packagings can have different pin numbers for the same schematic pin.

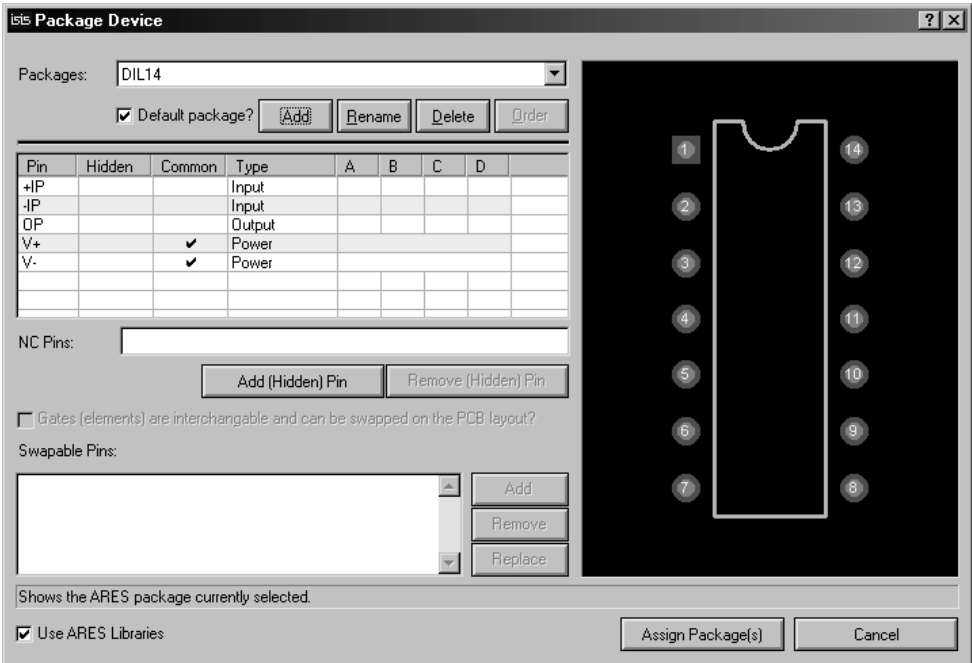
Having launched the packaging tool, the first thing to do is to create a packaging:

- Click the *Add* button. This will launch the ARES library browser.
- Select the PACKAGE library, and double click the DIL14 part.

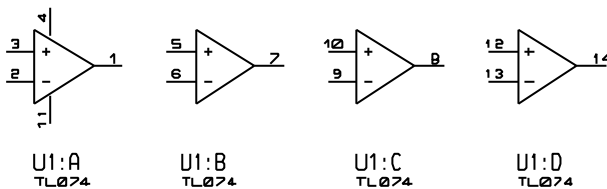
Then, you need to make the following changes to the default settings for the packaging:

- Change the number of elements from 1 to 4. This corresponds with the fact that there are four op-amps within the one physical DIL14 package.
- Mark the V+ and V- pins as common pins. This means that they will have the same pin number on each element, and that you can wire to any or all of the pins on the elements on the schematic. All such wiring will be deemed interconnected.
- Click the *Gates can be Swapped* checkbox. This specifies that the elements are identical and the ARES can perform gateswap operations on this part.

You should then see a display similar to that below:



Now to assign the pin numbers. The actual pinning of the op-amps that we are aiming at is shown below:



Proceed as follows:

- Click left in the 'A' column box for pin +IP.
- Either, click pin '3' on the package display or enter '3' from the keyboard and press TAB. Either way, pin '3' will highlight on the package to show that you have assigned it, and the cursor will move to the -IP row.

- Now, repeat the process for the other pin numbers, until all the pins on the package display are highlighted. This of course, provides a visual cue that you haven't missed any pins.

Finally, click the *Assign Packages* button to return the *Make Device* wizard, and then store the device into your USERDVC library, as you did with the 7110.

There is, as you have probably realized, a great deal of functionality built into the packaging tool. For a detailed discussion and further examples, see *DEVICE LIBRARIES* on page 96.

### ***Making Similar Devices***

Having defined a TL074, you can almost instantly define types TL064 and TL084 as well. Simply place a TL074, tag it, and invoke the *Make Device* command; change the name to TL064 (or whatever) and save it. What could be simpler? If you needed to add something to the basic TL074 - perhaps some more graphics, you could simply add them on top of the placed TL074 before invoking the *Make Device* command. Alternatively, if the TL074 was almost right but needed some slight editing before it was suitable for the new device, you could decompose it back into its constituent parts by tagging it, selecting the *Decompose* command on the *Library* menu, editing and/or adding to them, and then making the new device.

### ***Replacing Components in a design***

You can now replace the four filter op-amps and with proper TL074 parts. To replace a component with one of a similar type, pick the new device, ensure the mouse is over the device you want to replace, click and hold down the left mouse button and drag the new device such that one or more pin-ends overlap. ISIS will then transfer the wires from the old component to the new component whilst keeping all other information about the old component (e.g. its reference, etc.) intact.

## ***SYMBOLS AND THE SYMBOL LIBRARY***

Tag the three lines that form the bracket around the inputs to the 7110. Then invoke *Make Symbol* on the *Library* menu, key in TEST for the symbol name and press ENTER. Now select the *Symbol* icon. You will see that the item TEST has appeared in the *Object Selector*. Pick this and try placing it on the drawing. Common uses for this are things like OPAMP, which is needed for many device types, and logos, emblems etc.

A special use of symbols is for the HEADER block. The default symbol was created out of 2D lines, a box and several special text primitives that are automatically replaced by properties associated with the current design and sheet. For example, a text object with the string:

@DTITLE



---

will automatically appear as the design title entered in the *Edit Design Properties* command's dialogue form. The complete list of keywords is presented in *The Header Block* on page 38.

## **REPORT GENERATION**

Now that the diagram is complete, you can generate Netlist, Bill of Materials (BOM) and Electrical Rules Check (ERC) reports from it. Each report is generated by invoking the appropriate command from the *Tools* menu. The report output is displayed in a pop-up text viewer window from where it can be saved to a file by selecting the 'Save as' button or placed on the clipboard for use in other packages using the 'Copy to clipboard' button; the 'Close' button clears the report display and returns you back to the editor. Note that the last report or simulation log generated is maintained by ISIS - to view a report again, select the *Text Viewer* command on the *System* menu.

The Bill of Materials report should be fairly self explanatory, although you can get a lot more out of the facility - see *Bill Of Materials* on page 135.

The Electrical Rules Check report will contain quite a few errors, since the tutorial circuit is not a complete design - of particular note is that the VBB pin of the 7110 is flagged as undriven, which could easily be forgotten in a real situation.

Further information regarding Report Generation is given in *Report Generation* on page 135 whilst *Netlist Generation* on page 123 details in intricacies of producing and using a netlist. For those of you who have bought ISIS to use with ARES, details of how to link the two packages together are provided in *Isis And Ares* on page 145.

## **A LARGER DESIGN**

In this last section, we shall take a look at a pre-prepared design - EPE.DSN. This is a multi-sheet, hierarchical schematic for micro-processor controlled EPROM programmer/emulator (EPE). As such, it represents a substantial piece of electronics at the lower end of design complexity that you might expect to design with your ISIS system.

The EPE design is structured as three A3 sheets (Processor, Emulator and PSU). Sub-sheets are used to represent an Emulation RAM bank (of which there are 4, giving 32 bit emulation capability) and a Programmable Power Supply (PPSU) of which 6 are required to deal with the range of 27 series EPROM pinouts.

Load the design into ISIS by using the *Load* command on the *File* menu, and then selecting EPE.DSN from the file selector. You will find it in the "Samples\Schematic & PCB Design" folder of your Proteus installation. Alternatively, you can view any of the sample designs by launching the SAMPLES help file from the *Help* menu.

The first sheet is the CPU - take a look round this with the usual pan and zoom facilities. Then, to see more of the design, invoke the *Goto Sheet* command on the *Design* menu. Select the second item from the selector and after some disk activity the Emulator Control sheet will be loaded. Zoom out so that you can see all of it. The 4 big blue boxes are the sub-circuits. The labelling text at the top is the sub-circuit ID (like a part reference) and the text at the bottom is the circuit name.

You can also zoom into the sub-circuits: point at one of the sub-circuits and press the CTRL and 'C' keys together (C is a short-hand for *Child*). ISIS swaps out the Emulator Control sheet and loads an ERAM bank sheet. Take a look round the ERAM bank circuit and in particular, take note of a few of the component numbers. To get out, press the CTRL and 'X' keys together. Zoom into another ERAM bank and compare the component numbers in this one with the first - although both sub-circuit instances share the same circuit (if you modify one instance of the circuit, this will be instantly reflected in the others which simplifies design modification) each has its own set of component annotations; this is *Design Global Annotation* at work.

Now that you know about loading the various sheets and hierarchy roaming you may as well explore the rest of the EPE design. It is a good mix of analogue, digital and microprocessor circuits which shows how ISIS is well suited to all types of schematic.

# GENERAL CONCEPTS

## SCREEN LAYOUT

### The Menu Bar

File View Edit Library Tools Design Graph Source Debug Template System Help

The *Menu Bar* runs across the top row of the screen and its main purpose (the selection of commands from the menus) is the same as with any other Windows application. In addition, the title bar area above the menu names is used to display certain prompt messages which indicate when the program has entered a particular editing or processing mode.

### The Toolbars

As with other modern Windows applications, ISIS provides access to a number of its commands and modes through the use of toolbars. The toolbars can be dragged to any of the four edges of the ISIS application window.

#### Command Toolbars

The tools located along the top of the screen (by default) provide alternative access to the menu commands, as follows:

*File/Print commands*



*Display Commands*



*Editing Commands*



*Design Tools*



If you are working on a relatively small monitor, you can hide any or all of the command toolbars using the *Toolbars* command on the *View* menu.

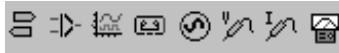
#### Mode Selector Toolbar

The toolbar located down the left hand edge of the screen select the editor mode, i.e. what happens when you click the mouse on the *Editing Window*.

*Main Modes*



Gadgets



2D Graphics



Note that the mode toolbar cannot be hidden, as its functions are not duplicated on the menus.

### **Orientation Toolbar**

The orientation toolbar displays and controls the rotation and reflection for objects placed onto the layout.

Rotation



Reflection



The edit box allows you type a rotation angle in directly; but note that in ISIS, only orthogonal angles may be entered.

When an existing object is tagged, the rotation and reflections icons highlight in red to show that they will modify the orientation of an object on the layout. When the icons are not highlighted, they serve to determine the orientation for new objects.

### **The Editing Window**

The *Editing Window* displays the part of the schematic that you are currently editing. The contents of the *Editing Window* may be redrawn using the *Redraw* command on the *View* menu. This also redraws the *Overview Window*. You can use this feature after any other command that has left the display somewhat untidy.

### **Panning**

You can reposition the *Editing Window* over different parts of the design in several ways:

- By clicking left at a point on the *Overview Window* - this re-centres the *Editing Window* about the marked point.
- By moving the mouse over the *Editing Window*, holding down the SHIFT key, and 'bumping' the pointer against one of its edges. This pans the display in the appropriate direction. We refer to this feature as *Shift-Pan*.
- By pointing in the *Editing Window* and pressing the Zoom key (see below). This re-centres the display about the cursor position.

- By using the *Pan* icon on the toolbar.

### **Zoom In / Zoom Out**

You can magnify or reduce the display of the board using the *Zoom In* and *Zoom Out* commands which are also invoked by the F6 and F7 shortcut keys. Pressing F8 will display a view of the entire board. You can also use the corresponding icons on the toolbar.

If the keyboard is used to invoke the command, then the *Editing Window* will be redrawn to display a region centred around where the mouse cursor was pointing before. This also provides a way to effect a pan by pressing the zoom key for the current level and simultaneously pointing with the mouse at where you want centre of the new display to be.

### **Variable Zoom**

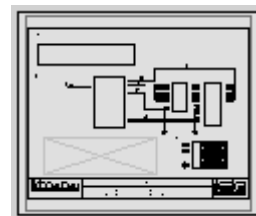
An arbitrary degree of magnification can be achieved using the Shift-Zoom feature. A given area of the board can be selected to fill the *Editing Window* by holding down the SHIFT key, pressing the left mouse button and dragging out a box around the desired area. The area can be marked on either the *Editing Window* or the *Overview Window*.

You can also zoom to an area by clicking the *Zoom Area* icon on toolbar.

### **The Overview Window**

This window normally shows a simplified representation of the whole drawing, and has a half-inch grid on it. The cyan box marks the outline of the sheet border, whilst the green box indicates the area of the design currently visible in the *Editing Window*.

Clicking left at a point on the grid re-centres the *Editing Window* around this point, and redraws the *Editing Window*.



At other times, *Overview Window* is used to display a preview of an object that is selected for placement. This *Place Preview* feature is activated in the following circumstances for any object which may be oriented:

- When an object is selected from the object selector.
- When the rotate or mirror icons are adjusted.
- When an object type icon is selected for an object whose orientation can be set (e.g. the *Component* icon, *Device Pin* icon, etc.).

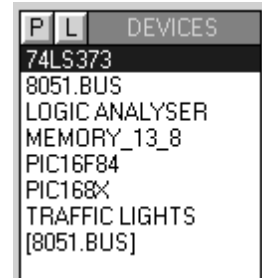
The place preview display is cleared automatically as soon as you proceed the place the object, or when you perform any operation other than those listed above.

The width and height of the *Overview Window* can be adjusted by dragging its borders. If you drag the vertical border right over to the other side of the application Window, ARES will re-organize the display so that the *Overview Window* and *Object Selector* are located at the right hand side.

### ***The Object Selector***

The Object Selector is used for picking components, terminals, generators, graph types and so on from those that are available. It always carries a label indicating what it is listing and this serves as a prompt additional to the state of the Icon Panel as to which mode is current.

The width and position of the *Object Selector* can be adjusted in conjunction with the width and height of the *Overview Window*, as described above.



The buttons at the top left of the object selector activate various functions such as library browsing and library management. The exact functions available depend on the current editor mode.

### ***Co-ordinate Display***

The current co-ordinates of the mouse pointer are displayed down at the bottom right of the screen by default. The read-out can be in imperial or metric units and a false origin may be set. Further details are given under CO-ORDINATE SYSTEM on page 26.

The *X-Cursor* command will display a small or large cross, in addition to the mouse arrow, at the exact location of the current co-ordinates. This is particularly helpful when used in conjunction with the Real Time Snap feature, since it gives you an immediate indication of what ARES thinks you are pointing at.

## **CO-ORDINATE SYSTEM**

All co-ordinates in ISIS are actually held in 10nm units, purely to be consistent with ARES. However, the coordinate read-out is restricted to 1 thou units. The origin is held to be in the centre of the work area and so both positive and negative values are used. The co-ordinates of the pointer are displayed at the bottom right of the screen.

We refer to a 1 thou increment as a unit.

## ***False Origin***

Although the *Origin* command appears on the *View* menu, it should only be used via its keyboard short cut (key 'O'). Its function is to zero the co-ordinate display at the current mouse position, a feature that is most useful when laying out a complex pattern of pads given a set of dimensions on a drawing of the component.

When a false origin is set, the co-ordinate display changes colour from black to magenta as a reminder.

Cancelling a false origin is done by invoking the *Origin* command a second time.

## ***The Dot Grid***

A grid of dots is displayed in the *Editing Window* - this can be toggled on and off using the *Grid* command on the *View* menu. The dot spacing reflects the current snap setting., unless this would result in a ridiculous number of dots, in which the spacing is increased.

## ***Snapping to a Grid***

You will notice that when the pointer is over the *Editing Window*, the increments of the co-ordinate display are in fixed steps - initially 100th. This is called *snapping* and enables you to position components and other objects on a neat grid. The degree of snap may be selected with the *Snap* commands on the *View* menu, or directly with keys F4, F3, F2 and CTRL+F1.

If you wish to see exactly where the snapped position is, you can use the *X-Cursor* command on the *View* menu that will display either a small or large cross at this location.

## ***Real Time Snap***

Furthermore, when the pointer is positioned near to pin ends or wires, the cursor location will also be snapped onto these objects. This function is called *Real Time Snap* and allows you to connect to or from pins and wires that are not on the currently selected snap grid. You can toggle this function using the *Real Time Snap* command on the *Tools* menu, or by keying CTRL+'S'.

With a very large drawing on a slow computer, real time snap may cause some lag between the cursor and the pointer. You may find it best to disable the feature in these circumstances.

# FILING COMMANDS

ISIS makes use of the following file types:

Design Files	(.DSN)
Backup Files	(.DBK)
Section Files	(.SEC)
Module Files	(.MOD)
Library Files	(.LIB)
Netlist Files	(.SDF)


Design files contain all the information about one circuit, and have the file extension 'DSN'. Previous versions of ISIS have used 'ISS', 'IDS' and 'IWS'; these can be converted automatically provided that you have the file converters IDSCVT40.DLL and/or IWSCVT40.DLL installed. Backup copies of design files made when saving over an existing file are given the extension "DBK".

A section of a drawing can be exported to a section file and subsequently read into another drawing. Section files have the extension 'SEC'. and are read and written by the *Import* and *Export* commands on the *File* menu.

Module files have the extension 'MOD' and are used in conjunction with the other features for hierarchical design. See *Hierarchical Designs* on page 117 for further information.

Symbol and device libraries have the extension 'LIB'.

Netlist files produce when exporting connectivity to ProSPICE and ARES have the extension 'SDF'; other extensions are used when producing netlist files in 3<sup>rd</sup> party formats.

 The Proteus VSM simulation system uses other file types as well. See the Proteus VSM manual for further details.

## Starting a New Design

The *New Design* command will clear out all existing design data and present a single, empty A4 sheet. The design name is set to UNTITLED.DSN and this name will be used by the *Save Design* command and also as the default filestem in other file selectors.

Should you wish to start a new design and give it a name at the same time, you can use the *Load Design* command and enter the new filename in the file selector.

## Loading a Design

A design may be loaded in three ways:



- From the DOS prompt as in:  
    ISIS <my\_design>
- By using the *Load Design* command once ISIS is running.
- By double clicking the file in *Windows Explorer*.

## ***Saving the Design***

You can save your design when quitting ISIS via the *Exit* command, or at any other time, using the *Save Design* command. In both cases it is saved to the same file from which it was loaded. The old version will be prefixed with the text 'Backup of'.

The *Save As* command allows you to save the design to a different file.

## ***Import / Export Section***

The *Export* command on the *File* menu creates a section file out of all currently tagged objects. This file can then be read into another sheet using the *Import* command. After you have chosen the section file, operation is identical to the *Block Copy* function.

These commands have nothing to do with graphics export to DTP packages. The *Export Graphics* commands handle this functionality.

## ***Quitting ISIS***

When you wish to end an ISIS session, you should use the *Exit* command on the file menu, key 'Q'. If you have modified the design, you will be prompted as to whether you wish to save it.

# **GENERAL EDITING FACILITIES**

## ***Object Placement***

ISIS supports many types of object, and full details of the purpose and behaviour of each type is given in the next chapter. However, the basic steps for placing an object are the same for all types.

### **To place an object:**

1. Select the appropriate icon from the *Mode Selector* toolbar for the category of object that you want to place.

2. If the object type is Component, Terminal, Pin, Graph, Symbol or Marker, select the name of the object that you want from the selector. For Components, Terminals, Pins and Symbols, this may first involve picking it from the libraries.
3. If the object is orientable, it will have appeared in the *Overview Window*. You should now adjust its orientation to that which you require by clicking on the *Rotation* and *Mirror* icons.
4. Finally, point on the *Editing Window* and click left to place or drag the object. The exact procedures vary for each object type but you will find it all fairly intuitive and similar to other graphics software. If you really want to read about the details, they are given in *Object Specifics* on page 63.

### ***Tagging an Object***

Any object may be tagged by pointing at it and clicking right. This action highlights the object and selects it for further editing operations.

- Any wires connected to an object that is tagged are also tagged.
- A group of tagged objects may be assembled either by clicking right on each object in turn, or by dragging a box around the objects using the right button. Only objects wholly enclosed in the box will be tagged.
- All the objects may be untagged by pointing at no object and clicking right.

### ***Deleting an Object***

You can delete any tagged object by pointing at it and clicking right. All wires connected to the object will also be deleted, except in the case of a dot connected to exactly 2 wires, in which case the wires will be joined.

### ***Dragging an Object***

You can drag (i.e. re-position) any tagged object by pointing at it and then dragging with the left button depressed. This applies not only to whole objects, such as components, but also individually to their labels.

- If the *Wire Auto Router* is enabled and there are wires connected to it, then these will be re-routed or 'fixed up'. This can take some time (10 seconds or so) if the object has a lot of connected wires; the pointer becomes an hour glass while this is happening.
- If you drag an object by mistake, and all the wiring goes horribly wrong, you can use the Undo command, key U to restore things to their original state.

## ***Dragging an Object Label***

Many object types have one or more *labels* attached to them. For example, each component has a reference label and a value label. It is very easy to move these labels in order to improve the appearance of your schematics.

### **To move a label:**


1. Tag the object by pointing at it (or the label) and clicking right.
2. Point at the label, press the left mouse button.
3. Drag the label to the required position. If you need to position it very finely, you can change the snap resolution (using the keys F4, F3, F2, CTRL+F1) even whilst dragging.
4. Release the mouse button to finish.

## ***Resizing an Object***

Sub-circuits, graphs, lines, boxes and circles may be resized. When you tag these objects, little white squares called *handles* will appear and the object can be re-sized by dragging the handles.

### **To resize an object:**

1. Tag the object by pointing at it and clicking right.
2. If the object can be resized, a set of little square handles will appear on it.
3. Resize the object by pointing at a handle, pressing the left mouse button, and dragging it to a new position. The handles disappear whilst you are dragging so that they do not obscure your view of the object itself.

 See the section *Resizing 2d Graphics* on page 87 for information about advanced use of handles when resizing 2D graphics path objects.

## ***Reorienting an Object***

Many of the object types may be oriented - that is rotated through 0°, 90°, 270° and 360° and reflected in x and/or y. If such an object is tagged, the *Rotation* and *Mirror* icons will change colour from blue to red, and will then affect the tagged object.

### **To reorient an object:**

1. Tag the object by pointing at it and clicking right.
2. Click left on the *Rotation* icons to rotate it anti-clockwise, right to rotate it clockwise.

3. Click left on the *Mirror* icon to toggle its reflection in x, and right to toggle its reflection in y.

It is worth noting that if the *Rotation* and *Mirror* icons are red, operating them will affect an object somewhere on the diagram, even if you cannot currently see it. This becomes important if, in fact, you wish to manipulate a new object which you are about to place. If the icons are red, first untag the existing object by pointing at an empty area of the design in the *Editing Window* and clicking right. The icons will then revert to blue, indicating that it is 'safe' to adjust them.

### ***Editing an Object***

Many of the objects have graphical and/or textual properties that may be edited through a dialogue form, and because this is a very common operation we have provided a variety of ways to achieve it.

#### **To edit a single object using the mouse:**

1. Tag the object by pointing at it and clicking right.
2. Click left on it, as if to drag, but release the mouse button immediately, without moving the mouse.

#### **To edit a succession of objects using the mouse:**

1. Select the *Instant Edit* icon.
2. Point at each object in succession and click left.

#### **To edit an object and access special edit modes:**


1. Point at the object.
2. Press CTRL+'E'.

For text scripts, this will invoke the external text editor. Also, if the mouse is not over any object, this command will edit the current graph, if any.

#### **To edit a component by name:**

1. Key 'E'.
2. Type in the reference name (part ID) of a component.

This will locate and bring up the dialogue form for any component in the design, not just those on the current sheet. After the edit, the screen is re-drawn with the component in the centre. You can thus use this command to locate a component, even if you do not actually want to edit it.

 Details pertaining to the operation of the dialogue forms associated with each object type are given in *Object Specifics* on page 63.

## ***Editing an Object Label***

Component, terminal, wire and bus labels can all be edited in much the same way as objects:

### **To edit a single object label using the mouse:**

1. Tag the object by pointing at it and clicking right.
2. Click left on the label, as if to drag, but release the mouse button immediately, without moving the mouse.

### **To edit a succession of object labels using the mouse:**

1. Select the *Instant Edit* icon.
2. Point at each label in succession and click left.

Either way, a dialogue form with *Label* and *Style* tabs is displayed. The editing of local text styles is fully covered in the tutorial on graphics and text styles- see *Editing Local Styles* on page 44.

## ***Copying all Tagged Objects***

### **To copy a section of circuitry:**

1. Tag the required objects either individually, or by dragging out a tag-box as described in *Tagging An Object* on page 30.
2. Click left on the *Block Copy* icon.
3. Drag the copy outline to the require position and click left to place a copy.
4. Repeat step [3] as required to place multiple copies.
5. Click right to finish.


When components are copied, their references are automatically reset to the un-annotated state so as to ready them for automatic annotation, and prevent the occurrence of multiple instances of the same part IDs.

### ***Moving all Tagged Objects***

#### **To move a set of objects:**

1. Tag the required objects either individually, or by dragging out a tag-box as described in *Tagging An Object* on page 30.
2. Click left on the *Block Move* icon.
3. Drag the outline to the required position and click left to place it.

The behaviour of wires during block move is somewhat subtle. Essentially, ISIS will move all wires or parts of wires enclosed in the tag-box without re-routing them, and then, where wires cross the boundaries of the tag-box, it will reroute from the last point inside the tag-box to the first point outside it. It follows that you can control whether a section of wiring is preserved or re-routed according to whether you include it in the tag-box or not.

 You can also use block move to move just sections of wiring, without moving any objects at all. Further discussion of this is given in *Dragging Wires* on page 36.

### ***Deleting all Tagged Objects***

#### **To delete a group of objects:**

1. Tag the required objects either individually, or by dragging out a tag-box as described in *Tagging An Object* on page 30.
2. Click left on the *Delete* icon.

If you delete something by mistake, you can recover it using the *Undo* command.

## ***WIRING UP***

### ***Wire Placement***

You may have noticed that there is no *Wire* icon. This is because ISIS is intelligent enough to detect automatically when you want to place a wire. This avoids the tedium of having to select a wire-placement mode.

#### **To connect a wire between two objects:**

1. Click left on the *connection point* of the first object.
2. If you want ISIS to auto-route the wire, just click left on a second connection point. On the other hand, if you wish to determine the wire's route yourself, you can click left on one or intermediate points which will become corners in the wire's route.

A connection point can connect to precisely one wire. For components and terminals there is a connection point at the end of each pin. A dot has four connection points at its centre so that four wires can be joined at a junction dot.

Since it is common to wish to connect to existing wires, ISIS also treats wires as continuous connection points. Furthermore, as such a junction invariably means that 3 wires are meeting at a point it also places a dot for you. This completely avoids ambiguities that could otherwise arise from missing dots.

You can abort the routing of a wire by pressing ESC at any stage of the process.

### **The Wire Auto-Router**

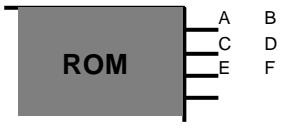
The *Wire Auto-Router* (WAR) saves you the chore of having to mark out the exact route of each wire. The feature is enabled by default, but can be overridden in two ways.

If you simply click left at two connection points, the WAR will attempt to chose a sensible path for the wire. If, however, you click on one connection point, and then click at one or more positions which are not connection points, ISIS will assume you are manually routing the wire and will let you click at each corner of the wire's route. The route is completed by clicking left on a second connection point.

The WAR can be completely disabled using the *Wire Auto-Router* command on the *Tools* menu. This is useful if you want to route a diagonal wire directly between two connection points.

### **Wire Repeat**

Suppose you have to connect the data bus of an 8 bit ROM to the main data bus on the circuit diagram and that you have placed the ROM, bus and bus entries as shown overleaf.



You would first click left at A, then B to place a horizontal wire between them. By clicking twice at C, you will invoke the Wire Repeat function which will then place a wire between C and D. Clicking left twice on E will join E and F and so forth.

Wire Repeat copies exactly the way the previous wire was routed. If the previous wire was automatically routed then the repeated wire will also be automatically routed. On the other hand, if the previous wire was manually routed then its exact route will be offset and used for the new wire.

### Dragging Wires

Although wires follow the general scheme of tag then drag, there are various special techniques that you can apply to them. In particular:

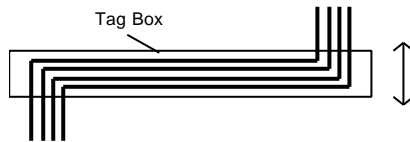
- If you point at a corner and drag then the corner simply follows the pointer.
- If you point in the middle of a horizontal or vertical wire segment, the segment will be dragged vertically or horizontally respectively and the adjacent segments will be stretched to maintain connectivity.
- If you point in the middle of a diagonal segment, or at either end of the wire, then a corner will be created and then dragged. Note that in order for the latter to work, the object to which the wire connects must not be tagged, as otherwise, ISIS will think you are trying to drag the object.

It is also possible to move a wire segment, or a group of wire segments using the block move command.

#### To move a wire segment or a group of segments:

1. Drag out a tag-box around the wire segment(s) you wish to move. It is quite acceptable for this 'box' to be a line, lying along a single segment, if this is convenient.
2. Click left on the *Move* icon.
3. Move the tag-box in the direction orthogonal to the wire segments, as shown in the diagram, opposite.
4. Click left to finish.

If it all goes wrong, you can use the *Undo* command to recover the situation.



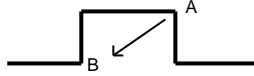
A further technique provides a quick way of eliminating unwanted kinks in wires, perhaps where they have been routed around an object which has since been moved.

#### To remove a kink from a wire:

1. Tag the wire you wish to manipulate.
2. Point at one corner of the kink and press the left mouse button.
3. Drag the corner such that the kink is doubled over itself, as in the diagram, below.



4. Release the left mouse button. ISIS will remove the kink from the wire.



## THE AUTOMATIC ANNOTATOR

ISIS can automatically chose component references for all or some of the components in a design - this process is called *Auto-Annotation*.

The process is initiated using the *Global Annotator* command on the *Tools* menu.

Note that the Global Annotator cannot annotate heterogeneous multi-element parts. This is because with several un-annotated relay coils and contacts, for example, there is no way for it to know what goes with what.

### Value Annotation

This facility is intended for use where an analysis program computes some values for a standard circuit, and you want to import them. An example block would be:

```
VALUES
R1 , 10k
C1 , 100n
END
```

The value of R1 would be set to 10k and C1 to 100n.

## MISCELLANEOUS

### The Sheet Border

When a new sheet is created either as the first sheet of a new design, or by using the *New Sheet* command, it starts off at the size currently selected on the *Set Sheet Sizes* dialogue on the *System* menu. The extent of the sheet is shown by a dark blue sheet outline but this in itself will not actually appear on hard copy.

If a sheet border is required on the final output, you must place a graphics box (or whatever) over the sheet outline.

The *Set Sheet Sizes* command is worthy of further discussion since it serves two distinct purposes:

- To change the paper size for the current sheet, you simply invoke the command and click on the button for the required sheet size.
- To re-define the dimensions of a paper size, highlight the appropriate data entry field(s) and key in the new dimension(s). If you change the dimensions of the current sheet size, this will affect the current sheet immediately but not any other sheets in the design. To affect these, you must invoke *Set Sheet Sizes* and click OK for each sheet in turn.

The *Set Sheet Sizes* dialogue form allows you to define the sizes of five standard sheet sizes (A4 through to A0) as well as a non-standard size and to select which one is the size if the current sheet.

For each sheet size (*A4-A0* and *user*) there are two edit fields. The left field defines the width (x-dimension) and the right side defines the height (y-dimension). Whenever ISIS loads a sheet that does not match any of the standard (A4-A0) sheet sizes it places the dimensions of the sheet into the *User* field and sets that to be the default sheet size for the loaded sheet.

The default values for the standard sheet sizes should work for most printers but please note that all printers have 'margins' in which they cannot print – if you attempt to print something that impinges on the margins it will most likely not be drawn. The size of the margin where a printer cannot print varies between printers and the best way to determine it is to draw a sample design with a box that surrounds your chosen sheet size and then printing it. If one or more edges of the box do not appear on the printed output then your sheet size extends into the printers margins and you should reduce it.

There is one more point to note about sheet sizes: Every sheet in a design carries its actual dimensions with it in the design file. If a design is loaded into a copy of ISIS with a different sheet size configuration, this will not have any effect unless and until the *Set Sheet Sizes* command is used.

### ***The Header Block***

It is common practice to have on each sheet of a drawing a header block which shows details such as the design and sheet titles, the document, revision and page numbers, and the design's author.

In order that you have full control over how this information is presented, the header block is defined as a symbol library entry called HEADER. This was made in the usual way by placing graphics objects, tagging them and invoking the *Make Symbol* command. The significant thing is where, for example, the current design title is required, a text object with the string **@DTITLE** was placed and this is automatically replaced with the actual design title at display/print time.

The complete list of such keywords is shown overleaf.

---

<b>@DTITLE</b>	The design title taken from the <i>Edit Design Properties</i> command form.
<b>@STITLE</b>	The sheet title taken from the <i>Edit Sheet Properties</i> command form. Do not confuse this with the Sheet Name.
<b>@DOCNO</b>	The design document number taken from the <i>Edit Design Properties</i> command form.
<b>@REV</b>	The design revision number taken from the <i>Edit Design Properties</i> command form.
<b>@AUTHOR</b>	The design author taken from the <i>Edit Design Properties</i> command form.
<b>@CDATE</b>	The design creation date - generated automatically and in a fixed format.
<b>@MDATE</b>	The design modification date - generated automatically and in a fixed format.
<b>@WS_CDATE</b>	The design creation date - generated automatically and formatted according to the Windows 'short date format' (see below).
<b>@WL_CDATE</b>	The design creation date - generated automatically and formatted according to the Windows 'long date format' (see below).
<b>@WS_MDATE</b>	The design modification date - generated automatically and formatted according to the Windows 'short date format' (see below).
<b>@WL_MDATE</b>	The design modification date - generated automatically and formatted according to the Windows 'long date format' (see below).
<b>@CTIME</b>	The design creation time - generated automatically and formatted according to the Windows 'time format' (see below).
<b>@MTIME</b>	The design modification time - generated automatically and formatted according to the Windows 'time format' (see below).
<b>@PAGENUM</b>	The current sheet page number within the design.
<b>@PAGECOUNT</b>	The total number of sheets in the design.
<b>@PAGE</b>	The sheet page number within the design expressed as X/Y where X is the sheet page number and Y is the total number of sheets in the design.
<b>@FILENAME</b>	The current design's filename.

**@PATHNAME** The full path and filename of the current design file.

The Windows long and short date formats and Windows time formats are set using the *Regional Settings* or *International* applets in *Windows Control Panel*.

Note that the above keywords must appear at the start of a 2D text string and that the string should not contain additional text. For example, do not place strings of the form:

```
AUTHORED BY @AUTHOR ON @WS_MDATE
```

as these will not work! To achieve the above you need to place four strings along side each other (AUTHORED BY, @AUTHOR, ON and @WS\_MDATE).

By use of the 2D graphics objects and such special text objects it is possible to define any kind of header block you want. In particular, you can incorporate your company logo into the header. Once defined, the header can be placed onto each sheet of the drawing like any other graphics symbol.

### ***Send to Back / Bring to Front***

Sometimes several objects (especially Graphics) overlap and it becomes difficult to point at the one you want. By default, ISIS takes the most recently placed object, but you can re-arrange the order using the *Send to Back* and *Bring to Front* commands. Each command operates on the currently tagged objects.

You can also use these commands to order the draw order when creating new devices. For example, if you are creating a new op-amp symbol, you may need to send the op-amp body (the triangle graphic) 'to the back' in order to ensure it's fill pattern doesn't obscure the '+' and '-' input symbols, which of course, need to be 'in front'.

# GRAPHICS AND TEXT STYLES

## INTRODUCTION

ISIS implements a sophisticated scheme to allow you to customise the appearance of a schematic in terms of line styles, colour fills, text fonts, text effects, etc. The system is extremely powerful and allows you to control some or all aspects of the schematic appearance globally whilst allowing certain objects to carry their own local appearance attributes.

All graphical objects in ISIS (component bodies, wires, junction dots, etc.) are drawn according to a *graphics style*. A graphics style is a complete description of how to draw and fill a graphical shape (such as a line, box, circle or whatever) and includes attributes for line styles (solid, dotted, dashed etc.), thickness, colour, fill style, fill foreground and background colour, etc. Similarly, all labels and script blocks in ISIS (terminal labels, pin names, etc.) are drawn according to a *text style*. A text style is a complete description of how to draw some text and includes attributes for the font face (e.g. Arial, Times Roman, etc.), character height, width, colour, etc.

In ISIS, most objects, such as 2D graphics, wires, terminal labels, etc., each have their own *local* style in order that they can be customised on an object-by-object basis - that is, for example, that one wire can have a different appearance to another wire. The term *local* is used as the style settings are local to the object. Other objects such as pin names, sub-circuit bodies, etc. are always drawn in one of the predefined styles and therefore these objects can only be customised on an all-or-nothing basis - that is, for example, sub-circuits can have whatever appearance you want, but all sub-circuits must appear the same.

Most objects that have their own style have their *local* style initialised from the most appropriate of the *global* styles when they are placed. For example, when you place a terminal, its label's text style is automatically initialised with the *TERMINAL LABEL* style and when you place a wire its graphics style is automatically initialised with the *WIRE* style. 2D Graphics Objects are slightly different in that, for these objects, the right-hand *Object Selector* displays a list of available graphics styles and the newly-placed graphics object is initialised with whatever style is currently selected.

Now comes the clever part. Each *local* style keeps a track of the *global* style used to initialise it. In addition, each such local style also has a set of *Follow Global* options with one such option for each attribute in the style. The *Follow Global* option, when selected, indicates that the associated style attribute should always assume the value of the *global* style and, when not selected, that the value of the local style attribute be used. By default, for all newly-placed

objects, all the *Follow Global?* options are selected such that the appearance of the newly-placed object fully and completely follows the global style it is based on.

The benefits of having local and global styles and the ability of local styles to follow some or all the attributes of a global style are that:

- it allows you to edit the overall appearance of a design via a single edit of the global style - you do not need to edit all objects concerned individually.
- it allows you to define library symbols which will automatically 'blend in' with the appearance of the drawing onto which they are placed.
- it allows you to 'fix' some or all parts of the appearance of a component or other object.

For example, suppose you create a new component and put it in a library. Providing you draw the component's graphics in the *COMPONENT* style, then when the component is subsequently loaded from the library in to a new drawing, it will automatically take on and follow the *COMPONENT* style for that drawing.

## TUTORIAL

As a starting point for the tutorial on graphics and text styles, load the sample design *STYLETUT.DSN* from the "*Samples\Tutorials*" directory of your Proteus installation. As you can see, the basic colour scheme in this design is blue outlines with yellow fills for components and red for terminals, etc. and if you zoom in about some text, you will see that it is all in the Labcenter Electronics 'vector' font.

### Editing Global Styles

We will begin by looking at how to edit the global styles. These styles are, as their name implies, global to the design and editing these styles allows you to make global changes to the appearance of a schematic.

Let us start by changing the component colour scheme a little. All the components in the design were picked from the standard libraries and as such, the component graphics are all drawn in the *COMPONENT* style. We can change this style using the *Set Graphics Styles* command on the *Template* menu. Select this command and you will see the *Edit Global Graphics Styles* dialogue form displayed. This form gives access to all the graphics styles defined for the current design - drop the *Style* selector down to see the full list. As the buttons below the *Style* list indicate, it is possible to create, edit and delete new styles in addition to the editing the 'predefined' styles.

Make sure the *COMPONENT* style is selected in the *Styles* list. You will see in the *Line Attributes* and *Fill Attributes* sections the attributes for the style. Under *Line Attributes*

---

select a red colour and then under *Fill Attributes* select a *Fill Style* of 'none'. After both changes you will see that the sample at the right of the dialogue form updates to show a new preview of the style.

Now select the *TERMINAL* style in the *Styles* list. The changes you made to the *COMPONENT* style are automatically saved and the dialogue changes to display the settings of the *TERMINAL* style. This time, under *Line Attributes* select a blue colour for lines and then under *Fill Attributes* select a *Fill Style* of 'solid' which enables the *Fg. Colour* control where you can select a yellow fill colour.

Close the form using the *Close* button and the schematic is redrawn to show the changes. The new colour scheme would probably work better on a white background and as you will probably print your designs on white paper, let's now select white as our 'paper' colour. From the *Template* menu select the *Set Design Defaults* command. The *Edit Design Defaults* dialogue form allows you to edit most of the standard colours used by ISIS including the background or 'paper' colour. Change this to a light grey or white and close the form.

So we've seen how to change graphics styles. How about text styles? Changing these is pretty much the same as changing text styles except that style attributes you get to change are different.

However, before we start changing the text styles of individual elements, let's change the text font used in the drawing 'en mass'. As you've already noted, the drawing is currently displaying all its text in the Labcenter Electronics 'vector' font. To change the default design font, again select the *Set Design Defaults* command from the *Template* menu to display the *Edit Design Defaults* dialogue form. Under *Font Face For 'Default' Font* you will see *Vector Font* selected - change this to the standard Windows *Times New Roman* font face and close the form. The design is redrawn and all text can now be seen to be in *Times New Roman*.

Wherever ISIS offers a list of TrueType™ fonts (e.g. as part of editing a text style) there are always two additional font faces listed at the top of the list. These are *Default Font* and *Vector Font*. The *Default Font* option is a place holder for whatever font is selected on the *Edit Design Defaults* dialogue form whilst the *Vector Font* option selects the internal Labcenter Electronics *Vector Font*. In the case of STYLETUT.DSN all the text and labels in the design has been placed and/or edited to use the *Default Font* face which means that it all displays in the font set on the *Edit Design Defaults* form - select a new font face there and all the text in the design is changed immediately!

The main use of the *Vector Font* is where you need to both produce hard copy output on a pen plotter. Windows does not properly support the use of TrueType fonts on plotters, and the *Labcenter Plotter Driver* will output all text in the *Vector Font* whatever it is defined as on the drawing. The *Vector Font* has the additional advantage that it is guaranteed to come

out *exactly* the same size on all hard copy devices as it does on the screen. This is not always the case with TrueType fonts.

On the other hand, if you only need to use bitmap devices (i.e. printers) these issues do not arise and you can use all your TrueType™ fonts as you see fit. It has to be said that schematics which use more than two font faces are likely to look very odd!

Having changed our designs 'default font' to be *Times New Roman* lets make some object specific text style changes. From the *Template* menu, select the *Set Text Styles* command to display the *Edit Global Text Styles* dialogue form. Notice how similar this dialogue form is to the *Edit Global Graphics Styles* dialogue form you saw earlier - as we said, editing text styles is very similar to editing graphics styles.

By default, *COMPONENT ID* should be the *Style* selected for editing and you will see that the *Font Face* is set to *Default Font* which is, as we explained, the placeholder for the font selected on the *Edit Design Defaults* dialogue form, currently *Times New Roman*. Because of this, some fields on the dialogue form that aren't appropriate to a TrueType™ font, such as character width, are greyed out. Change the *Font Face* to be *Courier New* and select *Bold?* under the *Effects* options. Now select the *PIN NUMBER* style in the *Style* list (the changes to the *COMPONENT* style are automatically saved) and then uncheck the *Visible?* checkbox under the *Effects* options. Use the *Close* button to close the dialogue form.

You will see that on the redrawn schematic the component IDs are now in bold *Courier New* and that the pin numbers of the OP-AMP are no longer visible. The latter effect is often useful when producing drawings for illustration or as block diagrams.

### ***Editing Local Styles***

So far we have only changed 'global' styles. Changing these has had an effect right across the schematic. We will now look at making changes to 'local' styles by editing the op-amp's **U1** component reference label.

Lets start by taking a look at a local style: click right on the **U1** label to tag it (and the op-amp) and then click left on it to edit it causing the *Edit Terminal Label* dialogue form to be displayed. Finally, select the *Style* tab to see the 'local' style settings. You will see the following: a *Global Style* selector at the top, a set of style attributes (*Font face*, *Width*, etc.) to the left and for each attribute a *Follow Global?* check box to the right. Both text and graphics local styles have this format. Wherever a *Follow Global?* checkbox is checked, the associated style attribute is initialised from the global style indicated in the *Global Style* selector. Unchecking the *Follow Global?* checkbox enables the dialogue control for the associated style attribute allowing you to specify a local (object-specific) value for that attribute. Not all the *Follow Global?* checkboxes are enabled as, in this case, the *Width* attribute is meaningless for a TrueType™ font face.



The change we are going to make is to make the **U1** label a different colour. Start by unchecking the *Follow Global?* checkbox to the right of the *Colour* control and you will see the *Colour* control is enabled and displaying the current colour set in the whatever global style this local style is following (in this case, the *COMPONENT ID* style). Change the colour to be dark blue, then close the form and untag the op-amp so that it is redrawn in its normal colours. The **U1** label is now dark blue.

To verify that the label's colour is indeed now independent of the global style but that the remainder of the attributes are still following its global style, select the *Set Text Styles* command on the *Template* menu again, change the colour of the *COMPONENT ID* style (selected by default) to magenta, check (enable) the *Italic* option and then close the form. The op-amp's component label remains dark blue as its colour is not longer following the global *COMPONENT ID* style however its *Italic* attribute is still following the global style, so it is now italicised. All the other components' IDs, which have remained following the global *COMPONENT ID* style, have both changed colour and become italicised.

For completeness, we will now show you how to edit a local graphics style. Select the *Box* icon. Now make sure the *COMPONENT* style is selected and then, placing the mouse at the top left of the circuit, click left and drag out a box that surrounds (and covers!) the circuit diagram. We now need to change the local style of the box to be transparent. Tag the box and then click left on it to edit it - the *Edit Box's Graphic Style* dialogue form is displayed. Apart from the different style attributes, notice how this form is similar to that of the *Edit Terminal Label* dialogue form's *Style* tab. The operation of this form is the same as the operation of that form. Uncheck the *Follow Global?* checkbox for the *Fill Style* control and then, it now being enabled, change the fill style to 'none'. Close the form with the *OK* button and untag the box.

This is a simple example of the power of local and global styles though in general use of ISIS you will never need to modify a local style. The main use of local styles is with 2D graphics objects (lines, boxes, etc.) when creating a new library part and even then it is recommended that you avoid setting up local styles unless absolutely necessary. As you have seen, once an object has a local setting its appearance is fixed unless it is re-edited and, when applied to new library parts, this means that some or all of a library part will not blend in to the drawing in which it is used. In proper use, this is usually the intention - for example, that the base of a transistor always have a solid fill regardless of the fill selected in the global *COMPONENT* graphics style. However, when misused, it can lead to library parts that simply don't port between different colour schemes or users.

## ***Working With The Template***

We will finally touch on the subject of templates. All the information about text styles, graphics styles, junction dot shape and size, paper colour, etc. - in fact everything you can

change via the *Template* menu commands - is called a design template. When installed, ISIS comes with a default template that is stored in the file **DEFAULT.DTF** in the library directory of your Proteus installation. When ISIS is started or you select the *New* command, ISIS loads its initial template settings from this file and any modifications made to the template settings are only saved back to this file if you select the *Save Default Template* command from the *Template* menu. The existing template is not backed-up so don't *Save Default Template* unless you particularly want to reuse the template settings in the future (for example, you have a new 'house' style you want all your designs to conform to).

For portability and consistency, ISIS also stores a copy of the current template in schematic design files and recovers it when the file is loaded. You've already seen this behaviour in loading the tutorial design and if you (later) load some of the other sample designs supplied with ISIS you will see that they employ a wide variety of styles.

So what happens if you set up a 'house' style, use *Save Default Template* to make it the default template, design and save lots of schematic designs, and then decide to change the 'house' style? The simplest answer is to use either the *Apply Template From Design* or *Apply Default Template* command from the *Template* menu. Both these commands load a new template in to the current design, either from another design file or from the default template file though note that objects with local style settings will continue to use them. In this way you can set up a new default template and then apply it to all your old designs. Alternatively, if you lose your default template, you can recover it from an old design and then save it using the *Save Default Template* command.

## **TEMPLATES AND THE TEMPLATE MENU**

All the information that controls the appearance of a schematic - graphics styles, text styles, design colours, wire junction dot size and shape, etc. - is termed the *template*. All the commands for modifying the template are on the *Template* menu are described subsequently.

Note that any changes to the template only affect the currently running copy of ISIS though they will be saved and preserved in any schematic design saved. To make the changes available the next time you start a design you need to use the *Save Default Template* command on the *Template* menu to update the default template.

## INTRODUCTION

ISIS makes extensive use of the concept of properties. A property is considered to consist of a *keyword* which identifies a particular property and a *value* which is assigned to that property for a particular object. For example, in conjunction with ARES, we make use of a property called **PACKAGE** which contains the PCB footprint.

Properties can be associated with objects, sheets and the design itself and the relationships between the various types need to be clearly understood if you are to get the best from what is an extraordinary powerful scheme, unmatched by any other schematics software we have seen.


## OBJECT PROPERTIES

There are two distinct types of object property - system properties and user properties. The former comprise a set of reserved keywords which have special functions within ISIS itself, whereas the latter relate either to external programs such as ARES and ProSPICE, or may relate to your own particular use of the software.

### System Properties

System properties are properties whose keywords have special meaning within ISIS. For example, the **DEVICE** property of a component object directly determines the library part assigned to it. Some of these properties are textual - e.g. component **REF** and **VALUE** labels are directly accessible from the *Edit Component* dialogue form, but others such as the aforementioned **DEVICE** property are manipulated as a result of graphical operations.

In general, you need only concern yourself with system properties if you wish to read their values with the search and tag commands, or modify them with the *Property Assignment Tool*. For example, you might wish to tag all the 7400 components in your design. This requires you to know that the system property that will hold the library part name is called **DEVICE**.

 Details of the system properties used by each type of object are given in *Object Specifics* on page 63.

### *User Properties*

Components, sub-circuits and VSM gadgets can carry an unlimited number of extra properties, in addition to their standard system properties. These user properties are held in a block of text known as a *property block* and consisting of lines such as:

```
SUPPLIER=XYZ Electronics
```

You can edit these user property blocks directly on the dialogue forms of the various objects, as well as manipulating them with the *Property Assignment Tool*.

#### **To edit an objects user properties**

1. Bring up the object's dialogue form by tagging it and clicking left.
2. If the object can hold user properties, the dialogue form will have a text editor box labelled *Properties*. Point below any existing text in this box and click left.
3. Edit the text as required. Each property should consists of a single line comprising a keyword and value separated by an equals sign (=).

User property keywords should, as a rule, consist of letters, numbers and underscores only. *Under no circumstances should they contain spaces, commas, double quotes or equals signs (, " =).*

In accordance with the general behaviour of ISIS text, if a property assignment is enclosed in curly brackets ('{' and '}') then it will not be displayed on the screen. For example, entering:

```
{PRIMITIVE=DIGITAL}
```

will define an object as requiring digital simulation, but this text will not actually appear.

Occasionally, one may want just the value to appear, in which case you can do:

```
{MODFILE=}OPAMP
```

In theory you can put curly brackets in other places. However, when the *Property Assignment Tool* modifies property blocks, it works on the assumption that if used at all, curly brackets have been placed as shown in the above examples. If they have been placed elsewhere, then you may get unexpected results.

### **Property Definitions (PROPDEFS)**

It is possible to provide extra details for specific user properties of a device. For example, common user component properties are **PACKAGE** and **MODFILE**. Given that appropriate Property Definitions have been created for the component libraries, these properties are displayed in their own fields on the *Edit Component* dialogue form. The property definitions include a description of the property, a data type (such as integer, floating point or string)

and for numeric data types a valid input range. A default value may also be (and often is) specified.

This scheme makes it much easier to see what properties are valid for a particular model, and what those properties mean. It is also then possible to supply alternate package types and simulator models in a way that is obvious to the end user of a particular library part.

For further information on how to create and manage property definitions for your own library parts, see page 60.

Properties which are not known for a particular device (or all properties for a device which does not have property definitions) still appear in the textual properties block as described above, and the scheme is thus backward compatible with old designs and/or library parts which do not have property definitions.

## ***SHEET PROPERTIES***

### ***Introduction***

Each sheet of a schematic can hold a set of property assignments. These may be considered as defining constants (either numeric or textual) which may be used within object property assignments on the particular sheet. In themselves they are not terribly useful, but their real power comes to light when they are used in the context of object property expressions.


For example, if a sheet property is defined in a block such as:

```
*DEFINE  
PI=3.142
```

you could then define a resistor as having the value:

```
VAL=EVAL(500/PI)
```

At netlist time, this syntax causes the netlist compiler to evaluate the property, and the resistor's value will appear in the netlist or bill of materials appear as 159.134.

 For further information about property expressions see *Property Expression Evaluation* on page 54.

### ***Defining Sheet Properties***

Sheet properties can be defined in the following ways:

- Directly, using a **DEFINE** script block. You can use this to define constants for use in expressions, as in the above example.

- As a result of parameter mapping using a **MAP ON** script block. In this case, the value of the parent property specified by the **MAP ON** statement is used to select a set of sheet property definitions from a table. This is most commonly used for creating universal simulation models, in which several similar devices are modelling using the same circuit but with different sets of sheet properties for each device type.

Further discussion of this is given in the Proteus VSM manual.

- By inheriting the properties of a parent object. In other words, if the parent object has the object property assignment:

R3=10k

then the child sheet will automatically acquire this as a sheet property. This provides the basis for parameterized circuits (section [ *Parameterized Circuits* ]) in which several instances of a given hierarchy module can be assigned different component values.

If the same property is defined both on a child sheet in a **DEFINE** or **MAP ON** block and in the parent object property block, the value from the parent will be used. This provides the means to provide default values for sheet properties which can be overridden as required.

### Scope Rules for Sheet Properties

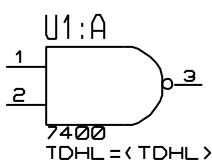
It is important to realise that sheet properties may only be referenced on the sheet for which they are defined. In particular, the sheet properties on a parent sheet are not accessible by any of its children unless they are passed though the parent object property block. If you do need to access a property in this manner, you can add a line such as:

```
TDHL=<TDHL>
```

to the appropriate parent object(s). If **TDHL** is defined as a sheet property on the parent sheet, it will then become an object property of the parent object, and thus be defined as a sheet property for the child sheet, where it can in turn appear in further object property expressions.

This arrangement is analogous to passing parameters in a C computer program.

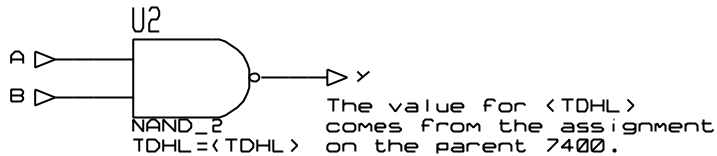
### PARENT SHEET



```
*DEFINE  
TDHL=10n
```

The value for <TDHL> here comes from the sheet property defined in the \*DEFINE block.

## CHILD SHEET



## DESIGN PROPERTIES

The design properties for a given schematic are determined by accumulating all the sheet properties that are declared on the root sheets of the design. Since root sheets cannot have parents, it follows that design properties can only be declared using **DEFINE** script blocks on the root sheets.

For SDF netlist output, the design properties appear in the **PROPERTIES** block of the netlist, and may be interpreted by whatever application is reading the netlist. In the case of Proteus VSM, design properties are used to define simulator options such as the number of steps, the operating temperature and so forth. The specifics of this are given in the Proteus VSM manual.

### To create a list of design properties:

1. Go to the root sheet of your design by selecting the *Goto Sheet* command from the *Design* menu.
2. Select the *Script* icon from the *Mode Selector* toolbar.
3. Point where you want the property definition block to appear and click left.
4. Type in the first line of the script as
 

```
*DEFINE
```
5. Type in the property assignments as required.

Note that design properties are also sheet properties for the sheets on which they are defined. However, the standard scope rules for sheet properties still apply, and design properties are not accessible in property expressions on other sheets.

## PARAMETERIZED CIRCUITS

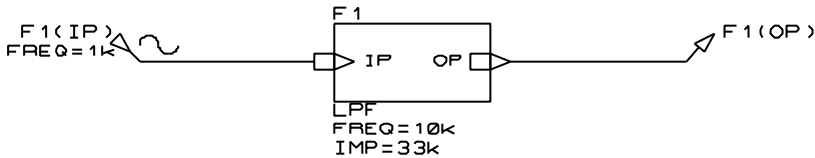
## Introduction

ISIS has a unique and tremendously powerful feature which combines sheet properties, object properties and hierarchical design to facilitate parameterized circuits. A parameterized circuit is one in which some of the component values (or other properties) are given as formulae rather than as constant values. Naturally, the formulae contain variables or *parameters* and the values for these are taken from the sheet properties defined for that particular instance of the circuit. It then follows that, in the context of a hierarchical design, different instances of the circuit can have different parameters and therefore different component values.

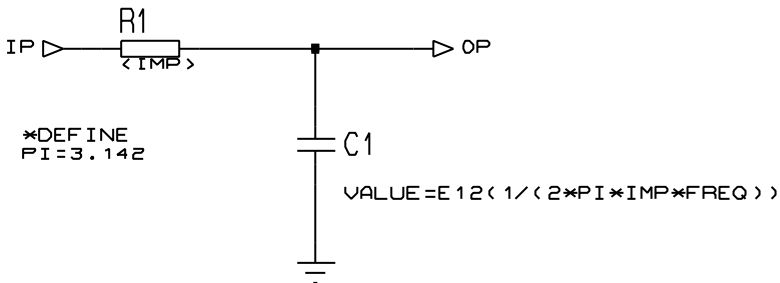
For more information about hierarchical design see *Hierarchical Designs* on page 117. If you have no idea what a hierarchical design is, we suggest you skip this section entirely for the time being!

## An Example

Parameterized circuits are best illustrated by an example design which you will find as LPF.DSN in the "Samples" directory of your Proteus installation. The root sheet of this design is shown below:



It consists of a single sub-circuit which has two user properties defining the desired frequency and impedance for the filter. The module attached to the sub-circuit is the actual parameterized circuit:



There are several points to note about it:

- The **DEFINE** block defines one sheet property: **PI**. This serves as a constant in the property expression for the capacitor value.



- The resistor's value field contains the string <IMP>. This syntax with the chevrons (<>) causes the netlist compiler to substitute the <IMP> with value of the **IMP** sheet property, that is 33k. In this case, no expression evaluation is performed - the substitution is purely literal.
- The capacitor has a user property assignment (in its property block) which specifies its value as an expression. The **E12** function specifies both that the expression should be evaluated by the netlist compiler, and also that it should be rounded to the nearest E12 value. The other options are **EVAL** (no rounding) and **E24** (round to **E24** value).

**PI**, **IMP** and **FREQ** are all sheet properties. **PI** comes from the **DEFINE** block whilst **IMP** and **FREQ** come from the parent sub-circuit.

If you generate the Bill of Materials, you will see the following:

QTY	PART-REFS	VALUE
---	-----	-----
Resistors		
-----		
1	R1	33k
Capacitors		
-----		
1	C1	470p

ISIS has evaluated  $1/(2*3.142*33000*10000)$  to get approximately 0.00000000482 and then rounded this to the nearest E12 value - 470p.

There are in fact two distinct processes going on in the above example:- *property substitution* and *property expression evaluation*. Both have their advantages and disadvantages and the following sections discuss each in more detail.

### Property Substitution

This is the mechanism that is used for the resistor value and will operate whenever the netlist compiler encounters a property value containing a property keyword enclosed in chevrons (<>). If the keyword matches a sheet property, then the expression in chevrons is replaced with the value of the sheet property. If no sheet property exists, then a netlist warning is generated and the property is removed from object.

There are two main cases where property substitution is useful:

- You can use it in a parameterized circuit in which the parameters are not numeric. Packages for PCB design are perhaps the most common example of this - it is all very well getting ISIS to compute that a 470pF capacitor is to be used, but you may still need to package it for PCB design. If you attach to the capacitor the user property:

```
PACKAGE=<C1_PACKAGE>
```

then you can add the property:

```
C1_PACKAGE=CAP10
```

to the sub-circuit. When netlisted, the C1 will appear with the property

```
PACKAGE=CAP10
```

Property expression evaluation cannot be used for this, because CAP10 will not evaluate as a number.

- The other major use for property substitution is for setting up sweep analyses with Proteus VSM. In this case, you want the simulator to evaluate the expressions, not ISIS and it may then be appropriate to build up component properties using property substitution rather than property expression evaluation. Further discussion of this is given in the VSM manual.

### ***Property Expression Evaluation***

In contrast to property substitution which is a purely textual process, property expression evaluation involves ISIS in numerically evaluating a property value that is in the form of a formula, and replacing it with its value. In addition, ISIS can also round the resulting number to an E12 or E24 series value.

There are three forms of syntax:

```
EVAL ( . . . )
```

```
E12 ( . . . )
```

```
E24 ( . . . )
```

In all cases, the parentheses should contain a mathematical expression which may include the operators plus (+), minus (-), times (\*) and divide (/) and values. Values can be constant numbers, or the names of sheet properties. Multiplication and division have higher precedence but further levels of parentheses may be used to override this as required.

Some example expressions are shown below:

```
EVAL( 1 / ( A+B ) )
```

A and B are sheet properties.

```
E12( 20k+2*F*PI )
```

20k automatically treated as 20000.

```
E24( 3+4*5 )
```

Evaluates to 24.

---

Although in some ways more powerful than property substitution, there are some limitations:

- The evaluator can only handle numeric values - expressions involving strings are not allowed.
- You can only reference sheet properties in the formula - you cannot access other object properties or the values of other components.
- There is **no** support for mathematical functions (e.g. sin, cosine, square-root) etc.

We may eliminate some or all of these deficiencies in a future version.

### ***The Rounding Functions E12 (), E24 ()***

The property expression evaluation mechanism supports the ability to round the resulting value to the nearest E12 or E24 value. This prevents parameterized circuits ending up with non-available or multi-digit floating point values.

You should note:

- Rounding will be disabled for expressions that yield zero or negative values. This is unlikely to be a problem as negative valued resistors and capacitors are hard to come by anyway.
- The rounding is done on a geometric rather than an arithmetical basis so the mid-point between 3k3 and 4k7 is taken to be approximately 3.94. This, we feel, is in keeping with the thinking behind the E12 and E24 series themselves.
- Bear in mind that if a parameterized circuit contains several rounded values, there is no mechanism to round them all in the optimum directions, taking into account the way they may interact. This means that for critical filter designs and such like, you may be better computing the values manually (considering the various possible mixes of values) and using parameter substitution to load your chosen values directly into the circuit.

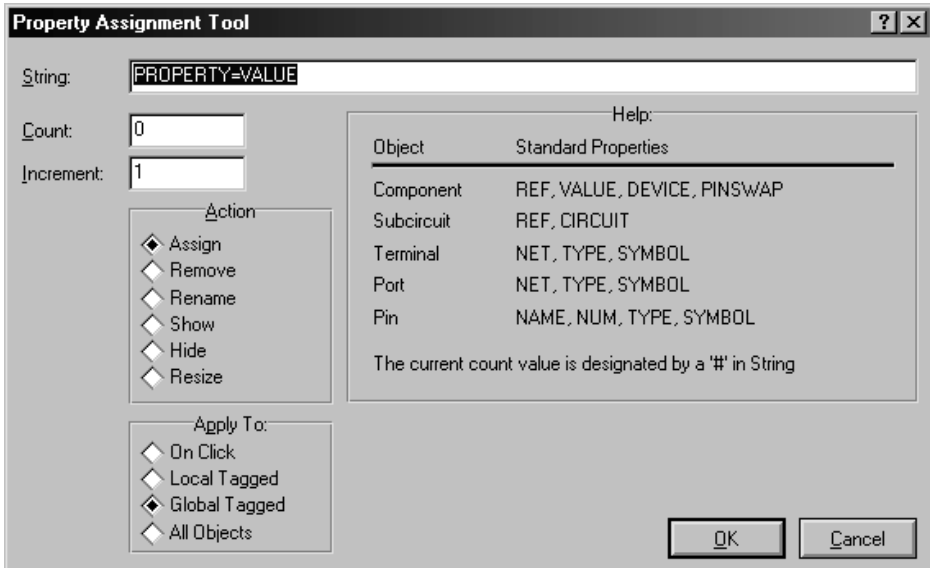
Of course, if you have Proteus VSM, you can run a simulation and see how the values ISIS has chosen affect the circuit performance.

## ***THE PROPERTY ASSIGNMENT TOOL***

The *Property Assignment Tool* (PAT) is a feature which enables you to assign, remove, show and hide object properties belonging selected objects in your design.

## The PAT Dialogue Form

The PAT is operated from a modestly complex dialogue form with the following fields:



- String**                    The property assignment, or property keyword on which the selected action will be performed for each object.
- Count**                    The initial value for the counter. The counter is incremented on each application of the PAT. The current value of the counter may be included in the string by entering a hash ('#') character.
- Action**                    The action you wish to perform. See *Pat Actions* on page 56 for descriptions the various types of action.
- Apply**                    The application mode in which you want the PAT to operate. See *Pat Application Modes* on page 58 for descriptions the various types of action.

## PAT Actions

The *Property Assignment Tool* can perform the following actions:

- ASSIGN**                    The string should contain a property assignment of the form:  

*keyword=value*

and this property will be assigned to the selected objects.

If you want to assign values that run in a sequence, such as D0, D1, D2 etc. then use the hash (#) character in the value and set the initial count as required.

Both user and system properties may be assigned; assigning system properties may cause graphical changes to your drawing.

**REMOVE**

The string should contain a property keyword only and this property will be removed from the selected objects.

Only user properties can be removed.

**RENAME**

The string should contain both a property assignment of the form:

```
current_keyword=new_keyword
```

The name to the left of the assignment is the existing property name you wish to rename; the name to the right is the new name you wish to rename it to.

Only user properties can be renamed.

**SHOW**

The string should contain a property keyword only and this property will be made visible for the selected objects.

All user properties and system properties can be shown.

**HIDE**

The string should contain a property keyword only and this property will be made invisible for the selected objects.

All user properties and system properties can be hidden.

**RESIZE**

The string should contain an assignment such as

```
REF=20,16
```

and the given property will be assigned a new height and width for the selected objects. Only textual system properties may be resized.



Changing a labels text size implicitly sets the *Height* and *Width* attributes of the labels text style to be local and that, thereafter, changes to the global text style for the label will not be reflected in these labels.

## ***PAT Application Modes***

The selected PAT action can be applied to the design in the following ways:

**ON CLICK** On selecting the OK button, the *Instant Edit* icon selected. Each object on which you click left has the selected action performed upon it.

When you select a different icon, the PAT is cancelled.

This mode is the only way to apply the PAT to wires, in order to assign wire labels. ISIS cannot perform PAT assignments on tagged wires because there would be no way for it to know where to put the wire labels.

**LOCAL TAGGED** The selected action is performed on all tagged objects on the current sheet only.

You can select the tagged objects either individually, or with the *Search & Tag* commands.

**GLOBAL TAGGED** The selected action is performed on all tagged objects right across the entire design.

You can select the tagged objects either individually, or with the *Search & Tag* commands.

ISIS is capable of remembering different tag states for different instances of an object in a hierarchical design, should this be an issue. However, the user property blocks are shared between instances so you cannot change a user property in just one instance.

**ALL OBJECTS** The selected action is performed on all the objects in the design.

## ***The Search and Tag Commands***

The search and tag commands facilitate the selection of particular groups of objects for subsequent processing by the *Local Tagged* or *Global Tagged* options of the PAT.

There are three search commands:

**SEARCH** This is the normal search operation which you should use to begin all new search tasks. It sets the tags of objects which meet the search condition and clears the tags of objects which do not.

**AND SEARCH**

This search operation can be used to eliminate objects from the currently tagged set. It clears the tags of all objects which do not meet the search condition, and leaves alone the tags of objects which do.

**OR SEARCH**

This search operation can be used to include further objects into the currently tagged set. It sets the tags of all objects which meet the search condition, and leaves alone the tags of those which do not.

**Examples**

The *Property Assignment Tool* and *Search & Tag* commands provide great power and flexibility when it comes to manipulating object properties. However, they can seem somewhat daunting to beginners. With this in mind, some step by step examples to get you started are given overleaf.

**To label a set of bus taps**

1. Invoke the PAT by keying 'A'.
2. Set the string to `NET=D#` and click OK. The action will have defaulted to *Assign* and the mode to *On Click*.
3. Click left on each wire where you want the wire label to appear. The cursor keys and the enter key can be used as an alternative to the mouse. The wires you click on will be assigned new wire labels in the sequence D0, D1, D2, etc.

**To assign a package to all the BC108s in a design**

1. Invoke the *Search and Tag* command by keying 'T'.
2. Set the property to `VALUE` and the string to `BC108`, then click OK. The mode will have defaulted to *Equals*. All the components with the value BC108 will be tagged.
3. Invoke the PAT by keying 'A'.
4. Set the string to `PACKAGE=TO18` and click OK. The action will have defaulted to *Assign* and the mode to *Global Tagged* (assuming there were tagged BC108s). All the tagged BC108s will be given the new user property.

### To rename all ITEM properties to CODE properties:

1. Invoke the PAT by keying 'A'.
2. Set the string to ITEM=CODE, the action to *Replace* and the mode to *All Objects*, then click OK. All objects with the property ITEM=*value* will be replaced with a property CODE=*value*.

### To hide all the package properties

1. Invoke the PAT by keying 'A'.
2. Set the string to PACKAGE, the action to *Hide* and the mode to *All Objects*, then click OK. All the **PACKAGE** properties will be hidden.

### To resize the component references

1. Invoke the PAT by keying 'A'.
2. Set the string to REF=10 , 8, the action to *Resize*, and the mode to *All Objects*, then click OK. All the component references will be shrunk to the new size.

### To assign larger packages to 1000uF capacitors

1. Invoke the *Search & Tag* command by keying 'T'.
2. Set the property to DEVICE and the string to CAP ELEC, then click OK. This will tag all the electrolytic capacitors.
3. Invoke the *AND Search* command from the Tools menu.
4. Set the property to VALUE, the string to 1000u, and the mode to *Begins*, then click OK. This will take care of capacitors with the values 1000u or 1000uF.
5. Invoke the PAT by keying 'A'.
6. Set the string to PACKAGE=ELEC-RAD30 and click OK. The action will have defaulted to *Assign* and the mode to *Global Tagged* (assuming there were tagged capacitors). All the tagged capacitors will be given the new package.

## PROPERTY DEFINITIONS

### Creating Property Definitions

Property definitions are entered using the *Component Properties* page in the *Make Device* dialogue form. For more detailed information use the context sensitive help on the dialogue form and/or refer to the instruction on making a device on page 97.



## ***Default Property Definitions***

A number of properties will be applied to most devices that you create. For example, anything that is going onto a PCB will need a **PACKAGE** property, and anything that has a simulator model will need a **MODFILE**, **MODEL** or **SPICEMODEL** property. You may also want to apply your own properties such as **STOCKCODE**, **SUPPLIER** or **COST** to most of the devices that you create.

To make this easier, there is a list of default property definitions which can be defined using the *Set Property Definitions* command on the *System* menu. The properties defined with this command are available on the *Name* field combo on the *Edit Device Properties* dialogue form.

The information manipulated by this command is held in the file PROPDEFS.INI located in library directory of your Proteus installation.

## ***Old Designs***

Note that components and library parts in designs created with PROTEUS versions prior to 4.5 will not contain property definitions. If you wish, you can have ISIS apply the default property definitions to such components 'on the fly' i.e. as they are edited.

To enable this feature, invoke the *Set Property Definitions* command from the *System* menu and enable the *Apply Default Properties to Components in Old Designs* checkbox.



# OBJECT SPECIFICS

## COMPONENTS

A component is an instance of a device, picked from one of the device libraries. Since some devices are *multi-element* it follows that in some cases, several components on the schematic may, in fact, all belong to one physical component on the PCB. In such cases, the logical components are annotated with names such as U1:A, U1:B, U1:C, U1:D to indicate that they all belong to the same physical part. This form of annotation also enables ISIS to select the correct set of pin numbers for each element.

Apart from physical terminals, components are the only objects you can place which will generate physical entities in the PCB design. Everything else you place serves either to specify connectivity, or else is present only for the purpose of human readability. This is an important point, since attempts to use other ISIS objects (in particular, graphics symbols) to represent objects on the PCB will fail.

### **Selecting Components from the Device Libraries**

When you start ISIS on an empty drawing, the device selector is empty. Before you can place any components, you must first pick them from the libraries into the selector.

#### **To select components from the device libraries:**

1. Ensure that the *Object Selector* is showing devices by selecting the *Component* icon from the *Mode Selector* toolbar.
2. Click left on the **P** button on the *Object Selector*. This will cause the *Device Library Pick Form* to appear.
3. In the *Library* selector, select the library you wish to pick devices from.


The *Prefixes* selector will be displayed or hidden according to whether or not the library you have selected contains prefixes for its parts (see below).


The *Extensions* selector will be displayed or hidden depending on whether or not the parts in the library you select have two or more extensions in their names. An extension is any text at the end of the part name following a dot or full-stop and serves to differentiate different versions of the same device (e.g. normal, DeMorgan, IEC-617, etc. parts). Where the library only contains parts with out an extension or only contains parts with the same extension, the *Extensions* selector is not shown.

The *Objects* selector is updated to show the contents of the library according to the settings of the *Extensions* selector. Where the *Extensions* selector is hidden, all library parts will be shown.

4. Where the *Extensions* selector is shown, check or uncheck one or more of the extensions to show or hide parts with that extension.
5. Where the *Prefixes* selector is shown select the correct prefix for the part you require.  
When you pick the part, the currently selected prefix will be inserted in front of the part name. For example, with the TTL.LIB library selected and the 74LS prefix selected in the *Prefixes* selector, picking a 249 part will in fact pick a 74LS249.
6. Single click a part in the *Objects* selector to browse the part in the *Browser* window or double-click a part in the *Objects* selector to pick it in to the design.
7. When you have finished picking devices, click the *Minimise* button of the pick form to

An alternative method for picking library parts is to use the *Pick* command, key 'P'. With this command, you can type in the name of the device you want directly.

 You can resize and position the pick form window and then save its position via the *Save Window Position* command on Windows 'system' menu (click on button on the far left of the window's title bar).

 Further information about libraries, and the procedures for making your own library parts is given in *Library Facilities* on page 91.

## Placing Components

Component placement follows the general scheme outlined in *Object Placement* on page 29.

### To place a component:

1. If the device type you want is not listed in the *Object Selector*, first pick it from the libraries as described in the previous section.
2. Highlight the device name in the *Object Selector*. ISIS will show a preview of the device in the *Overview Window*.
3. Use the *Rotation* and *Mirror* icons to orient the device according to how you want to place it.
4. Point at the position in the *Editing Window* where you want the device to appear, and click left. If you hold the mouse button down, you can drag the device around until you have it exactly where you want it. When you release the mouse button, it will be placed onto the drawing.


## Replacing Components

Since deleting a component removes the wires attached to it, changing a component by deleting it and then placing a new one can be somewhat tedious. Instead ISIS provides a special mechanism to facilitate this type of operation.

### To replace one component with another:

1. Pick the new device type and highlight its name in the *Object Selector* as described above.
2. Use the *Rotation* and *Mirror* icons to orient the new device according to how you want to place it.
3. Position the mouse pointer inside the old component, and then place the new one such that at least one of its pin ends touches one of the pin ends of the old component. *The mouse pointer must have been inside the old component when the new one was placed, if auto-replace is to be activated.*

ISIS will attempt to replace the old component with the new one whilst preserving as much of the old wiring as it can. It matches pins first by position, and then by pin name. Attempts to over-place wildly different parts are unlikely to give useful results, but the *Undo* command will recover the situation if such a replace is performed by accident.

 It is also possible to effect a replacement by assigning to the **DEVICE** property using the *Property Assignment Tool*. See *Properties* on page 47 for details of how to use this feature.

## Editing Components

A component may be edited using any of the general editing techniques (see *Editing An Object* on page 32), and also specifically by using the *Edit Component* command on the *Edit* menu, key 'E'. As with most editing features in PROTEUS a dialogue form will appear with the appropriate editing fields. However, with the *Edit Component* dialogue form the available fields are dependant on the given properties of the component. You will find, therefore, that this dialogue form can be radically different for different components. Context Sensitive Help is available for the dialogue form but please note that, for the reasons given above, some fields may not have a help topic associated with them.

You can customise the appearance of components, pins, pin names, etc. by editing the associated global graphics and text style. See *Editing Global Styles* on page 42 for more information.

## Component Properties

Components have the following system properties:

Property Name	Description
REF	Component reference designator label.
VAL	Component value label.
VALUE	Component value label or the VALUE user property if the assigned text is too long for the label.
DEVICE	Library part. If assigned, this will invoke the automatic replacement logic which will attempt to maintain connectivity.

If any other property names (e.g. STOCKCODE, TOLERANCE, etc.) are assigned, then these will create user properties in the component's text block.

### Hidden Power Pins

If a component has been defined with hidden power pins, then the nets to which these will be connected can be viewed or edited by clicking the *Hidden Pins* button on the *Edit Component* dialogue form.

By default, hidden pins connect to a net of the same name - for example, a hidden VDD pin will connect to VDD, a hidden VSS pin to VSS and so on.

### DOTS

Junction dots are used to mark interconnection between wires. In general, ISIS will place and remove them automatically but it is sometimes useful to place a dot at a specific position and then route wires to and from it.

Wires which touch or cross are *never* considered connected unless there is a dot at the point of contact. Conversely, where there is a dot, there will always be a connection unless you have contrived to superimpose wires and dots by dragging or moving them into contact with each other.



You can customise the appearance of junction dots by editing the *WIRE DOT* graphics style. See *Editing Global Styles* on page 42 for more information. You can also set their size and shape using the *Set Junction Dots* command.

## ***Placing Dots***

### **To place a dot:**

1. Select the *Dot* icon from the *Mode Selector* toolbar.
2. Point where you want the dot to be placed in the *Editing Window*.
3. Click left to place the dot.

## ***Auto Dot Placement***


ISIS will automatically place a dot whenever you route a wire off an existing wire such that there are then 3 dots at that point.

## ***Auto Dot Removal***

When a wire or wires are deleted, ISIS will detect where this leaves dots with two or no connected wires. Such dots are automatically removed unless this would result in an unbroken wire loop.

## ***WIRE LABELS***


Wire labels are used to assign particular net names to groups of wires and pins, and also to assign *net properties* to specific nets. They are not in fact proper objects in the general scheme of the software. Instead, their behaviour is similar to other labels such as the reference and value labels of a component. It follows that the procedures for their placement and removal are somewhat different from those for other objects.

 See *Netlist Generation* on page 123 for information about the role of wire labels in netlisting and connectivity.

## ***Placing And Editing Wire Labels***

### **To place or edit a wire label:**

1. Select the *Wire Label* icon from the *Mode Selector* toolbar.
2. If placing a new label, point at the wire at the position where you want the label to be placed or, for an existing label, point anywhere along the wire or at the label itself.
3. Click left to place the label. The *Create Wire Label* or *Edit Wire Label* dialogue form will be displayed.

 See *Editing Local Styles* on page 44 for more details of the controls on the *Style* tab.

4. Type in the required text for the wire label.
5. Click OK or press ENTER to close the dialogue form.

Note the following:


- You cannot place a wire label other than on a wire.
- You can place more than one label on a wire. If you wish them all to have the same name, and for all of them to update automatically whenever any of the names are changed, the click the *Auto-Sync* checkbox.
- ISIS will orient the wire label according the orientation of the wire segment on which it is placed. This can be changed on the *Edit Wire Label* dialogue form.


### To change the appearance of a wire:

1. Ensure the *Wire Label* icon is not selected.
2. Tag the wire by pointing at it and clicking right.
3. Click left on the tagged wire.

The *Edit Wire Style* dialogue box is displayed.

4. Uncheck the *Follow Global?* checkboxes of those style attributes of the graphics style you want to change. If a style attribute and its *Follow Global?* checkbox are both disabled it is because the style attribute is not meaningful given other attribute settings, either locally or through following the global style attribute.
5. Set the style attribute to the required setting.
6. Press ENTER or click the OK button to close the dialogue form and keep the changes. Press ESC or click the CANCEL button to close the dialogue form and abandon changes.

 To change the appearance of all wires on the schematic, use the *Set Graphics Style* command on the *Template* menu to edit the *WIRE* graphics style.

 See the section *Graphics And Text Styles* on page 41 for information on styles.

### ***Deleting Wire Labels***

#### To delete a wire label:

1. Tag the wire and wire label by pointing at either and clicking right.
2. Bring up the *Edit Wire Label* dialogue form by clicking left on the **label**.




Note that clicking left on the tagged wire or bus will, except when the *Wire Label* icon is selected, bring up the *Edit Wire Style* dialogue. To display the *Edit Wire Label* dialogue you must click on the label itself.

3. Ensure the label string text is fully selected (it is by default) and press DEL to delete it.
4. Click OK or press ENTER to close the dialogue form and save changes.

### **Using a Wire Label to Assign a Net Name**

The normal use of wire labels is to indicate that a particular wire, and all the pins to which it is graphically connected, belong to a given net. Then, that group of pins are deemed connected to any other groups of pins which have also been assigned the same net name, even where there is no graphical connection. Occasionally, one may also label particular nets purely for human benefit.

 See *Net Names* on page 123 for more information about net names.

### **Using a Wire Label to Assign a Net Property**


Net properties are used to assign special information to a net. In the PROTEUS system, the primary usage for this feature is to assign routing strategies for use in ARES. A net property assignment has the form:

<prop>=<value>

For example, a wire label with the string

STRAT=POWER

would result in the **STRAT** property for the connections being given the value POWER. In the ISIS/ARES context, this would result in ARES using the POWER strategy for the connections.

 See *Net Properties And Routing Strategies* on page 148 for more information about strategies.

### **Wire Label Properties**

A wire or wire label (they are equivalent for this purpose) has the following property:

Property Name	Description
NET	The wire label text.

Wire labels may only be assigned via the *Property Assignment Tool* when it is used in *On Click* mode. This is because there is no way for ISIS to determine where a wire label should be placed on a tagged wire, unless you mark the position with the mouse.

## SCRIPTS

A major feature of ISIS is its ability to support free format text scripts, and the numerous uses it puts them to. These uses include:

- Defining variables for use in property expressions and parameter mapping.
- Defining primitive models and scripts for use with the ProSPICE simulator.
- Annotating designs with substantial quantities of text.
- Storing property and packaging information when a component is decomposed.

### ***Placing and Editing Scripts***

The procedures for placing and editing scripts are almost identical as both operations involve the *Edit Script Block* dialogue form.

#### **To place a script :**

1. Select the *Script* icon from the *Mode Selector* toolbar.
2. In the *Editing Window*, point to where you want the top-left of the script to be and click left.

An *Edit Script Block* dialogue form is displayed.

2. Enter the text for the new script in the *Text* field. You can also adjust the attributes of a script at this time - see *To edit a script* below for full details of the dialogue form.
3. Close the dialogue form with the OK button to place the new script or with the CANCEL button to quit the form and cancel the script placement.

#### **To edit a script:**

1. Either:
  - (a) Tag the script by pointing at it and clicking right and the click left on the tagged script (without moving the mouse).
  - (b) Point at the script with the mouse and type CTRL+E to edit it.


The *Edit Script Block* dialogue form is displayed.

2. Adjust the script attributes as required.

The *Edit Script Block* dialogue form has two tabs: *Script* and *Style*. See the graphics and text style tutorial (*Editing Local Styles* on page 44) for more details of editing local text styles.

### 3. Close the form.

To close the form and save changes either click the OK button or use the CTRL+ENTER keys. To close the form and abandon changes, either click the CANCEL button or use the ESC key.

-  You can resize the *Edit Script Block* dialogue form to make the *Text* field bigger and then save the size using the *Save Window Size* command on the Window's 'system' menu (click on the button at the far left of the title bar).

## Script Block Types

ISIS currently supports the following script block types:

SCRIPT BLOCK TYPE	BLOCK HEADER
Part Property Assignment	<b>*FIELD</b>
Sheet Global Net Property Assignment	<b>*NETPROP</b>
Sheet Property Definition	<b>*DEFINE</b>
Parameter Mapping Table	<b>*MAP ON</b> <i>varname</i>
Model Definition Table	<b>*MODELS</b>
Named Script	<b>*SCRIPT</b> <i>type name</i>
SPICE model scripts	<b>*SPICE</b> <i>type name</i>

A brief description of the usage and format of each type is given in the following sections.

### Part Property Assignments (**\*FIELD**)


Field blocks are primarily intended to facilitate the assignment of properties to connectors made from *physical terminals*. A special means to do this is required since when physical terminals are used, there is no single entity on the schematic that represents the connector and to which properties could be directly attached.


An example block is shown below:

```
*FIELD
J1 , PACKAGE=CONN-D9
J2 , PACKAGE=CONN-D25
```

This assigns connector J1 the PCB package for a 9 pin D connector, and J2 the package for a 25 way connector.

You can assign properties to ordinary components as well, but there is generally little point since you can add them directly to the components.

 See *Physical Terminals* on page 81 for more information about physical terminals.

 See *Isis And Ares* on page 145 regarding packaging considerations for PCB design.

### **Sheet Global Net Property Assignments (\*NETPROP)**


A net property is normally assigned by the placement of a wire label with the syntax:

```
prop=value
```

However, there are occasions when you wish a large number of nets to carry the same property. Most often, the requirement is for all the circuitry on a particular sheet to be of a particular type (perhaps extra wide tracking for a power supply) and it is for this purpose that a **NETPROP** block may be used. For example, if the block:

```
*NETPROP  
STRAT=POWER
```

is included on a sheet, then all nets whose wires do not carry explicitly net properties will be assigned the net property **STRAT=POWER**.

 See *Using A Wire Label To Assign A Net Property* on page 69 for more information about net properties.


### **Sheet Property Definitions (\*DEFINE)**

A sheet property is essentially a variable which is defined on a given sheet and may be used in *property expressions* within components' property lists on that sheet. Sheet properties defined on the root sheets of a design, also appear in the netlist and may be used as control parameters by software such as the VSM simulators that reads the netlist.

An example block is shown below:

```
*DEFINE  
TEMP=40  
MINSTEPS=100
```

This defines two properties **TEMP** and **MINSTEPS**, and would be placed on the root sheet of a design such that they would be passed to VSM as simulation control parameters.

 See *Properties* on page 47 for an in depth discussion of property management.

## Parameter Mapping Tables (\*MAP ON varname)

This is considered a highly advanced topic, and one which is only of great importance if you are involved in creating universal models for use with VSM. Further information on this is given in the VSM manual, but for completeness, we provide an example PMT here:

```
*MAP ON VALUE
7400   : TDLH=12n , TDHL=7n
74LS00 : TDLH=10n , TDHL=6n
74S00  : TDLH=5n  , TDHL=3n
```

This table would be placed on the model schematic for a 2 input NAND gate, and would select different values for **TDLH** and **TDHL** according to the value (i.e. type) of the parent object. Also on the child sheet would be a NAND\_2 primitive with the properties:

```
TDLH=<TDLH>
TDHL=<TDHL>
```

At netlist time, the PMT would examine the parent object's **VALUE** property and if, say, it were 74LS00, then the sheet properties TDLH=10n and TDHL=6n would be defined for that instance of the model. Then, on processing the NAND\_2 primitive, these values would be substituted for <TDLH> and <TDHL> such that the primitive would acquire the correct timing to model a 74LS00.

A DEFAULT case is now supported.

## Model Definition Tables (\*MODELS)


These blocks are used only with the VSM simulators (currently, in fact, only with the Analogue simulator) and provide a shorthand method for managing the large numbers of properties that some of the simulator primitives use. For example, the bipolar transistor model has over 30 different properties and it would be fairly hopeless if you had to assign all of these individually for each transistor on the circuit.

Instead, you can use a **MODELS** block to define the properties for a particular type of transistor, and then refer to that set of properties using a model name.

An example block is shown below:

```
*MODELS
741_NPN : BETAF=80 , ISAT=1E-14 , RB=100 , VAF=50 , \
          TAUF=0.3E-9 , TAUR=6E-9 , CJE=3E-12 , CJC=2E-12
741_PNP : BETAF=10 , ISAT=1E-14 , RB=20 , VAF=50 , \
          TAUF=1E-9 , TAUR=20E-9 , CJE=6E-12 , CJC=4E-12
```

This defines the two transistor types used in a 741 IC. Individual transistors could then be given the values 741\_NPN or 741\_PNP and VSM would automatically give them the characteristics defined in the model table.

 See the VSM manual for further discussion of primitive models.

### ***Named Scripts (\*SCRIPT scripttype scriptname)***

In certain circumstances it is useful to be able to export a named block of text to an external application. In particular the SPICE and SPICE-AGE netlist formatters make use of the named scripts **PSPICE** and **SPICE-AGE** to supply control information for the simulations.

A typical named script block is shown below. Note that the end of the named script is marked by an **ENDSCRIPT** keyword, though this is optional if there are no other blocks in the script object.


```
*SCRIPT PROGRAM 7493

// Declare linkage to pins and local variables:
PIN CKA, CKB, RA, RB
PIN QA,QB,QC,QD
INT counta = 0, countb = 0

// Handle single-bit 'A' counter and assign output:
IF RA=H THEN
    counta = 0
ELIF CKA=HL THEN
    counta = (counta+1) % 2
ENDIF
QA = counta & 1


// Handle three-bit 'B' counter and assign output:
IF RB=H THEN
    countb = 0
ELIF CKB=HL THEN
    countb = (countb+1) % 8
ENDIF
QB = countb & 1
QC = countb & 2
QD = countb & 4

*ENDSCRIPT
```

 See the SPICE and SPICE-AGE application notes for information about the syntax of the named scripts associated with these simulators.

## ***SPICE Model Script (\*SPICE)***

These scripts allow SPICE input cards to be typed on the schematic and loaded directly into ProSPICE at simulate time. This can be handy if you want to develop or test SPICE models in native SPICE format in the native SPICE format.

 Further information on this is given in the Proteus VSM manual.

## ***BUSES***

A bus is a kind of shorthand for a large number of wires, and is commonly used on microprocessor schematics. ISIS has an unprecedented degree of support for buses, including not only the ability to run buses between hierarchy modules but also the facility to define library parts with *bus pins*. Thus it is possible to connect a CPU to an array of memories and peripherals by single bus wires, rather than a complex arrangement of wires, bus entries and buses.

### ***Placing Buses***

Buses are placed in very much the same way as ordinary wires except that they must run to and from bus connection points, rather than wire connection points. Also, unlike wires, buses may be placed in isolation from any other objects.


#### **To place a bus:**

1. Select the *Bus* icon from the *Mode Selector* toolbar.
2. Point where you want the bus to start. This may be a bus pin, an existing bus, or free space on the drawing.
3. Click left to start the bus, then click again at each corner of the desired bus route.
4. To finish the bus on a bus connection point (a bus pin or an existing bus) point at it and click left. To finish the bus in free space, click right.

### ***Bus Labels***

A bus may be labelled in exactly the same way as a wire. However, ISIS defines a special syntax for bus labels.

#### **To place or edit a bus label:**

1. Select the *Wire Label* icon from the *Mode Selector* toolbar.
2. Point on the bus at the position where you want the label to be placed.
3. Click left to place the label. The *Create Wire Label* or *Edit Wire Label* dialogue form will be displayed.  
 See *Editing Local Styles* on page 44 for more details of the controls on the *Style* tab.
4. Type in the required text for the wire label. This should be something like  $D[0..7]$  or  $A[8..15]$ . If you omit the range specification then the bus will take its base as zero and its width from the width of the widest bus pin connected to it. In general, however, you should always use range specification.
5. Click OK or press ENTER.

Note the following:

- You cannot place a wire label other than on a wire or a bus.
- You cannot place more than one wire label on one section of a bus. Attempting to do so will edit the existing wire label.
- ISIS will orient the wire label according the orientation of the wire segment on which it is placed. This can be changed on the *Edit Wire Label* dialogue form.

#### **To delete a bus label:**


1. Tag the wire and wire label by pointing at either and clicking right.
2. Bring up the *Edit Wire Label* dialogue form by clicking left on the label.  
Note that clicking left on the tagged wire or bus will, except when the *Wire Label* icon is selected, bring up the *Edit Wire Style* dialogue. To display the *Edit Wire Label* dialogue you must click on the label itself.
3. Ensure the label string text is fully selected (it is by default) and press DEL to delete it.
4. Click OK or press ENTER to close the dialogue form and save changes.



---

### To change the appearance of a bus:

1. Ensure the *Wire Label* icon is not selected.
2. Tag the bus by pointing at it and clicking right.
3. Click left on the tagged wire.  
The *Edit Wire Style* dialogue box is displayed.
4. Uncheck the *Follow Global?* checkboxes of those style attributes of the graphics style you want to change. If a style attribute and its *Follow Global?* checkbox are both disabled it is because the style attribute is not meaningful given other attribute settings, either locally or through following the global style attribute.
5. Set the style attributes to the required settings.
6. Press ENTER or click the OK button to close the dialogue form and keep the changes. Press ESC or click the CANCEL button to close the dialogue form and abandon changes.

 To change the appearance of all buses on the schematic, use the *Set Graphics Style* command on the *Template* menu to edit the *BUS WIRE* graphics style.

### **Wire/Bus Junctions**

Sometimes, even with the provision of bus pins, it is necessary to tap off a single signal from a bus, perhaps for decoding purposes. From a purely graphical point of view this is just a matter of placing a wire that ends on the bus:

#### **To place a bus tap:**

1. If you plan to route the wire *from* the bus to another object, ensure that the *Bus* icon is not selected.
2. Place the wire in the usual way. The bus will behave like an ordinary wire in these circumstances.

It may be helpful to you to understand that when you place a wire in this way, automatic dot placement will operate to give the wire and bus something to connect to. However, you will not see the dot as it adopts the same colour and width as the bus.

Having got a wire joined to a bus, you must specify which signal from the bus you are tapping.

### To annotate a bus tap:

1. Ensure that the bus carries a bus label such as D[ 0 . . 7 ]. This label would define eight nets called D0, D1, ... , D7.
2. Place a wire label on the wire to indicate which signal it is tapping.

If the bus doesn't connect to any bus pins or bus terminals, you can omit step [1]. In these circumstances, the bus itself plays no part at all in the determining the design connectivity.

*Placing a bus tap without annotating it will be reported as a netlist error as it is a wholly ambiguous situation. ISIS cannot know which signal you are trying to tap. Nor can a human reader for that matter!*

### Bus Properties

A bus or bus label (they are equivalent for this purpose) has the following property:

Property Name	Description
NET	The bus label text.

Bus labels may only be assigned via the *Property Assignment Tool* when it is used in *On Click* mode. This is because there is no way for ISIS to determine where a bus label should be placed on a tagged bus, unless you mark the position with the mouse.

## SUB-CIRCUITS

Sub-circuits are used to attach lower level sheets to higher level sheets in a hierarchical design. Each sub-circuit has a name that identifies the child sheet, and a circuit name that identifies the child circuit. On any given sheet, all the sub-sheets should have different sheet names, but may - and often will - have the same circuit names. More information about hierarchical design is given in *Hierarchical Designs* on page 117.


Sub-circuits can also have property lists, and this leads to the possibility of *parameterized circuits* in which different instances of a given circuit can have different part values (or other properties) as well as independent annotations. Further information about this powerful feature are given in *Parameterized Circuits* on page 51

### Placing Sub-Circuits

Placing a sub-circuit involves laying out the actual sub-circuit box, and then placing sub-circuit ports upon it. The same icon is used for both operations; ISIS determines what happens according to whether you point it free space, or at an existing sub-circuit box.

### To place a sub-circuit box:

1. Select the *Sub-Circuit* icon from the *Mode Selector* toolbar.
2. Point where you want the top left corner of the box. This must be a point not occupied by an existing sub-circuit.
3. Click left and drag out the box - then release the mouse button.

 You can customise the appearance of subcircuits by editing the *SUBCIRCUIT* graphics style. See *Editing Global Styles* on page 42 for more information.

### To place sub-circuit ports:

1. Select the *Sub-Circuit* icon from the *Mode Selector* toolbar.
2. Select the type of port you want from the *Object Selector*.
3. Point where you want the port. This must be a point on the left or right edge of the sub-circuit to which you want to attach the port.
4. Click left to place the port. ISIS will orient it automatically according to which edge of the sub-circuit you have placed it on.

Having placed a port or ports, you must annotate them. Hierarchical design works by connecting the ports on the parent object with like named logical terminals on the child sheet. It follows that both the ports and the terminals must be given unique names. This can be achieved in a variety of ways:


- Edit the terminal label using any of the methods described in *Editing An Object* on page 32.
- Use the *Property Assignment Tool* to assign the **NET** property of one or more ports. This is especially effective if a group of ports have names which run in an alphanumeric sequence.

It is quite legal to connect buses to ports. In this case, the name for the port should generally define the range for the bus, as in  $D[0..7]$ , although this is not mandatory. If no range is given, ISIS will use the range given for the bus section that connects to the port, or if that has no name or range then the width will be taken from whatever bus pins connect to the bus.

*Placing a port without annotating it will be reported as a netlist error since such an object has no meaning and cannot be tied up with anything on the child sheet.*

## Editing Sub-Circuits

A sub-circuit may be resized using the general procedure described in *Resizing An Object* on page 31 and edited using any of the general editing techniques described in *Editing An Object* on page 32.

 See *Parameterized Circuits* on page 51 for further information about parameterized circuits.

## Sub-Circuit Properties

Sub-circuits have the following system properties:


Property Name	Description
NAME	The sub-circuit instance name. This is also used as the name for the child sheet.
CIRCUIT	The name of the child circuit. If you do not assign this, ISIS will choose an automatic name when you first enter the child sheet.

If any other property names are assigned, then these will create user properties in the sub-circuit's text block. Such properties then become sheet properties for the child sheet and may be used in property expressions.

## TERMINALS

Terminals are used for specifying the interface to a circuit - ISIS does not allow a wire to 'float' - both ends must connect to something so all the inputs and outputs in your design will be indicated by terminals.

There are two types of terminal - *Logical Terminals* and *Physical Terminals*. The two are distinguished purely by the syntax of their labels.

 You can customise the appearance of terminals by editing the *TERMINAL* graphics style. See *Editing Global Styles* on page 42 for more information.


### Logical Terminals


A logical terminal serves merely to donate a net name to the wire to which it connects. Groups of wires with one or more net names in common are taken to be connected by the netlist generator. Logical Terminals thus provide a means to connect things together without using

wires. In particular they provide the means make connections between the sheets in a multi-sheet design.

As with Wire Labels and Bus Entries, the net name can contain any alphanumeric characters plus the hyphen ('-') and underscore ('\_'). Spaces can be used within PROTEUS but may cause problems for other software.

Logical terminals may also connect to buses. This provides an extremely efficient way to run buses up and down a hierarchical design.

 See *Net Names* on page 123 for discussion of the role of logical terminals in netlisting

 See *Hierarchical Designs* on page 117 for discussion of the role of logical terminals in hierarchical designs.

## **Physical Terminals**

A physical terminal represents a pin on a physical connector. For example, a terminal with the name:

J3:2

is taken to be pin 2 of connector J3. Whilst it is perfectly possible to deal with connectors in exactly the same way as all other components (i.e. define a device to represent them), using Physical Terminals has the advantage that the pins can be placed wherever it is most convenient.

For PCB design, where it is necessary to specify the package type for the connector, a **FIELD** property assignment block (see *Part Property Assignments (\*Field)* on page 71) must be used as there is no actual component to edit.

*Note that a bus terminal may not be physical, as there is no means to specify the pin numbering for the individual pins.*

## **Placing Terminals**

ISIS supports an unlimited variety of terminal symbols. However, when you first select the *Terminal* icon, a basic set of 7 types are automatically pre-loaded into the *Object Selector*.

### **To place a terminal:**

1. If the terminal type you want is not listed in the *Object Selector*, first pick it from the symbol library.
2. Highlight the terminal name in the *Object Selector*. ISIS will show a preview of the terminal in the *Overview Window*.

3. Use the *Rotation* and *Mirror* icons to orient the terminal according to how you want to place it.
4. Point at the position in the *Editing Window* where you want the terminal to appear, and click left. If you hold the mouse button down, you can drag the terminal around until you have it exactly where you want it. When you release the mouse button, it will be placed onto the drawing.

Having placed a terminal, you must then annotate it since an unannotated terminal will be ignored by the netlist compiler. There are a variety of approaches:

- Edit the terminal label using any of the general methods described in *Editing An Object* on page 32.
- Use the *Property Assignment Tool* to assign the **NET** property of one or more terminals. This is especially effective if a group of terminals have names which run in an alphanumeric sequence.

It is quite legal to connect buses to terminals. In this case, the name for the terminal should generally define the range for the bus, as in  $D[0..7]$ , although this is not mandatory. If no range is given, ISIS will use the range given for the bus section that connects to the port, or if that has no name or range then the width will be taken from whatever bus pins connect to the bus.

*Placing a terminal without annotating it will be reported as a netlist error since such an object has no meaning, logically or physically.*

### **Editing Terminals**

A terminal may be edited using any of the general editing techniques (see *Editing An Object* on page 32). In addition, since terminals often appear in groups, the *Property Assignment Tool* can be put to good use for annotating and setting the electrical type of terminals. The *Edit Terminal* dialogue form has the following fields:

<b>Name</b>	The net name for a logical terminal or the pin name for a physical terminal.
<b>Type</b>	The electrical type of the terminal.

## Terminal Properties

Terminals have the following system properties:

Property Name	Description
NET	The terminal net label.
SYMBOL	The symbol used for the terminal. This can be one of the standard terminal symbols, or else the name of a user defined terminal symbol.
TYPE	The electrical type of the terminal. This can be any of PASSIVE, INPUT, OUTPUT, BIDIR or POWER.

## PIN OBJECTS

A full description of how to create and edit your own devices is given in *Library Facilities* on page 91. Here, we just discuss the placement and editing of pin objects. These are used to represent each drawn pin of a device element and consist of some graphics (often just a single line) plus the capacity to carry and display a pin name and a pin number.

*Please note that you cannot wire off a pin object - you can only connect to pins that belong to fully constituted components that have been placed in the usual way.*

### Placing Pin Objects

An unlimited number of pin object types may be defined, and a good selection are provided in SYSTEM.LIB. However, when first select the *Device Pin* icon, a basic set of 6 types are automatically pre-loaded into the *Object Selector* and these will suffice for most purposes.

#### To place a pin:

1. If the pin type you want is not listed in the *Object Selector*, first pick it from the symbol library.
2. Highlight the pin type name in the *Object Selector*. ISIS will show a preview of the pin in the *Overview Window*.
3. Use the *Rotation* and *Mirror* icons to orient the pin according to how you want to place it.
4. Point at the position in the *Editing Window* where you want the pin to appear, and click left. If you hold the mouse button down, you can drag the pin around until you have it exactly where you want it. When you release the mouse button, it will be placed onto the drawing.

Having placed a pin, you will generally want to modify it to give it a pin name, number and electrical type. There are a variety of possibilities:

- Edit the pin manually using any of the general methods described in *Editing An Object* on page 32.
- Use the *Property Assignment Tool* to assign the **NAME**, **NUMBER** and **TYPE** properties of one or more pins. This is especially effective if a group of pins have names which run in an alphanumeric sequence such as buses.

If a pin represents a data or address bus, you may want to use a bus pin. In this case, pin numbers can only be specified using the *Visual Packaging Tool*. Equally, if the device has multiple elements, (e.g. a 7400) you must again specify pin numbers for each element using the packaging tool. In either of these cases, you should leave the pin numbers of the pins blank.

### **Editing Pin Objects**

A pin may be edited using any of the general editing techniques (see *Editing An Object* on page 32). In addition, since pins often appear in groups, the *Property Assignment Tool* can be put to good use for defining pin names, numbers and electrical types.

### **Pin Object Properties**

Pin objects have the following system properties:

<b>Property Name</b>	<b>Description</b>
NAME	The pin name.
NUM	The pin number.
SYMBOL	The symbol used for the pin. This can be one of the standard pin symbols, or else the name of a user defined pin symbol.
TYPE	The electrical type of the terminal. This can be any of PASSIVE, INPUT, OUTPUT, BIDIR, TRISTATE, PULLUP, PULLDOWN or POWER.



---

## SIMULATOR GADGETS

The interface to the ProSPICE simulator makes use of certain special objects within ISIS. The complete set consists of:

GENERATORS                      TAPES  
VOLTAGE PROBES    CURRENT PROBES  
GRAPHS

Instructions pertaining to the use of these objects are given in the Proteus VSM manual.

## 2D GRAPHICS

ISIS supports the following types of 2D graphics objects: lines, boxes, circles, arcs, scalable text and composite symbols. These are intended for use both directly on the drawing, for example to draw division lines and sectional boxes around parts of a design, and also for creating new library parts (devices, symbols, pins and terminals).

### Placing 2D Graphics

The following are the procedures for placing the various types of graphic objects.

#### To place a line:

1. Select the *Line* icon from the *Mode Selector* toolbar.
2. Select the *Graphics Style* you want the line drawn in from the *Object Selector*.
3. Click left to mark the start of the start of the line.
4. Click left again to mark the end of the line.

#### To place a box:

1. Select the *Box* icon from the *Mode Selector* toolbar. .
2. Select the *Graphics Style* you want the box drawn in from the *Object Selector*.
3. Point where you want the top left corner of the box and press the left mouse button.
4. Drag the mouse to where you want the bottom right corner of the box and release the button.

#### To place a circle:

1. Select the *Circle* icon from the *Mode Selector* toolbar.
2. Select the *Graphics Style* you want the circle drawn in from the *Object Selector*.

3. Point where you want the centre of the circle and press the left mouse button.
4. Drag the mouse to a point on the circumference of the desired circle and release the button.

### To place an arc:

1. Select the *Arc* icon from the *Mode Selector* toolbar.
2. Select the *Graphics Style* you want the line drawn in from the *Object Selector*.
3. Consider the arc as lying in one quadrant of an ellipse - you will first define this quadrant. Point at the position where one end of the quadrant will lie and press the left mouse button.
4. Drag the mouse roughly along the path of the quadrant and release the button when you reach the other end.
5. A pair of 'clipping lines' will now appear, allowing you to define which section of the quadrant you wish the arc to be drawn. Move the mouse around until you have just the desired section visible, and then click left.

### To place a path:

1. Select the *Path* icon from the *Mode Selector* toolbar.
2. Select the *Graphics Style* you want the path drawn in from the *Object Selector*.
3. Position the mouse over the *Editing Window* where you wish the first vertex of the path to be and click left to place it.
4. To enter a straight-line segment, simply move the mouse; to enter a curved segment, press and hold down the CTRL key and then move the mouse.

As the mouse is moved a rubber-banded line is displayed showing the type of segment (straight or curved) that will be created and its position.


5. Click left to place the second vertex. During placement, a placed vertex can not be deleted or 'undone' though the path may be edited after placement and unwanted vertices removed or segments altered.
6. Repeat steps four and five to complete your path or press ESC to cancel the path entry.

The path is not completed until you place a final vertex at the same point as the first vertex, so closing the path.

### To place graphics text:

1. Select the *Text* icon from the *Mode Selector* toolbar.
2. Select the *Graphics Style* you want the text drawn in from the *Object Selector*.
3. Use the *Rotation* and *Mirror* icons to orient the text according to how you want it to appear on the drawing.
4. Point at the position in the *Editing Window* where you want the bottom-left of the text to appear and click left.

The *Edit 2D Graphics Text* dialogue form is displayed.

 See *Editing 2d Graphics* on page 88 for a full reference this dialogue form.

5. Type the text into the dialogue form and set the justification, text size, etc. if required.
6. Press ENTER or click on the OK button to place the text, press ESC or click on the CANCEL button to abort placing the text.

### To place a symbol:

1. Select the *Symbol* icon from the *Mode Selector* toolbar.
2. Select the symbol you wish to place from the *Object Selector*. If the symbol you want is not in the selector, you must first pick from the symbol library. The *Symbol Library Pick* form can be displayed by clicking on the 'P' toggle on the selector.
3. Use the *Rotation* and *Mirror* icons to orient the symbol according to how you want it to appear on the drawing.
4. Point at the position in the *Editing Window* where you want the symbol to appear, and click left. If you hold the mouse button down, you can drag the symbol around until you have it exactly where you want it. When you release the mouse button, it will be placed onto the drawing.

### Resizing 2D Graphics

Lines, boxes, circles, arcs and paths may be resized by tagging them with the right mouse button and dragging one or more of the displayed 'handles' as follows:

- Lines have two handles which adjust the start and end points.
- Boxes have eight handles which adjust the corners and edges.
- Circles have four handles, all of which adjust the radius.
- Arcs have two handles which adjust the endpoints and the two Bezier control points.

- Paths have one handle per vertex (the point between two segments) plus two Bezier control points for each curved segment.

Paths support additional editing operations that allow the path to be modified without having to be deleted and re-entered. All these operations require you to hold the ALT (think of ALT=ALTer) key down:

- Click right on a vertex handle to delete that handle. The segments either side of the handle are also removed and a straight line replaces them.
- Click left on a line or curve segment to break it in to two straight lines with a common vertex at the point clicked.
- Hold the CTRL key down and click left on a line or curve segment to break the segment in to two Bezier segments with a shared vertex at the point clicked.


Don't forget that for all three of the above path editing operations, the ALT key must be held down!

The other types of graphic objects cannot be resized - instead you should delete and replace them to effect modifications.

### ***Editing 2D Graphics***

All 2D graphics objects can be edited in the usual way by first tagging it with the right mouse button and then clicking left on it (without moving the mouse).

All 2D graphic objects except 2D graphics text (discussed below) displays an *Edit Graphics Style* dialogue box that allows you to specify local, fixed, values.

 See *The Header Block* on page 38 for details of how to use 2D graphics text to display design information.

## **MARKERS**

### ***Marker Types***

Markers are used in the creation and editing of devices, symbols, terminals and pins. The following fixed set of types is provided:

<b>TYPE</b>	<b>PURPOSE</b>
ORIGIN	Defines the anchor point of any library part. The anchor point is the point around which the object may be rotated and corresponds to the mouse position at the time of placement.

---

NODE	Defines the position of the wire connection point for a pin or terminal.
BUSNODE	As above, but defines the pin or terminal as being of bus type. A bus line (thick, blue) will be drawn from the busnode to the origin.
LABEL	Defines the position and orientation of the label for a terminal.
PINNAME	Defines the position and orientation of the pin name for a pin.
PINNUM	Defines the position and orientation of the pin number for a pin.
INCREMENT	Used in creating Active Component simulator models, this marker defines a hot-spot for incrementing the state variable.
DECREMENT	Used in creating Active Component simulator models, this marker defines a hot-spot for decrementing the state variable.

## ***Placing Markers***

Markers are placed in the same way as graphics symbols (which is really what they are!).

### **To place a marker:**

1. Select the *Marker* icon from the *Mode Selector* toolbar.
2. Select the marker you wish to place from the *Object Selector*.
3. Use the *Rotation* and *Mirror* icons to orient the marker according to how you want it to appear on the drawing. This is only appropriate for *Label*, *Pinname* and *Pinnum* markers - orientation is irrelevant for the other types.
4. Point at the position in the *Editing Window* where you want the marker to appear, and click left. If you hold the mouse button down, you can drag the symbol around until you have it exactly where you want it. When you release the mouse button, it will be placed onto the drawing.



# LIBRARY FACILITIES

## GENERAL POINTS ABOUT LIBRARIES

As supplied there are two symbol libraries and over 25 device libraries. For an up to date listing of all the supplied libraries and library parts, read the file LIBRARY.PDF in the library directory of your Proteus installation. You will need to install the Adobe Acrobat reader if you have not already done so.

### ***Library Discipline***

USERSYM.LIB and USERDVC.LIB are set to read/write; the rest are set to read-only. The idea is that you should only add things to USERSYM.LIB (new symbols) and USERDVC.LIB (new devices). This means that we can supply you with updates to the parts we have defined without risk of overwriting similarly named objects in your own libraries.

You can, of course, create further libraries of your own using the *Library Manager*.

Should you really need to change things in the read-only libraries, you can set them to read/write in *File Manager* or *Explorer* under Windows or using the *Library Manager* from within Proteus.

Under no circumstances should you remove things from SYSTEM.LIB.

### ***The Pick Command***


The *Pick* command serves as an alternative to the library browser, primarily for when you know the name of the device or symbol you are looking for. You can search for an exact match, or else use the various pattern matching options to search for a part when you have a rough idea of its name.

If you pick a device or symbol that is already loaded into the layout, ISIS will update it from the libraries on disk.

Note also:

- Where devices are concerned, the replacement algorithm will match pin positions or pin names, so connectivity will be maintained even if you move or renumber a devices pins.
- Where a there are two or more devices or symbols with the same name spread across several libraries, the *Pick* command will load the newest one. This is particularly helpful

if you have changed one of our parts and put in USERDVC.LIB, since your version will be deemed newer than ours.

 See the section *Picking, Placing And Wiring Up Components* in the tutorial on page 7 for details of how to pick parts using the library pick forms.

## SYMBOL LIBRARIES

The symbol libraries are used to hold both general graphical symbols for direct placement onto drawings, and also symbols for terminals, module ports and device pins.

The various types of symbol are created with different name prefixes, and thus appear to all intents and purposes to be stored in separate 'compartments' of the symbol library. A set of these objects is defined in SYSTEM.LIB and is pre-loaded into the various object selectors when you start ISIS. Thus, when you select the *Terminal* icon, and you see the names DEFAULT, INPUT, OUTPUT etc. you are actually accessing symbols called \$TERDEFAULT, \$TERINPUT, \$TEROUTPUT and so on. When you place a terminal on the drawing, a terminal object is created and the appropriate symbol is assigned to it.

The significance of all this comes down to two points:

- Graphics symbols, terminals, module ports and device pins are all stored in symbol libraries (e.g. SYSTEM.LIB and the user library USERSYM.LIB). There are no special libraries for terminals, ports or pins.
- The procedures for making the various symbol types are all very similar.

### Graphics Symbols


A symbol is a group of 2D graphics objects which are treated as a single object. For instance, using three lines and two arcs you can form an AND gate symbol.



### To make a graphics symbol

1. Select one of the 2D graphics icons - e.g. *Line*, *Box* etc.
2. In the *Object Selector* select a graphics style appropriate to the type of symbol you are creating. For a symbol that will form the basis of future components, this is generally the *COMPONENT* style though for some symbols, such as OP AMP, where a small line is needed to link the body of the symbol to the base of a pin this would be drawn in the *PIN* style.



-  If the outline of the symbol consists of lines and arcs and you want the symbol to be filled, use the *path* object to create the outline.
- 3. Select and place graphic objects as required to form the symbol.  
Any graphic that needs to have a fixed appearance should be edited, the appropriate *Follow Global?* checkboxes unchecked and the graphics style attribute changed.
- 4. If you want to define the origin for the symbol, select the *Markers* icon, click on the *Origin* marker in the object selector and place it where you want the origin to be. If you don't place an *Origin*, ISIS will default the origin to the centre of the symbol.
- 5. Tag the objects that will comprise the symbol by dragging a tag-box around them with the right mouse button.
- 6. Invoke the *Make Symbol* command from the *Library* menu, select a name and library for the new symbol.
- 7. Click O.K. to complete the operation.

A symbol can only consist of 2D graphics objects - you cannot tag a whole section of circuitry including components, wires etc. and make that into a symbol. To manipulate circuit sections in this way, you should use the *Import* and *Export* section commands on the *File* menu.

## ***User Defined Terminals***

ISIS permits the definition of user defined symbols for use as logical or physical terminals. These are made in the same way as ordinary symbols except that you must place a *Node* marker to specify the position of the terminal's connection point, and a *Label* marker to specify the position and orientation of its net label.

### **To make a user defined terminal**

1. Select an appropriate 2D graphics icon - typically the *Line* icon - from the *Mode Selector* toolbar.
2. In the *Object Selector* select an appropriate graphics style. This will nearly always be the *TERMINAL* style though the *BUS WIRE* style may also be appropriate for small parts of the symbol.
3. Select and place graphics objects as required to form the body of the terminal.

Any graphic that needs to have a fixed appearance should be edited, the appropriate *Follow Global?* checkboxes unchecked and the graphics style attribute changed.

4. Select the *Markers* icon. Place a *Node* or *Busnode* marker where you want the wire or bus to connect the terminal, and a *Label* marker where you want its net label to be. You can also place an *Origin* marker to define where its origin will be.
5. Tag the objects that will comprise the terminal by dragging a tag-box around them with the right mouse button.
6. Invoke the *Make Symbol* command from the *Library* menu, set the type to *Terminal* and then select a name and library for the new terminal.
7. Click OK to complete the operation.

Note that the electrical type of user defined terminals will always default to passive. If you need user defined terminals to carry different electrical types you must assign them after placement with the *Property Assignment Tool*.

### ***User Defined Module Ports***

Module ports are the connectors used to attach wires to sub-circuits and it is possible to create user defined symbols for them. These are made in the same way as ordinary symbols except that you must place a *Node* marker to specify the position of the port's connection point, and a *Label* marker to specify the position and orientation of its label.

#### **To make a user defined module port**

1. Select an appropriate 2D graphics icon - typically the *Line* icon - from the *Mode Selector* toolbar.
2. In the *Object Selector* select an appropriate graphics style. This will nearly always be the *PORT* style though the *BUS WIRE* style may also be appropriate for small parts of the symbol.
3. Select and place graphics objects as required to form the pin symbol.

Any graphic that needs to have a fixed appearance should be edited, the appropriate *Follow Global?* checkboxes unchecked and the graphics style attribute changed.

In drawing the port, you should orient it in a manner suitable to be placed on the left hand edge of a sub-circuit - ISIS will mirror it in X when it is placed on the right hand edge.

4. Select the *Markers* icon. Place a *Node* or *Busnode* marker where you want the wire or bus to connect the port, and a *Label* marker where you want its net label to be. You should also place an *Origin* marker to correspond with where the edge of the sub-circuit will be.
5. Tag the objects that will comprise the port by dragging a tag-box around them with the right mouse button.

6. Invoke the *Make Symbol* command from the *Library* menu, set the type to *Module Port* and then select a name and library for the new terminal.
7. Click OK to complete the operation.

## ***User Defined Device Pins***

Device pins are, in fact, drawn as symbols and consequently you can define your own symbols for them. A variety of device pin symbols are supplied in SYSTEM.LIB. Nevertheless, there may be some situations in which it is appropriate to define your own.

### **To make a user defined device pin**

1. Select an appropriate 2D graphics icon - typically the *Line* icon - from the *Mode Selector* toolbar.
2. In the *Object Selector* select a graphics style. This will nearly always be the *PIN* style for a standard pin or the *BUS WIRE* style for a bus pin.
3. Select and place graphics objects as required to form the body of the terminal. Any graphic that needs to have a fixed appearance should be edited, the appropriate *Follow Global?* checkboxes unchecked and the graphics style attribute changed.
4. Select the *Markers* icon. Place a *Node* or *Busnode* marker where you want the wire or bus to connect to the pin, a *Pinname* marker where you want the pin name to be, and a *Pinum* marker where you want the pin number to be. You can also place an *Origin* marker to define where its origin will be.
5. Tag the objects that will comprise the pin by dragging a tag-box around them with the right mouse button.
6. Invoke the *Make Symbol* command from the *Library* menu, set the type to *Device Pin* and then select a name and library for the new pin.
7. Click OK to complete the operation.

### ***Editing an Existing Symbol***

Any type of symbol may be edited by placing an instance of it and then using the *Decompose* command from the *Library* menu.

#### **To edit a symbol:**

1. Place an instance of the symbol. This will be a Graphics Symbol, Terminal, Module Port or Device Pin, as appropriate.
2. Tag the object by pointing at it and clicking right.
3. Select the *Decompose* command from the *Library* menu. This will break the symbol into graphics and markers.
4. Add, delete or edit the graphics and markers as required.
5. Reconstitute the symbol according to the procedure from the previous sections appropriate to its type.

### ***Hierarchical Symbol Definitions***

ISIS quite happily allows a symbol to contain other symbols and/or other graphic objects. This allows you to make, for instance, a NAND gate out of the previously defined AND gate plus a circle. Note that symbols defined from other symbols are not "linked" to them in any way - if the lower level symbols are changed or even deleted, this will not affect the new one. This also means that a symbol can be defined as a modification of itself without difficulty, although this will destroy the old version.

## ***DEVICE LIBRARIES***

A device (in ISIS terminology) is a type of real world component such as an NPN transistor or a PIC microprocessor. It follows that a component placed on the drawing is an instance of a device. There are essentially three types of device:

- Single element devices. These are parts for which there is a one to one correspondence between the schematic symbol and the PCB package. Each pin has a single name and a single pin number.
- Homogenous multi-element devices. These are parts for which there are several *identical* elements in the one PCB package. Typical examples would be a 7400 quad NAND gate or a TL072 dual op-amp. The same pin will have a different pin number for each element, except for the power pins which tend to be *common*.
- Heterogeneous multi-element devices. These are parts where there are several *different* elements in the one PCB package, but where you wish to draw each element as separate component on the schematic. The most common example by far is a relay, in which you

wish to have the coil and one or more sets of contacts on different parts of the schematic.

ISIS also provides support for *bus pins*. Devices such as microprocessors and their associated peripherals can thus be drawn in very compact forms since their data and address buses may be represented by single pins. Wiring them up becomes a lot less tedious, too.

Whereas a single element device will have just one physical pin number associated with each schematic pin, multi-element parts and parts with bus pins all have multiple pin numbers for each device pin. This aspect of the device creation process is handled by the *Visual Packaging Tool*. In addition, this tool will allow you to create alternate packagings (each with its own set of pin numbers) for the same schematic part. A typical application of this would a microprocessor chip such as the PIC16F877 which is available in both DIL40 and PLCC44 packages.

## ***Making a Device Element***

Most of the devices you will encounter will involve only a single element. That is to say that by placing one component object, you account for all the pins in the physical part. This is in contrast to a multi-element part like a 7400 for which you need to place four gates to account for all the pins. Either way, the first stage in making a new library part is to place the graphics and pins for the device element or elements.

### **To make a device element:**

1. Place graphics objects to define the device body in the appropriate graphics style(s).
2. Place device pin objects to represent the pins.
3. Annotate the pins to assign name and types using any of the standard editing techniques (see *Editing An Object* on page 32) or the *Property Assignment Tool*.
4. Tag all the objects that comprise the element. Then invoke the *Make Device* command and assign any default properties.


These stages are worthy of further discussion:

### **Defining the Device Body**

The device body is essentially the complete graphic for the device, excluding its reference designator and pins. Very often, it will just be a box, in which case you should simply select and place a graphics box of the appropriate size in the *COMPONENT* graphics style. For components with more complex graphics such as transistors, op-amps and so forth you can use any of the graphic objects that ISIS provides in whatever graphics styles are appropriate and perhaps editing the objects and assigning local, fixed values to some or all of the graphics style attributes.

It is important to think carefully about what graphics styles you choose for the graphics in the new component. In general most graphics will be placed with the *COMPONENT* style selected and the graphic objects will not need editing as they default to fully following this parent *COMPONENT* style. Occasionally, however, you will want part of the graphics of the new device to be 'fixed'. For example, you may want the solid body of a transistor to always be filled, perhaps to black. In these cases it is appropriate to edit the graphic object after placement and uncheck some or all of the graphics style attributes and to set local values though where possible you should endeavour to leave as many attributes as possible following the parent, *COMPONENT*, style. Your other consideration should be how the graphic object placed will fit in to a schematic. For example, if you are designing an OPAMP you may wish to use short lines between the slope of the body and the base of power, compensation or some such pins in which case it is best if these lines are placed in the *PIN* style so that they look like part of the pin.


If you wish to define the origin for the device, you should also place an *Origin* marker at the appropriate point. If you don't specify an origin, ISIS will default it to the top leftmost pin end.

 See *2d Graphics* on page 85 for more information about placing and editing graphics objects.

### Placing the Pins

When making a device, you use the special device pin objects, available with the *Device Pin* icon selected to place each pin in turn. Several types of device pin are pre-loaded into the *Object Selector* when you first select the *Device Pin* icon, and further types can be picked from the symbol libraries as required. You may also define pin types of your own.

As you place each pin object, you need to be sure that it is oriented correctly. The blue cross that appears at one end of each pin object designates the connection point for the pin; the other should generally be in contact with some part of the device body.

 For further information about pin objects see page 83.

### Annotating the Pins

This third phase of the device creation process is probably the trickiest, and can lead to some obscure problems much later on in the design cycle if not carried out correctly. *You have been warned!*

Each pin can carry a pin name, pin number and electrical type. The latter is used for electrical rules checking, and also by the ProSPICE simulator. The electrical type of pins must be correctly specified for digital simulator models in particular.

You have two basic approaches available to you in annotating the pins:

- Edit each pin in turn (point at it and press CTRL+'E' is probably easiest), and proceed to edit its properties using the dialogue form.
- Use the *Property Assignment Tool* to assign to the **PINNAME**, **PINNUM** and **TYPE** properties of the pins.

In most cases, you will find it appropriate to use a mix of these techniques.

In assigning pin names and numbers, bear in mind the following:

- A pin must always have a name. If you type in a number when there is no name, the pin name will automatically be made the same as the pin number.
- If you give two or more pins the same name, they will be deemed to be electrically interconnected in the netlist and thus on the PCB layout.
- To place pins with overbars in their names, , use dollar ('\$') characters to mark the start and end of the overbar. For example RD/\$WR\$ would display as RD/WR .
- In general, it is easier to assign pin numbers using the *Visual Packaging Tool*, and this is the only way to assign pin numbers for a multi-element part or a part with bus pins. However, for a simple single element device, you can enter pin numbers at this stage, if you wish.

In assigning pin types, the following table may be helpful:

Pin Type	TYPE ID	Example Uses
Passive	PS	Passive device terminals
Input	IP	Analogue or digital device inputs
Output	OP	Analogue or digital device outputs
Bidir	IO	Microprocessor or RAM data bus pins
Tri-state	TS	ROM output pins
Pull Down	PD	Open collector/drain outputs
Pull Up	PU	Open emitter/source outputs
Power	PP	Power/Ground supply pins

If you are unclear how to actually perform the required editing operations, the following sections of the manual are also relevant:

- 📖 See *Pin Objects* on page 83 for a full discussion of the *Edit Pin* dialogue form.
- 📖 See *The Property Assignment Tool* on page 55 for examples of how to use the *Property Assignment Tool*.

### Invoking the Make Device Command

The final stage of creating a single element device is to tag all the objects (graphics and pins) which comprise the device, and then invoke the *Make Device* command from the *Library* menu. The use of the this command is discussed in detail in the following sections.

### ***The Make Device Command***

The *Make Device* command is a multi-stage dialogue form sometimes referred to as a *Wizard* in other applications. There are four pages:

- Device properties - these are things like the name of the device, the prefix to use for new components and also properties associated with component animation as used by *Proteus VSM*
- Packaging - this page displays the packagings that have been defined (for an existing component) and provides access to the *Visual Packaging Tool*.
- Component Properties - this page provides the means to create and edit property definitions and default property values.
- Library selection - the final screen allows you to choose the library in which the device will be stored.

Detailed context sensitive help is provided for all the fields on these pages. If you are unsure of the purpose of a particular field, click the '?' at the top right of the dialogue form and then click the field itself to see an explanation of its purpose. Given this, we shall restrict the discussion here to general information about each page.



## Device Properties Page

This page has two major sections:- *General Properties* and *Active Component Properties*.

The screenshot shows a dialog box titled "Make Device" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog is divided into two main sections:

- General Properties:** This section contains three text input fields. The first is labeled "Device Name:" and contains the text "U4741". The second is labeled "Reference Prefix:" and contains the text "U". The third is labeled "External Module:" and is currently empty. A small instruction above the fields reads: "Enter the name for the device and the component reference prefix." and "Enter the name of any external module file that you want attached to the device when it is placed."
- Active Component Properties:** This section contains four input elements. The first is a text input field labeled "Symbol Name Stem:". The second is a text input field labeled "No. of States:" containing the value "0". The third and fourth are checkboxes labeled "Bitwise States?" and "Link to DLL?", both of which are currently unchecked. A small instruction above these elements reads: "Enter properties for component animation. Please refer to the Proteus VSM SDK for more information."

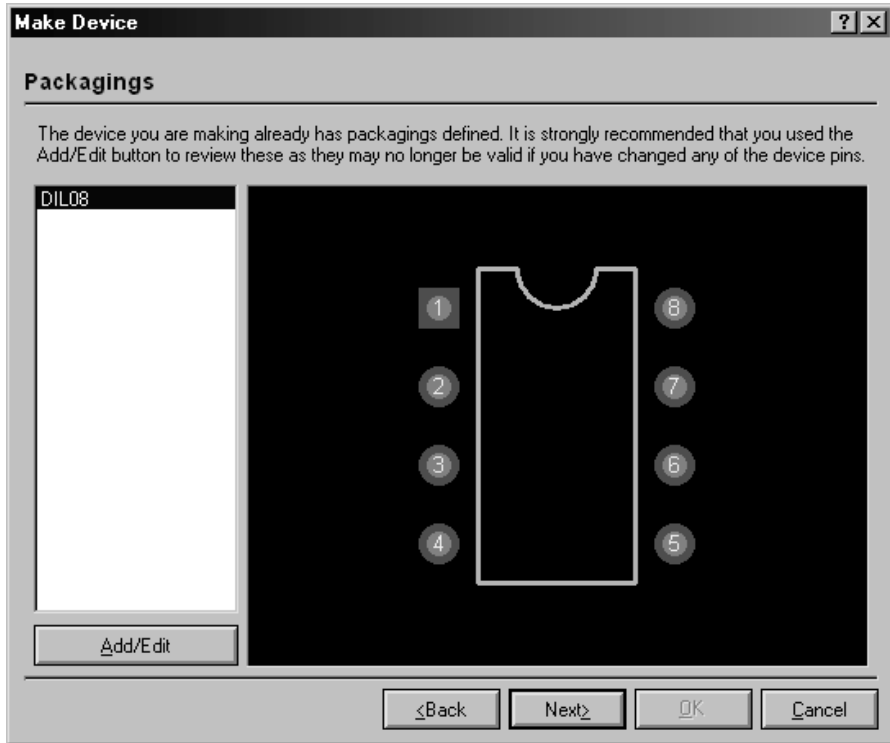
At the bottom of the dialog, there are four buttons: "<Back", "Next>", "OK", and "Cancel".

The *General Properties* section determines the name of the device, and the reference prefix. This is the letter or letters that appear in front of the part ID for newly placed components. Note that if you make this blank, newly placed components will be un-annotated and that their part value and properties text will also be hidden. This is useful for block diagram type drawings, or for components such as the *Virtual Oscilloscope*, which are not really part of the design.

The *Active Component Properties* section is used for creating animated components for use with Proteus VSM. See the Proteus VSM SDK documentation for more information.

## Packagings Page

This page displays the set of packagings that have been defined for the device; for a new device the list will be empty.

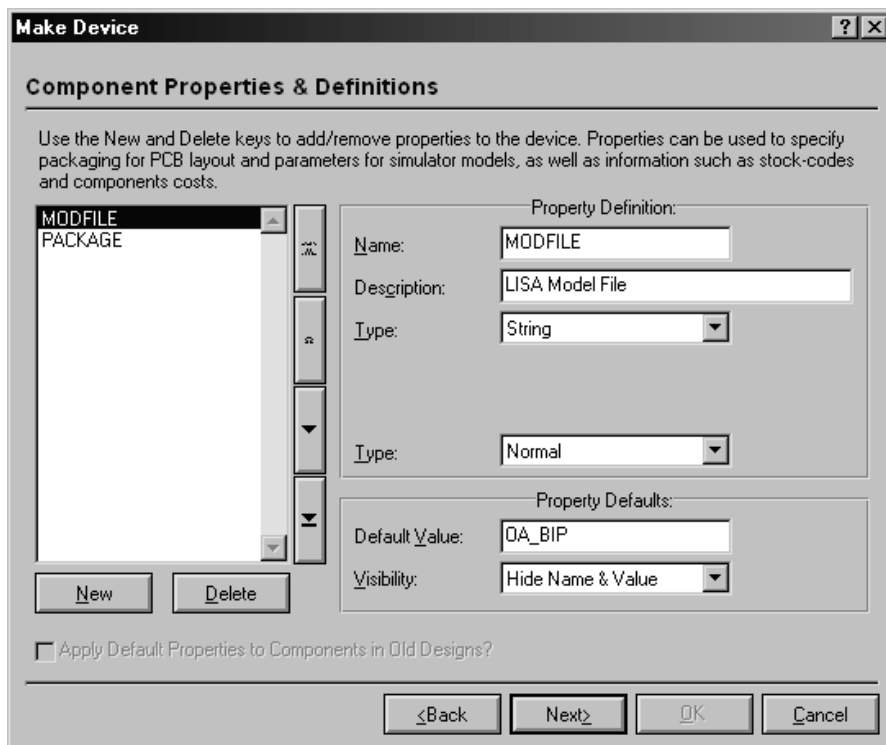


Pressing the *Add/Edit* button will launch the *Visual Packaging Tool* which is described in more detail on page 105.

Note that a different procedure is required to package a heterogeneous multi-element part. In this case, the packaging tool must be invoked for the complete set of elements as placed on the schematic. See page 110 for more information.

### ***Component Properties & Definitions Page***

This page is used to define property definitions and default values for the component's properties. The selector on the left shows the properties that have been defined, whilst the *Property Definition* and *Property Defaults* sections determine the type, visibility and default value of the property.



See *Property Definitions* on page 60 for more information about component properties and property definitions

### **Data Sheet and Help Page**

This page allows you to associate a data sheet (PDF file) and/or a help topic with the device. If a data sheet is defined, a *Data* button will appear on the *Edit Component* dialogue form and if a help topic is defined, a *Help* button will appear. You will see these buttons for many of the components in the supplied libraries.

**Make Device** ? X

**Device Data Sheet & Help File**

You can link your device to a data sheet (Acrobat .PDF file) and/or a help file. These can then be accessed via special buttons on the 'Edit Component' dialogue form.

**Data Sheet:**

Data Sheet Filename:

FTP Server:

FTP Path:

FTP User Id:

FTP Password:

CD Title:

CD Path:

**Help Topic:**

Help File:

Context Number:

<Back    Next>    OK    Cancel

Data sheets can be located in one of three places:

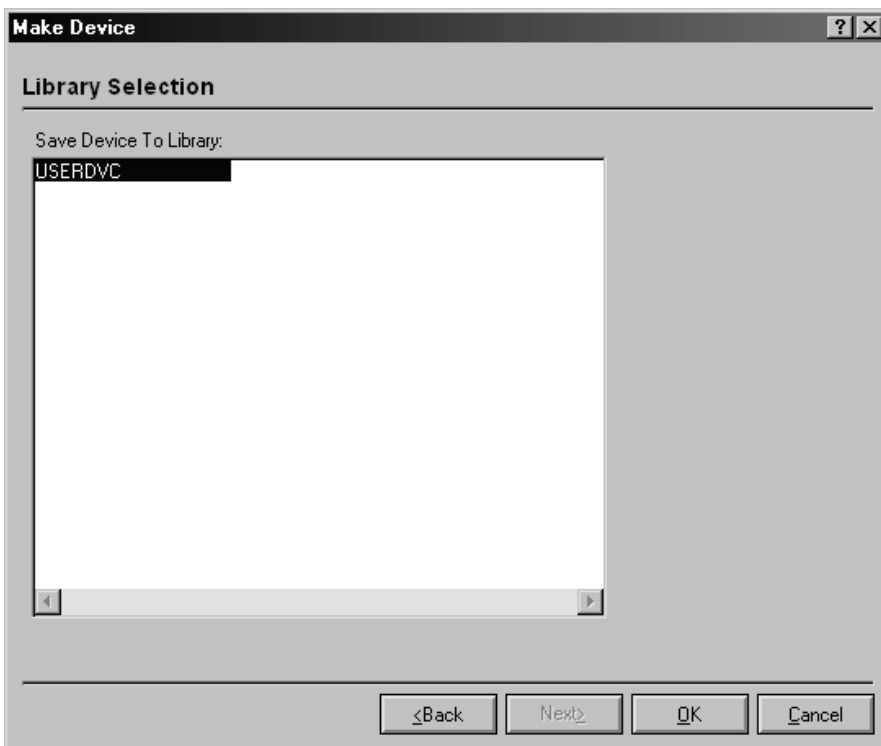
- In the *Data* directory of your Proteus installation. You can change the location that Proteus search for data sheets by clicking the *Path Settings* button on the *Data Sheet Not Found* dialogue form that appears when Proteus cannot find a data sheet.
- On an FTP server - either on the Internet or on your company Intranet. You may need to enter a user ID and password for some servers.
- On a CD or CD-R.

The general idea is that there should be a central repository for data sheets, and that Proteus copies them to individual users installations on as-needed basis.

Referencing a help topic is most useful if you are creating complex models for the ProSPICE simulator, and need to create documentation to be associated with the model. This is likely to be of relevance onto to advanced users and model developers.

### ***Library Selection Page***

The final page allows you to select into which library the new device will be stored. Only read/write (as opposed to read only) device libraries are displayed.

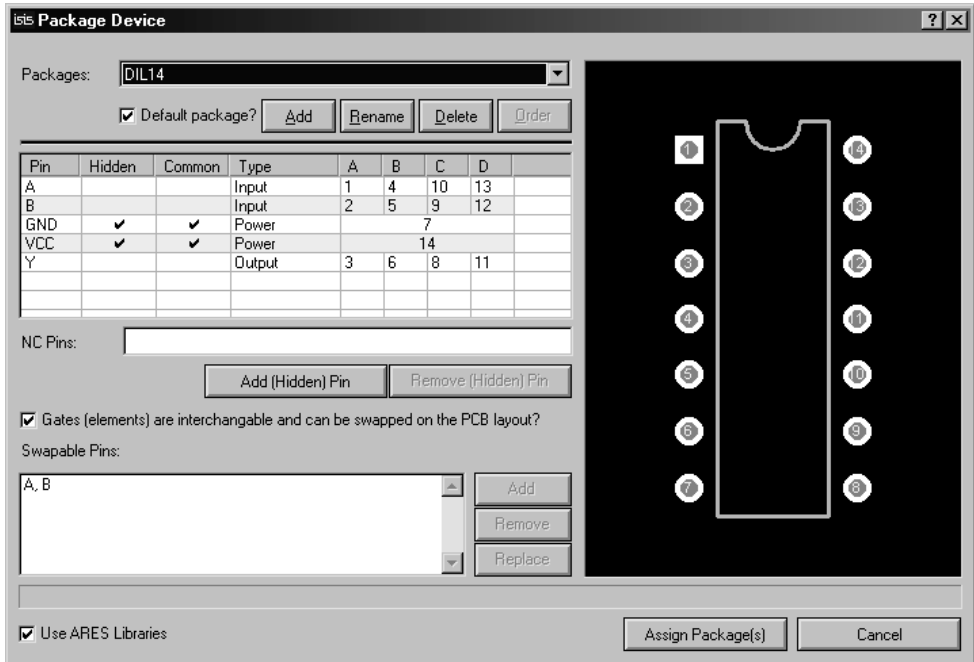


Once you click the OK button, the device will be stored to the selected library. Also, if it exists within the design you will be prompted as to whether you wish to update all the components that use it with the new definition.

### ***The Visual Packaging Tool***

The visual packaging tool provides a graphical environment in which to assign one or more PCB footprints to a schematic part. For each PCB package, a table mapping pin numbers to pin names is created such that different packagings can have different pin numbers for the same schematic pin.

The packaging tool also facilitates the entry of different pin numbers for each element in a multi-element part, and each bit of a bus pin.



The packaging tool can be invoked in one of two ways:

- By clicking the *Add/Edit* button of the *Packaging Page* in the *Make Device* dialogue form.
- By placing the element or elements of the device to be packaged on the schematic, tagged them and selecting the *Packaging Tool* command from the *Library* menu.

The second method is the only way to package heterogeneous multi-element parts.

As with the *Make Device* command, detailed context sensitive help is available on all the fields. To see the help on a particular field, click the '?' at the top right of the form, and then click the field itself.

### **Packaging Selector**

The *Packages* section of the packaging tool shows a list-selector for the packagings that have been defined so far and buttons for adding, renaming, deleting and ordering them. One package may be selected as the default, and this is the one that will be used for newly placed components.

Assuming that you have ARES installed, the *Add* button will launch a copy of the package library browser enabling you to choose a footprint for the new packaging.

## ***Pin Grid***

The main business of the packaging tool is carried out using the pin grid. Here you can enter pin numbers for each pin in each element, and also select which pins are common between the elements of a multi-element device.

The electrical type is displayed for information only, and cannot be changed here - you must define it when placing the individual pins around the device element.

## ***Number of Elements***

This field determines the number of pin-number columns that appear in the pin grid. For a multi-element heterogeneous part, it should represent the total number of elements.

## ***Package Viewer***

The package viewer displays the PCB footprint chosen for the current packaging. As each pin is assigned to the pin grid, it is highlighted such that you can see which pins remain unassigned. Also, if you point at a pin with the mouse, ISIS will display the associated name (if assigned) and the pin number.

If the text cursor is in the pin grid, then clicking on a pin in the package view will enter its number into the pin grid and the pin will highlight to show that it is now assigned.

## ***Hidden Pins***

Hidden pins are generally used to specify power supply connections to components in such a way that they do not clutter up the schematic. A hidden pin can be defined in one of two ways:

- By clearing the *Draw Body* checkbox in the *Edit Pin* dialogue form. This must be done when making the device element. Such a pin will appear in the *Pin Grid* when the packaging tool is first invoked, and will be present in all packagings.
- By clicking the *Add Hidden Pin* button. This creates a hidden pin that is specific to a given packaging, and is implicitly common to all its elements.

## ***Common Pins***


A common pin is one which has the same pin number on all the elements of a multi-element device. Typically this will apply to power pins and enable/strobe pins that are common to all buffers within a multi-element driver.

If a common pin is not hidden, then any wiring to it on one element will be connected wiring on the other elements.

A common pin is also deemed to be associated with all the elements in a heterogeneous multi-element part, even if it is not present on some of them. This has implications for the gate swap rules within ARES.

### **Gate Swap**


Selecting this checkbox will flag to ARES that the gate elements can be swapped.

 See page 151 for more information about Pin and Gate Swaps.

### **Swappable Pins**

The swappable pins section allows you to define groups of pins that are electrically interchangeable. For example, the two inputs A and B of a 7400 quad NAND gate are electrically identical and could be swapped on the PCB for convenience of wiring.

You can add a swap-group to the list by highlighting the pins in the pin grid (hold the ctrl key down and click left on the pin names), and then clicking the *Add* button.

 See page 151 for more information about Pin and Gate Swaps.

### **NC (Not Connected) Pins**

When packaging a large device, it may be useful to 'prove' that all the pins have been accounted for by verifying that all the pins are highlighted in the *Package View*. Since some pins may be specified as non-connected, it is useful to be able to record that they have been accounted for. This can be achieved by entering their numbers in the *NC Pins* field. Use commas to separate the pin numbers.

### **Making a Single Element Device**

Most of the devices you make will have a one-one correspondence between the schematic symbol and the PCB package. In these cases, the procedure is pretty much the same as that for creating a device element, as described on page 97.

#### **To make a single element device**

1. Place graphics objects to define the device body in the appropriate graphics style(s).
2. Place device pin objects to represent the pins.
3. Annotate the pins to assign name and types using any of the standard editing techniques (see *Editing An Object* on page 32) or the *Property Assignment Tool*.
4. Tag all the objects that comprise the device. Then invoke the *Make Device* command.
5. Enter a name and reference prefix for the device in the *Device Properties* page, then click *Next* to move to the *Packaging* page.

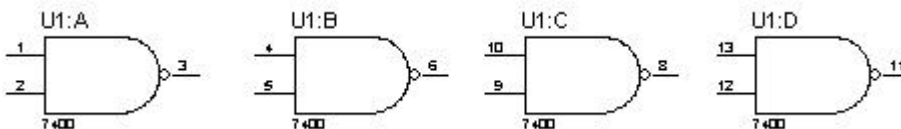


6. Assuming that the device has a PCB footprint, click the *Add/Edit* button and use the *Visual Packaging Tool* to assign a package and pin numbering. When you are done, click *Assign Packages* to return to the *Packaging* page.
7. Click *Next* to move to the *Component Properties* page and enter any property definitions and default values for other component properties.
8. Click *Next* to move to the *Data Sheet* page and specify the location of any data sheet or help topic that you wish to associate with the device.
9. Click *Next* to move to the *Library Selection* page and choose the library in which you wish to store the device.
10. Finally, click OK to complete the process.

If you prefer, you can also assign pin numbers for single element devices at step [3]. In this case, the pin numbers will appear automatically within the *Visual Packaging Tool*.

### **Making a Multi-Element Homogenous Device**

A multi-element homogenous device is one like a 7400 in which the physical part consists of several *identical* elements which you wish to place as separate components on the schematic. To handle such a device, ISIS must allow for different sets of pin numbers to be applied to the same element. For a 7400, there are 4 sets of pin numbers - one for each gate - and which set is used for a given gate is determined by the component reference suffix. For example, if you label a 7400 gate as U1:C then ISIS will use the third set of pin numbers: 8, 9 & 10. This is all handled by the *Visual Packaging Tool*.



### **To make a multi-element homogenous device**

1. Make the device body and place its pins as discussed on page 97.
2. Annotate the pins to assign name and types using any of the standard editing techniques (see *Editing An Object* on page 32) or the *Property Assignment Tool*.
3. Tag all the objects that comprise the device. Then invoke the *Make Device* command.
4. Enter a name and reference prefix for the device in the *Device Properties* page, then click *Next* to move to the *Packaging* page.
5. Click the *Add/Edit* button on the *Packaging* page to launch the *Visual Packaging Tool*.

- Specify the number of elements in the device and then fill out the pin number fields for each element in turn. Check the *common* column to indicate pins (such as power pins) that have the same number on all elements.
- Proceed as for a single element device from step 7.


### ***Making a Multi-Element Heterogeneous Device***

A multi-element heterogeneous device is defined as a part which consists of several *different* elements, each of which will be placed as a separate component on the drawing. The most common example is probably a relay where you have a coil and one of more sets of contacts. The requirement is to place the coil at one point in the drawing, and the set(s) of contacts elsewhere.

As with a multi-element homogenous device, the *Visual Packaging Tool* is used to deal with the assignment of pin numbers to each element.

#### **To make a multi-element heterogeneous device**

- Place graphics objects and device pins to define appearance of the elements. Make sure the elements are sufficiently separated in the *Editing Window* so that you can tag the objects that comprise each one by themselves.

 The process for creating a device element is described in detail on page 97.

- Annotate the pins to assign pin names and types only.
- Tag all the objects that comprise the first element. Then invoke the *Make Device* command from the *Tools* menu. Enter the *Device Name* with in the form

*NAME* : A

where *NAME* is whatever you want to call the part as a whole. The suffix : A tells ISIS that this is the first element of a heterogeneous device.

- Click *Next* twice to move to the *Component Properties* page and enter any property definitions and default values for other component properties. Note that you do not invoke the packaging tool at this stage for a heterogeneous part.
- Click *Next* to move to the *Data Sheet* page and specify the location of any data sheet or help topic that you wish to associate with the device.
- Click *Next* to move to the *Library Selection* page and choose the library in which you wish to store the device element and click OK to actually store it.
- Repeat the procedure from step [3] for the other elements, giving them names such as

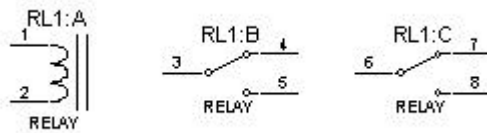
*NAME* : B , *NAME* : C

and so on.

8. Select the *Component* icon and place one instance of each element of the device in a free area of the schematic.
9. Drag a tag box around the placed elements, and then invoke the *Packaging Tool* from the *Library* menu. You should see that there are pin number columns for each element, but that pins which are unavailable for a particular element have '---' in the number column.
10. Create a packaging and enter the pin numbering in the usual way.
11. Click the *Assign Packages* button to store the packagings into the library parts. Note that the library part that you created for each element of the device will be updated.

**An Example**

Since this is a rather complex process, we will clarify it further with a simple example. Consider a relay coil and contacts as shown below.



Having made the device elements RELAY:A, RELAY:B and RELAY:C, you would invoke the *Visual Packaging Tool* and set up the *Pin Grid*, as follows:

Pin	Hidden	Common	Type	A	B	C
C1			Passive	1	---	---
C2			Passive	2	---	---
COM			Passive	---	3	6
NC			Passive	---	4	7
NO			Passive	---	5	8

**Making a Device with Bus Pins**

The ISIS device library system has the capability to support devices in which a single pin on the drawn component represents several pins on the physical part. Such a pin is called a *bus pin* and is intended to facilitate the efficient representation of microprocessors and their support chips. As with multi-element devices, the process of entering pin numbers for each bit of the bus is achieved with the *Visual Packaging Tool*.

### To make a device with bus pins

1. Make the device body and place its pins in the same way as for a single element device. You must use the *Bus* pin, or a user defined pin with a *Busnode* marker, for the bus pins.
2. Annotate the pins to assign pin names and types only. The bus pins must be given their full bus specification as in D[ 0 . . 7 ]. However, you can choose to hide the bus range - i.e. the [0..7] part - from the schematic by unselecting the *Draw Bus Range* checkbox.
3. Tag all the objects that comprise the device, invoke the *Make Device* command.
4. Enter a name and reference prefix for the device in the *Device Properties* page, then click *Next* to move to the *Packaging* page.
5. Assuming that the device has a PCB footprint, click the *Add/Edit* button and use the *Visual Packaging Tool* to assign a package and pin numbering. You will see that each bit of the bus is given its own row in the *Pin Grid*. Beyond this, the process for assigning pin numbers is exactly the same as for an ordinary part.

When you are done, click *Assign Packages* to return to the *Packaging* page

6. Proceed as from step 7 for making a single element device.

### **Property Definitions and Default Properties**

There are a variety of applications in which it is useful to arrange for library parts to carry default user properties which are automatically assigned to each component as it is placed. Some common examples include:

- **PACKAGE** properties for PCB design. Most of the library parts in the supplied libraries carry these properties pre-defined for through hole packaging.
- **PRIMITIVE**, **MODEL** and **MODFILE** properties for VSM simulation. Again, all our library parts for which there is simulator support carry these default properties.
- Stock and/or supplier order codes. If you add default properties for these to your libraries, then they can be included in the Bill of Materials report.

### To add a default properties to a device whilst making it:

1. Follow the steps in *Making A Device Element* on page 97.
2. When you get to the *Make Device* dialogue form, use the *Component Properties* page to assign property definitions and default values for the properties..
3. Store the device to a library in the ordinary way.

### To add to or edit the default properties of an existing device:

1. Pick, place and tag an instance of the device. There is no need to decompose it.

2. Invoke the *Make Device* command and click *Next* to move to the *Component Properties* page. Any existing property definitions will be displayed.
3. Edit the properties as required.
4. Click *Next* twice more and then store the device back to a library by clicking OK.

Default properties may also be assigned to library parts *en masse* by applying an ADI script from with *Library Manager*.

## ***Dealing with Power Pins***

The handling of power pins tends to be a somewhat confusing subject with different schematic capture packages doing different things. Further difficulty arises from the need to accurately define what happens with the power rails whilst not cluttering up the drawing with what is essentially trivial information.

ISIS provides you with a variety of approaches to handling power pins:

- Make the power pins visible and physically wire them to the appropriate points in the circuit. This has the advantage that you can see exactly what is connected where, but tends to be very inconvenient where there are lots of ICs on a schematic.
- Hide the power pins and let ISIS default their connections to like named nets. A hidden VCC pin will thus connect to the VCC net, and a hidden GND pin to the GND net.
- Hide the power pins and specify explicitly which nets they connect to using like named user properties. For example, the user property

VCC = +5V

applied to an object with a hidden VCC pin will cause that pin to be connected to the +5V net.

You can, of course, use a mix of these techniques as appropriate to different parts of a drawing. In general, we have found the use of hidden power pins to be almost essential for digital designs whereas in analogue work, it is often simplest just to wire up the power pins to the appropriate supply rails. This state of affairs is reflected in the construction of the supplied library parts.

### **To create a hidden pin on a simple, single element device:**

1. Place a pin object for the pin in the usual way.
2. Bring up the *Edit Pin* dialogue from by clicking right then left on the pin.
3. De-select the *Draw body* checkbox. This hides the pin body, name and number irrespective of the settings of the *Draw name* and *Draw number* checkboxes.

4. If the pin is to be a power pins, set the pin type to *Power*.

The existence of the pin object will then be marked only by a dark blue cross at its node end, and the pin will not be drawn at all on any device in which it is included. At netlist time, it will be automatically connected to a net with the same name as its pin name, unless explicitly overridden with an appropriately named user property.

### To create a hidden power pin within a packaging:

1. Create the device element(s) in the usual way, but do not place pin objects for the hidden pins.
2. When creating the packaging(s), use the *Add Pin* button to create an extra row within the *Pin Grid* for the hidden pin.
3. Type in a name for the hidden pin e.g. VCC, and then assign the pin a number

As with the other type of hidden pin, at netlist time ISIS will connect all hidden pins specified in the packaging to nets with the same name as the given pin names, unless you override this action with an appropriately named user property.

### To override a hidden pin's net:

1. Tag *all* the elements of the component for which the hidden pin exists. This is most easily achieved using the *Search and Tag* command with its *Begins With* option.
2. Use the *Property Assignment Tool* to assign a user property of the form.

*PINNAME=NET*

where *PINNAME* is the name of the hidden pin, and *NET* is the name of the net you want it connected to.

### ***Editing an Existing Device***

Any device element, whether a self contained, single element device, or a constituent of a multi-element may be broken into its constituent pins and graphics using the *Decompose* command from the *Library* menu.

### To edit device graphics or pins:

1. Place an instance of the device as a component.
2. Tag the object by pointing at it and clicking right.
3. Select the *Decompose* command from the *Library* menu. This will break the device into 2D graphics, pins and possibly an *Origin* marker. You will also get a text script which contains the device name, prefix, packaging and any default component properties.

4. Add, delete or edit the 2D graphics, pins and markers as required.
5. Reconstitute the device using the *Make Device* command as previously described. If you tag the text script generated by the *Decompose* command as well as the graphics before making the device then you can avoid having to re-enter the device properties.

If the library part is in use in the current drawing, you will get a prompt requesting whether to perform a *Design Global Update*. If you select this, then all components on the design that use the device will be updated with the new one using the same mechanism as described for the *Pick* command. Note that the properties of already placed components will not be modified as ISIS doesn't know which properties have been manually edited.

### To edit device properties:

1. Pick, place and tag an instance of the device. There is no need to decompose it.
2. Invoke the *Make Device* command and click *Next* to move to the *Component Properties* page. Any existing property definitions will be displayed.
3. Edit the properties as required.
4. Click *Next* twice more and then store the device back to a library by clicking OK.

As with device elements, you will be prompted as to whether you want to update components in the current design.

### To edit device packagings:

1. Pick, place and tag an instance of the device. There is no need to decompose it.
2. Invoke the *Packaging Tool* command from the *Library* menu.
3. Modify the packagings as required.
4. Click *Assign Package(s)* to store the changes back into the device library.

As with device elements, you will be prompted as to whether you want to update components in the current design.

Two other points are worth making:

- Remember that all design files carry with them their own copies of the library parts that they use. Consequently, changing a part in a library will not directly change any designs that make use of it. To effect such a change, you must load the drawing into ISIS and use the *Pick* command from the *Library* menu to effect a replace operation.
- When ISIS is installed, our own libraries are set to be read-only. This is to deter you from changing them - an action which would backfire on you when we issue updates. It

follows that if you wish to change any of the supplied library parts, you should store them to USERDVC.LIB, or some other library of your own.



# MULTI-SHEET DESIGNS

## MULTI-SHEET FLAT DESIGNS

### Introduction

With very large designs, or just to create some structure in smaller ones, it is common practice to draw the various sections of the design on separate sheets. Connections between the sheets are then indicated by means of common net names. For example, if two nets on different sheets are both labelled as MREQ, then they are assumed to be connected - see *Net Names* on page 123 for more information on net names.

### Design Menu Commands

ISIS supports multi-sheet designs and keeps all the sheets of a design in one file. Three commands on the *Design* menu give you all the facilities you need:

- *New Sheet* - creates a new root sheet and loads it.
- *Goto Sheet* - presents a menu of the sheets enabling you to move about the design. For a hierarchical design, the selection consists of the entire hierarchy tree and you can thus move instantly to any sheet in the design.
- *Remove Sheet* - removes and deletes the current sheet. You can only delete root sheets, and you cannot delete the last root sheet.

The titles presented by the *Goto Sheet* command are taken from the *Sheet Title* field of the *Edit Sheet Properties* dialogue form or else the *Sheet Name* if no sheet title has been given.

The sheet names assigned to sheets also determine their ordinal position in the design - i.e. the order in which they will print if you print out the lot. New root sheets start with the names ROOT10 , ROOT20 etc. and this gives you room to insert ROOT15 or whatever if you want a sheet in between.

## HIERARCHICAL DESIGNS

### Introduction

A hierarchical design is one which consists of two or more levels of sheets. The highest level is likely to be a block diagram showing the structure of the overall system and each block will have a sub-sheet with a section of the design on it. Depending on the complexity of the design, these sub-sheets may themselves contain further black boxes or *modules*; ISIS sets

no limit on the hierarchy depth although you are doing something very odd if you need more than half a dozen levels.

A second use for hierarchy concerns the replication of a part of a design - a simple example would be a stereo amplifier which has two mono channels and a common power supply. There is nothing to stop you simply drawing one channel, exporting it as a SEC file and then importing it to a second sheet. However, should you then wish to alter the mono circuit - even if it is only a cosmetic change - you are going to have to alter both channels. Where more than two copies of a circuit are involved this can mean serious hassle. With a hierarchical approach you have two modules labelled LEFT and RIGHT but each one is associated with the same circuit data. Naturally you still need different references for the same component in each instance of the mono amplifier and this is catered for by means of *Design Global Annotation*.

In ISIS, hierarchy also facilitates the creation and use of *Parameterized Circuits*, and is of considerable use when developing simulator models for VSM. The former subject is covered at length in *Parameterized Circuits* on page 51 whilst the latter is dealt with in the VSM manual.

### ***Terminology***

Before we delve any further into what is quite an abstract concept, we need to define some terminology...

#### Circuit

A circuit is a collection of components, other objects and the associated wiring in the general case. For example we can talk about the mono amplifier circuit.

#### Sheet

A sheet is an instance of a circuit and has a unique set of annotation data that gets mapped onto the components in the circuit. Where a sheet is attached to a module in the next level up - the *parent sheet* - we can call it a *sub-sheet* or *child sheet*. Therefore, we can say that the left and right channels of our amplifier are drawn on the left and right sub-sheets. The sheets at the top level of the design are called the *root sheets*.

#### Module

A module is an object which has an associated sub-sheet. There are two types of module: *sub-circuits* and *module-components*.

## Sheet Property

A sheet property is a property assignment that is attached to a particular sheet, and is available for use in property expressions for any of the objects on the sheet. In hierarchical design, any user properties of the parent module become sheet properties for the child sheet.

## Parameterized Circuit

A parameterized circuit is one in which one or more component values or other object properties is given as property expression involving one or more sheet properties. Since these sheet properties can be specified on the parent module (be it a sub-circuit or a module-component), it follows that the circuit itself can have different component or property values from one instance to another. Typical applications for this are filter circuits in which some resistor and capacitor values are different for each instance.

## **Sub-Circuits**

Editing a sub-circuit in the usual way allows you to enter a reference, circuit name and possibly some user properties which become sheet properties on the child sheet. The reference would be LEFT or RIGHT in our amplifier example and the circuit name could be AMP.

Connections between the parent sheet and the sub-sheet are made by means of like named module ports on the left and right edges of the sub-circuit, and terminals on the child sheet.

Sub-circuits are most useful in cases where the exact interface between parent and child sheets is not clear at the outset - you can easily add and remove ports and terminals.

### **To set up a hierarchy with a sub-circuit:**

1. Select the *Sub-Circuit* icon and drag out a box for the sub-circuit body using the left mouse button.
2. From the *Object Selector*, select and place the appropriate types of module port on the left and right edges of the sub-circuit body. You will need one port for each interconnection between the parent and child sheets. By convention, one generally puts inputs on the left and outputs on the right.
3. Either directly, or using the *Property Assignment Tool*, assign names to the module ports. These names must correspond with the Logical Terminals that you will place on the child sheet.
4. Edit the sub-circuit itself and give it an instance name (e.g. LEFT) and a circuit name (e.g. AMP). Several sub-circuits may share the same circuit name, but on a given sheet, each should have a unique instance name.

5. Point at the sub-circuit and press CTRL+'C'. This will cause ISIS to load the child sheet. Unless you have specified the name of a circuit that already exists, you should now see a blank drawing.
6. Select the *Terminal* icon and place terminals to correspond with the module ports on the sub-circuit.
7. Again, either directly or with the *Property Assignment Tool*, annotate them to have corresponding name which correspond with the module ports. A netlist compiler warning is generate for any module port which is not matched by a terminal.
8. Draw the circuitry for the child sheet, connecting it where appropriate to the terminals.

### **Module-Components**

Any ordinary component can be made into a module by setting the *Attach Hierarchy Module* checkbox on the *Edit Component* dialogue form. The component's value is taken to be the name for the associated circuit, and the component's reference serves as the instance name.

Connection between parent and child is achieved by having terminals on the child that correspond with the pin names of the parent module-component. This works for hidden power pins too, although this is irrelevant if the *Global Power Nets* option on the *Edit Design Properties* command form is selected.

Module-components are most suited to handling components which need to be expanded in some way for simulation, but kept as a component for PCB design. The *Depth* control on the *Netlist Generator* command form (*Depth* on page 132) provides the means to select what will happen.

#### **To set up a hierarchy with a module-component:**

1. Select and place the component itself in the ordinary way.
2. Bring up the component's *Edit Component Dialogue* form and set the *Attach Hierarchy Module* checkbox. Also ensure that the component's reference and value are suitable as an instance name and a circuit name respectively. In general, the module component will be an IC of some sort, and so its value will make sense as the name of an attached circuit.
3. Point at the component and press CTRL+'C'. This will cause ISIS to load the child sheet. Unless you have specified the name of a circuit that already exists, you should now see a blank drawing.
4. Select the *Terminal* icon and place terminals to correspond with the pins of the parent component.

5. Either directly or with the *Property Assignment Tool*, annotate the terminals to have corresponding name which correspond with the component's pins. If unknown and not drawn on the screen, the name of pin may be established by pointing at its end and pressing CTRL+'E'. A netlist compiler warning is generated for any pin which is not matched by a terminal.
6. Draw the circuitry for the child sheet, connecting it where appropriate to the terminals.

## External Modules

Having created a module-component and its child sheet, it is possible to store the child circuit externally to the design such that it can be re-used in other designs. Furthermore, you can modify the library part for the parent component such that newly placed instances automatically bind to the child module.

### To set up an external module and an associated library part:

1. Set up a hierarchy with a module component as described in the previous section.
2. Zoom to the child sheet, invoke the *Edit Sheet Properties* command from the *Design* menu, and click the *External .MOD File* checkbox. This will create a MOD file with the same name as the library part associated with the parent component.

In the first instance, this file will be created in the same directory as the design file.

3. Return to the parent sheet, tag the parent component and invoke the *Make Device* command.
4. Enter the name of the module file in the *External Module* field.
5. Click *Next* until you come to the *Library Selector* page, then click *Ok* to store the device back into its library.

Newly placed instances of the device will then attach automatically to the MOD file.

## Moving About a Hierarchical Design

There are two ways to move about the hierarchy:

- The *Goto Sheet* command will display the complete design hierarchy in graphical form and you can then move directly to any sheet in the design.
- The *Zoom to Child* and *Exit to Parent* commands on the *Design* menu allow you to move one step down or up the hierarchy. The form must be used via its keyboard shortcut of CTRL+'C' whilst pointing at the module whose sheet you wish to enter.

If you zoom into a module which has not yet been given a circuit name, a new internal one will be chosen automatically. You can edit the circuit name later, but this will detach the old circuit

from the module rather than re-naming the circuit. Any circuits that have been orphaned in this way remain in the design file, but can be removed using the *Tidy* command on the *Edit* menu.

### ***Design Global Annotation***

Where several sub-circuit objects share one circuit name, you will find that editing done through one of them is reflected in them all. It follows from this that you need only draw out each circuit type once. However, each instance has its own set of component references for the objects in the circuit. This is, of course, essential for PCB design where each instance will require separate physical components to make it. The automatic annotator handles this *Design Global Annotation* seamlessly, and you need only be aware that changing a component reference on one sub-sheet does not affect the references on other instances of the circuit. It is possible to disable this feature for specialist applications - see *Non Physical Sheets* below

### ***Non-Physical Sheets***

In some applications where there are multiple instances of a circuit it is preferable for all instances to carry the same annotation. If a flat netlist is produced the equivalent parts in each instance must be distinguished and this is achieved by means of Non Physical Sheets. In our amplifier example, the first IC in each channel would `LEFT_U1` and `RIGHT_U1` - these names being produced by concatenating the Sheet Name (not the sheet title) of the parent module, an underscore and the part reference.

A Non Physical Sheet is selected via the *Edit Sheet Properties* command, having first loaded the relevant sheet. Set the *Non-Physical Sheet* checkbox if you wish the naming of components to be local on a particular sheet.

This functionality is unlikely to be useful where a PCB design is the end result but in can be useful in simulation work, or where a module represents a daughter card.

Net names on sub-sheets are always prefixed in this way, unless they are power nets and the *Global Power Nets* option of the *Edit Design Properties* command form is selected. The result of this is that net names on sub-sheets are 'local'; this is more or less essential where several instances of the same circuit are used.

# NETLIST GENERATION

## INTRODUCTION

A schematic diagram contains two sorts of information: graphical and electrical. The process of generating a netlist is that of extracting the electrical data and writing it in a form that other CAD programs can use. Sadly there is no single standard for netlist files with most vendors 'doing their own thing'. In such circumstances only an international standards committee or a very large and successful vendor can hope to rectify the position. The former has produced EDIF which is so complex as to be virtually useless, and no single vendor has become large enough to set a *de facto* standard. Like many others we have decided to use our own format and provide also for conversion to some of the other file formats in common use.

Our format is called *SDF* standing for Schematic Description Format. SDF is designed to be compact, human readable, and exceptionally easy to process - Visual BASIC will do nicely. SDF is also intended to be an open format - the technical specification will be provided to anyone who asks.

## NET NAMES

So what exactly is a netlist? A netlist is a list of nets and, before you ask, a net is a group of pins that are connected together. In ISIS, a pin is defined by the reference of the component to which it belongs, its type (determined when the relevant device was created) and the pin name or number.

A net can also be named and one of the jobs of the *netlist compiler* is to merge all nets that have been given the same name. Connections between groups of pins can thus be indicated without having to draw wires between them. This facility is useful for avoiding clutter within a sheet and essential for specifying connections between sheets in a multi-sheet design. The following cause a name to be associated with a net:

- Attaching a wire label to any wire in the net - the net takes a name from the wire label.
- Connecting to a Logical Terminal - the net takes a name from the Terminal.

If several of the above occur with different names, the net takes on all the names and will be merged with any other nets which have any of the names. The final SDF netlist will choose one name for the net and a precedence scheme based on the types of the various name-donors is used to choose it - specifically in order of decreasing priority the ranking is:

Power Rails & Hidden Power Pins (see below)  
Bi-Directional Terminals

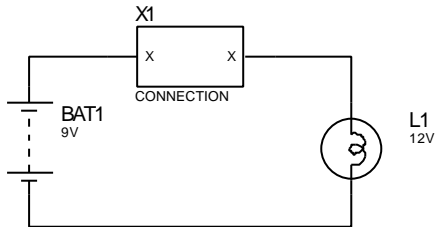
Output Terminals  
Input Terminals  
Generic Terminals  
Bus Entries & Wire Labels

As a special case, unnamed *Power* Terminals assume the name **VCC** and unnamed *Ground* Terminals assume the name **GND**.

Net names can contain any alphanumeric characters plus the minus sign ('-') and the underscore ('\_'). Spaces can be used in the PROTEUS environment but may cause problems for other software. The exclamation mark ('!') and asterisk ('\*') characters have special meanings which are documented later in this section. Net name comparison is case insensitive.

### **DUPLICATE PIN NAMES**

If a device element has more than one pin with the same name, these pins will be considered to be internally interconnected. For example, in a simulation of the circuit below, the light bulb will light, because the two X pins are deemed to be connected.



Typically, this is relevant for devices which have multiple power or ground pins, but it can also be handy when drawing schematics for wiring matrices such as keypad switch or led arrays.

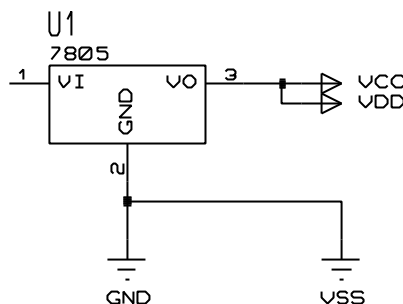
### **HIDDEN POWER PINS**

Many of the integrated circuits in the Device Libraries have hidden power pins. When the netlist generator encounters these, it creates a new net and assigns the name of the hidden pin to it. A 7400 gate will thus generate two nets - VCC carrying pin 14 and GND carrying pin 7. Since all like-named nets are merged, all like-named pins will get connected together.



You can make further connections to these nets by placing name-donor objects (e.g. Logical Terminals) with the same name and then connecting off them. For example, to connect a resistor to VCC, you would place an unnamed *Power* Terminal (which automatically takes the name VCC) and then wire from that to the appropriate end of the resistor.

In some designs, especially if there is a mix of CMOS and TTL logic, you may need to connect two groups of hidden power pins together -VCC and VDD / GND and VSS for example. This can be achieved by placing two *Generic* Terminals, connecting them together with a wire, and labelling them with the net names to be merged. A convenient place to effect this is often at the output end of the PSU circuit - the output of the regulator can connect to several terminals.



In some cases, you may want hidden power pins to connect to different nets from the pin name. This can be achieved by adding appropriately named user properties to the parts which carry the hidden power pin. For example, if attached to a 7400, the property

$$VCC = VCC1$$

would force pin 14 to be connected to VCC1. Note that in the case of a multi-element part such as a 7400, you must add the property to all the gates.

- You can view and edit the net names assigned to the hidden pins of a component by clicking the *Hidden Pins* button on the *Edit Component* dialogue form.
- Note that if you place several pins with the same name, and only some of them are hidden, then they will all be connected together, but not net name will be generated. Instead, they will be connected to the wiring attached to the visible pin(s).

## SPECIAL NET NAME SYNTAXES

### Global Nets

Occasionally, in a hierarchical design, it is useful to be able to make a connection on a child sheet directly to another sheet (root or child) without running up and down the hierarchy. Typically, this requirement arises either when debugging a design with VSM, or else relates to signals such as clocks which are global to the design. Either way, ISIS recognises a leading exclamation mark (!) in a net name as defining a global net. Hence a terminal labelled as !CLK will be deemed connected to any other terminal labelled !CLK, as well as to any terminals labelled just CLK on the root sheets.

Note also:

- It is not necessary to do this for power nets, unless you have de-selected the *Global Power Nets* option on the *Edit Design Properties* dialogue form.
- Unnamed power and ground terminals in fact donate the names !VCC and !GND and so are global unless labelled otherwise.

### ***Inter-Element Connections for Multi-Element Parts***

This feature was introduced specifically to deal with an obscure problem in the creation of VSM models and is unlikely to find general use. However, we document it here for completeness.

Consider a model for a 1458 dual op-amp. Clearly two instances of this model are required and they share the same power connections. However, the 1458 device only has power pins drawn on op-amp A. How then does one specify the power connections for op-amp B?

We solve this by declaring that the terminal net name \*V+ on a child sheet specifies an interconnection between the nets on all child sheets attached to the same parent part, and between those nets and any V+ pins on the parent component elements. The trigger for this mechanism is the leading asterisk (\*) character.

## BUS CONNECTIVITY RULES

ISIS supports bus pins and wiring between bus pins. In the main, this operates intuitively, but it is necessary to be aware of the behaviour of ISIS in some of the more subtle cases that can arise with this feature.

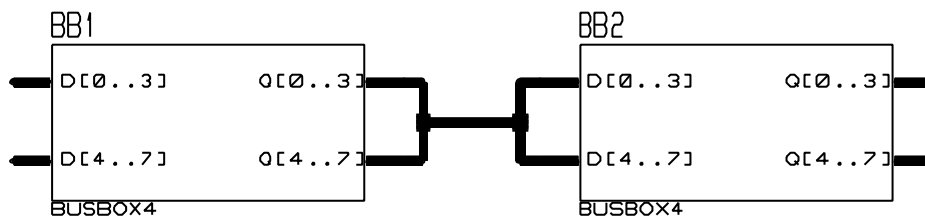
Users of ISIS 3.0X should take note that we changed the bus connectivity behaviour in release 3.1X, in order to make it more intuitive and fail safe.

### The Base Alignment Rule

Within the netlist compiler, all bus entities (pins, terminals and module ports) are assigned a bus range. This is held internally in terms of a base and width so the bus D[0..7] has a base of 0 and a width of 8.

The fundamental basis for ISIS bus connectivity is that all entities on a bus (except for bus labels around a junction dot) are connected by *Base Alignment*. This means that, for example, if two bus pins D[0..3] and Q[4..7] are connected by an unlabelled bus wire, then D0 will be connected to Q4, D1 to Q5 and so on.

The base alignment rule applies, even if the bus pins being connected are different segments of the same bus. For example, the diagram below will be interpreted as making a 4 bit bus which connects Q0 to Q4 to D0 to D4 and Q1 to Q5 to D1 to D5 and so on.

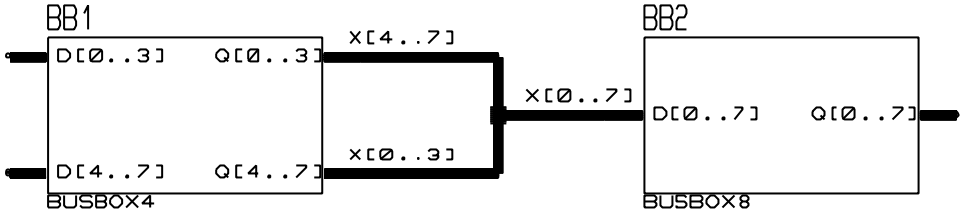


If this is not what is required, then you must use bus labels to designate the required connectivity as explained in the next section.

### Using Bus Labels to Change the Connectivity Rule

The only exception to the Base Alignment Rule is in the situation where several labelled bus sections are combined at a bus junction dot. In this case, the bus sections are combined on a *Like Bit* basis.

The example overleaf shows how some bus pins can be cross-connected using bus labels:

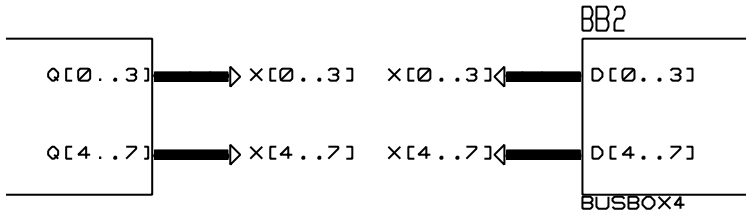


In this case, Q0 connects to D4, Q1 to D5, Q4 to D0, Q5 to D1 and so on. It is worth emphasising here that the choice of name stem for the bus labels is completely unconnected with the names of the bus pins - you could have used D[0..3] etc. but it would make no difference whatsoever to the connectivity.

It is also worth re-emphasizing that the Base Alignment Rule applies in all cases except bus labels at a junction dot, so that the connection between Q[0..3] and X[4..7] connects Q0 to X4 and so on.

**Using Bus Terminals to Interconnect Buses**

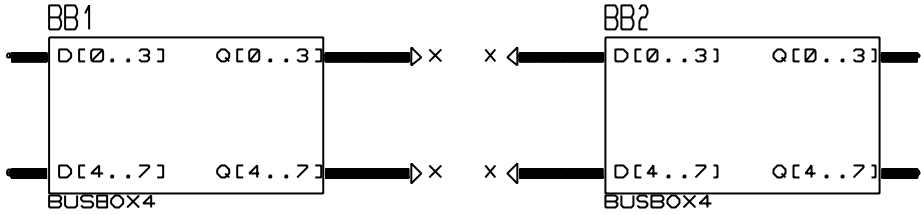
As with ordinary wires, it is possible to connect sections of bus without actually drawing in bus wiring. This can be achieved using bus labels and/or bus terminals as shown below:



If you omit the range specifier of a bus terminal or label, then it will take its range from the bus section to which it connects. The bus range is determined as follows:

- If there are bus labels present in the section, then these are combined on a like bit basis so that the meeting of X[0..3] and X[4..7] at a dot creates a range at the dot of X[0..7]. X[4..7] meeting X[8..11] would create a combined range of X[4..11].
- If there are no bus labels in a section, then the base of its range is 0 (since pins are always base aligned) and the width is that of the widest pin. To put this another way, an unlabelled bus section is always deemed to have a base of 0, irrespective of the ranges of the pins that connect to it.

The latter point does carry the potential to trap the unwary. Consider the diagram overleaf:



Since the X terminals all get ranged as X[0..3], the diagram in fact connects all four bus pins together on a 4 bit bus, rather than creating an 8 bit bus between Q and D. The moral of this is to use explicitly ranged bus terminal labels in all but the simplest cases and always if you are in any doubt.

Note that an isolated section of bus wiring in which there are no bus pins, and in which none of the labels or terminals carries a bus range, is not allowed since ISIS cannot then determine the names and number of individual bits to be interconnected. Instead, you should use a scheme such as the one shown below:



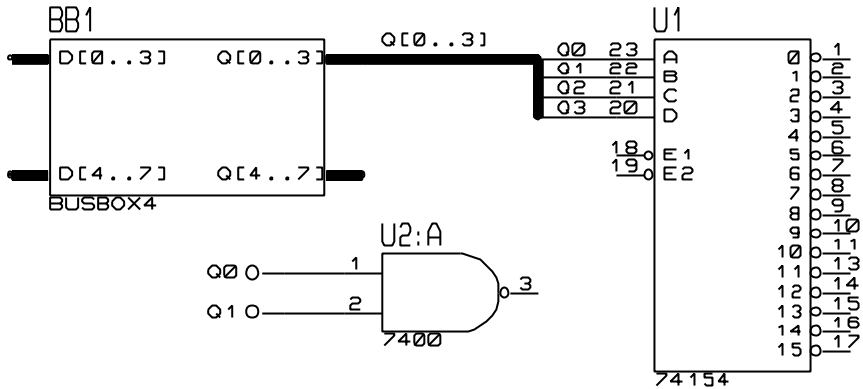
Failure to do this will result in a netlist compiler error.

### Connections to Individual Bits

In most circuits, even where all the major ICs use bus pins, it becomes necessary to connect to individual bits of a bus. To do this, you need to know about the net names that ISIS generates when it encounters a bus label or terminal. You should by now appreciate that when the netlist compiler encounters an ordinary Logical Terminal, or a Wire Label, the object donates a net name to the partial net. All partial nets which have one or more net names in common are then taken to be connected.

When a bus label or terminal is encountered, it generates a set of net names, which are assigned to the partial nets that constitute each bit of the bus. The bus label D[0..7] thus generates the net names D0, D1 ... D7.

In the circuit overleaf, both the NAND gate and the 74154 are connected to Q[0..3] by this mechanism.



There are several things to note from this example:

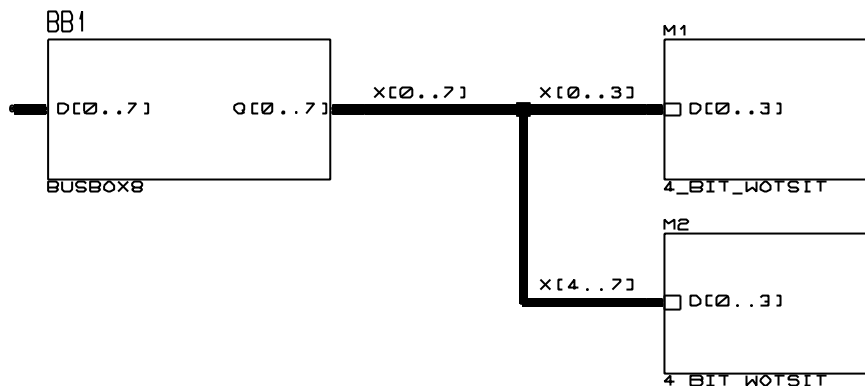
- Connections to the bus bits can be made without actually drawing a wire coming out of the bus. The bus label Q[0..3] gives the bus bits the net names Q0-Q3 and these can then be referred to and connected up in the usual way.
- The bus label Q[0..3] is mandatory. The bus pin Q[0..3] does not contribute any net names as this could lead to unwanted connectivity if there were several like named pins on the design. This behaviour is also consistent with the behaviour of ordinary, non-bus pins which do not donate net names either.

In fact, the junction between a wire and a bus has no significance at all, as far as ISIS is concerned.

- In the above circuit, you could just label the bus as Q, since the bus pin Q[0..3] will contribute the range information.

## Tapping a Large Bus

A not uncommon situation that arises in the use of buses is that of the need to break a large bus into several smaller buses which connect to chips which are perhaps 4 or 8 bits wide.



Here, the 8 bit Q output of the BUSBOX8 is split into two four bit buses which are then fed into the 4\_BIT\_WOTSIT sub-circuit modules. The Base Alignment Rule applies at the connection of X[4..7] to D[0..3] to give the desired result. The label X[0..7] is actually redundant in this case but it does no harm to be explicit.

This diagram also shows how bus connectivity can be combined with hierarchical design to give a very powerful scheme for representing array like circuitry. The module ports behave in the same way as pins for bus connectivity purposes.

## General Comment & Warning

The preceding sections describe the behaviour of bus pins and labels in the contexts that we envisage them being used. Clearly, other (bizarre) possibilities may be drawn and the hope is that the explanations of how bus pins work will enable you to perceive what will happen.

*However, if you are in any doubt whatsoever as to what connectivity will result from what you have drawn, we strongly recommend that you check the resulting netlist with a text editor before simply assuming that what you have drawn will result in the connectivity that you expect.*

This said, you will not easily go wrong if you remember the following two points:

- The Base Alignment Rule always applies, except for the case of labelled bus sections being merged at a bus junction dot.
- Use un-ranged bus terminal labels only in simple cases; an unlabelled bus terminal or module port will take a base of 0 if there are no other bus labels in the bus section.

## **GENERATING A NETLIST FILE**

The *Netlist Compiler* command on the *Tools* menu first presents the *Netlist Compiler* dialogue form and then a file selector from which you choose a filename for the netlist. In most cases, the default settings will suffice; they cause a flat, physical netlist to be generated in SDF format for all sheets in the design. The functions of the various controls are as follows:

### **Format**

A variety of netlist formats can be generated - SDF is the Labcenter Native format, the others are used for interfacing to 3rd party software. Some brief notes are provided in *Netlist Formats* on page 133 and detailed application notes regarding some packages are available from our technical support department. Since 3rd party software is subject to changes beyond our control, it is preferable to supply up to the minute application notes rather than commit things to the printed manual.

### **Logical/Physical/Transfer**

A logical netlist contains pin names whereas a physical netlist contains pin numbers. A more subtle effect is that in a physical netlist, the elements of multi-element parts such as the 7400 will be grouped (appearing, for example, as U1) whereas in a logical netlist they are kept separate (appearing, for example, as U1:A, U1:B, U1:C, U1:D). A logical netlist will generally be used for simulation whilst a physical netlist is best suited to PCB design.

Transfer mode is used in specialist applications of ISIS only, for which separate documentation is provided.

### **Scope**

The default scope is the whole design. Current scope restricts the netlist generator to just the sheet that is loaded. This is generally used when you wish to extract the netlist from a child sheet - perhaps a daughter card which is to be represented as a layout in ARES, but which is part of a larger design that needs to be simulated whole. It is also possible to create a 'virtual test jig' by having a parent sheet which contains circuitry to simulate the components on the child sheet.

### **Depth**

The default mode is to flatten the design. In this case objects with sub-sheets will be replaced by their implementation. If the netlist is not flattened then this replacement will not occur and objects with sub-sheets will appear as-is in the partslist and netlist.



The most common reason for not flattening a design is if some components have attached child sheets representing their simulator models, but you wish to produce a purely physical netlist for PCB design. Note that if you wish to use this approach, you cannot have a hierarchical design as well - even ISIS does not cater for selective flattening! However, with VSM, this is not an issue since you can easily link external model files to the design properly, using **MODFILE** properties. See the VSM manual for further details.

## **Errors**

Various kinds of error can occur whilst a netlist is being generated - the most common is two parts with the same name. If any errors do occur, they will be displayed in the pop-up text viewer window.

## **NETLIST FORMATS**

### **SDF**

Schematic Description Format - the native Labcenter format - used by VSM, ARES and any future Labcenter EDA products. Also very easy to read in and process to other forms. Contains all textual/connectivity information contained in the DSN file.

Use *Physical* mode for ARES.

### **BOARDMAKER**

Netlist format for Tsien Boardmaker II.

The user property PACKAGE is used for the package name if the file is generated from the *Netlist Compiler* dialogue form. If you want to use a different field, you must invoke the netlist generator from a script file.

Use *Physical* mode.

### **EEDESIGNER**

EE Designer III netlist format.

Comment on packaging as for Boardmaker.

Use *Physical* mode.

### **FUTURENET**

Netlist format used by Dash design tools. Also popular for general purpose netlist transfer.

Use *Physical* mode for Pin List, *Logical* mode for Net List.

### **MULTIWIRE**

Multewire netlist format. Also used by EAGLE PCB design.

File format does not contain packaging data.

Use *Physical* mode.

### **RACAL**

RACAL netlist format. Used by RedBoard, CADSTAR etc.

Comment on packaging as for Boardmaker.

Two files are created with CPT and NET extensions.

Use *Physical* mode.

### **SPICE**

SPICE netlist format, also ideal for P-Spice.

Ground net will be node 0, unnamed nets start at 1000; numeric net will be fed straight through. The file SPICE.LXB can be re-named to SPICE.LIB to obtain a set of SPICE compatible models.

Use *Logical* format.

Do not use this format for creating PROSPICE models – use the normal MDF output from the *Model Compiler* – it is far more flexible.

### **SPICE-AGE FOR DOS**

SPICE-AGE netlist format for Those Engineers analogue simulator (available direct from ourselves).

The file SPICEAGE.LXB can be re-named to SPICEAGE.LIB to obtain a set of SPICE-AGE compatible models.

### **TANGO**

Tango netlist format, also used by Protel and others. A good general purpose format too.

Comments on packaging as for Boardmaker.

Use *Physical* mode.

### **VALID**

Valid netlist format used for transfer of ISIS designs to VALID Transcribe package.

Use *Transfer* mode.

### **VUTRAX**

Netlist format for use with VUTRAX PCB design software as used by several design bureaux

Comment on packaging as for Boardmaker.

# REPORT GENERATION

## **BILL OF MATERIALS**

### **Generating the Report**

ISIS can generate a Bill of Materials which lists all the components used in the current design. Facilities are provided to give you extensive control over the contents and arrangement of this report

To actually generate the report, all you have to do is select the name of the Bill Of Materials configuration you want from the *Bill of Materials* sub-menu on the *Tools* menu. The report will appear in the *Text Viewer* and can then be saved or printed as required.

### **Bill of Materials Configuration**

The contents and formatting of the Bill of Materials is determined by a configuration script. One such script, *Default* is supplied with ISIS and you can create, edit or delete scripts using the *Set BOM Scripts* command on the *System* menu. The command displays a *Edit BOM Scripts* dialogue form.

Note that changes made to your Bill Of Materials configuration scripts only affect the currently running copy of ISIS. If you have made changes and you want them to be available next time you start ISIS use the *Save Preferences* command on the *System* menu to save the scripts to the registry.

A sample configuration script is shown below:

```
REFWIDTH=20
FIELD=VALUE,15
TOTAL=COST,10
CATEGORY=M,Modules
CATEGORY=R,Resistors
CATEGORY=C,Capacitors
CATEGORY=U,Integrated Circuits
CATEGORY=Q,Transistors
```

An explanation of the various keywords follows:

<b>REFWIDTH</b>	Determines the number of columns allocated to component references.
-----------------	---

<b>FIELD</b>	Specifies a component property and the number of columns to be allocated to displaying it.
<b>TOTAL</b>	Specifies a component property to be totalled, and the number of columns to be allocated to displaying the result.
<b>CATEGORY</b>	Specifies a category under which to collect similar components.  Parts in the design are categorised according to the alphanumeric part of the reference. Where no suitable category is available the part goes into a category labelled 'Miscellaneous'. Categories appear in the report in the order that they are listed in the configuration file.

The ISIS Bill of Materials configuration facility is very flexible, certainly more so than that provided by many rival products. However, it will not cater for every conceivable requirement. If you need specialised Bill of Materials reporting, your best bet is to develop some software of your own to read an SDF netlist, and write out the data in whatever format you require. This task can be achieved quite easily with only a modicum of programming know-how. Any dialect of BASIC will be up to the task.

## ***ASCII DATA IMPORT***

Although not strictly to do with report generation we shall discuss this here as it is of most significance when viewed in conjunction with the Bill of Materials report.

The idea of ADI is that much of the data you would store within components (e.g. costs, stock codes, tolerances) will remain the same for each component type across all your designs. ADI allows you to specify data for different component types in a simple ASCII file and to import it all into a design with just one command.

The ADI source file can be created with any ASCII text editor - for example the MS-DOS editor EDIT or the Windows editor - NOTEPAD. The file can contain any number of separate commands, each of which starts on a new line with its command keyword and ends on a new line with the keyword **END**. There are two ADI commands, the **IF...END** command and the **DATA...END** command.

The ADI 'interpreter' is run by invoking the ADI command and then selecting an ADI file with the file selector. The commands within the ADI file are then loaded and pre-compiled - any errors will be written to the error log, which is automatically displayed if errors occur. ISIS then applies each command in the ADI file, in order, to each component in the design.

### ***The IF...END Command***

The **IF...END** command allows you to test the existing properties of each component via an expression, and if the expression evaluates **TRUE**, then the sub-commands within the **IF...END** block are applied to the component in succession. The syntax is best explained by example:

```
IF DEVICE="CAP ELEC" AND NOT VALUE=10p
    VALUE=1n, HIDEKWD
    TOLERANCE, HIDE
    STOCKCODE, REMOVE
END
```

The expression to be applied to the component follows the keyword **IF**. Expressions consist of one or more terms separated by operators. Each term consists of a property value, optionally followed by an equals sign and a value.

- A term only evaluates **TRUE** if the property named exists, and in the case where a value has been specified, the property value matches the value specified character-for-character, case-sensitive.
- Operators consist of brackets - which may be used to enclose sub-expressions, and the keywords **AND**, **OR** and **NOT**. The operators **AND** and **OR** are executed left to right - there is no precedence. The **AND** operator evaluates **TRUE** only if the evaluation so far is **TRUE** and the term or sub-expression following it evaluates **TRUE**. The **OR** operator evaluates **TRUE** if the evaluation so far is **TRUE** or the term or sub-expression following it evaluates **TRUE**. The **NOT** operator is unary and has the effect of negating the result of the evaluation of the term or sub-expression that follows it.
- A special expression, consisting of the single keyword **TRUE** allows you to affect all components in the design.

In the example, the expression tests that the component is an instance of the CAP ELEC library device and that it does not have a value of 10p. If this is the case, then the three sub-commands after the **IF** expression and before the closing **END** keyword are applied to the component. Note that, because the device being tested for contains a space (CAP ELEC), it is enclosed in double quote ("" ) characters.

Each sub-command consists of a property name optionally followed by an equals sign and a new value and/or a comma and a command. If a value is specified, the property named is either created with the new value or has its existing value modified. If a command is specified it is executed following any assignment, as follows:

<b>SHOW</b>	Both the property name and its value are made visible.
<b>HIDE</b>	Both the property name and its value are hidden.

<b>HIDEKWD</b>	The property name is hidden; the visibility of the property value is unchanged.
<b>HIDEVAL</b>	The property value is hidden; the visibility of the property name is unchanged.
<b>REMOVE</b>	The property name and any value is removed from the component.

If no command is specified then the existing visibility of the property name and value is unchanged.

In the example, the property **VALUE** is assigned the value 1n and the property name is hidden, the value of the property **TOLERANCE** is left unchanged but the property/value pair is hidden and finally, the property **STOCKCODE** is removed from the component.

### ***The DATA...END Command***

The **DATA...END** command allows multiple test values to be tested against a fixed list of property names, and given a match, allows a set of separate values to be assigned to a separate set of fixed properties.

Consider the following example of a **DATA...END** command:

```
DATA DEVICE + VALUE : COST+, TOLERANCE, STOCKCODE-
RES          1k    : 0.01, 1%,          100-1001
RES          1k2   : 0.01, 1%,          [REMOVE]
"CAP ELEC"   1n    : 0.03, 5%,          200-1001
SWITCH       *     : 0.25, [SKIP],      300-1001
END
```

The **DATA** keyword is followed by the name or names of the properties to be tested, separated by plus characters. This list is followed by a colon and then a list of the name or names of the properties to be assigned, separated by commas. Each property name may optionally be directly followed by a plus or minus character that reflects whether you want the property named and the value assigned to it to be shown or hidden respectively; no character indicates that existing visibility should be maintained.

In the example, the **DEVICE** and **VALUE** properties are tested. Given a match, new values are assigned to the **COST**, **TOLERANCE** and **STOCKCODE** properties; the **COST** property and its value being made visible and the **STOCKCODE** property and its value being hidden.

Each line between the **DATA** keyword line and the **END** keyword consists of a list of the property values that you want to test for (separated by one or more spaces), a colon, and then the list of property values you want to assign, separated by commas. Each line is executed in turn; the value of each property named in the list to the left of the colon on the

**DATA** line is tested against the corresponding test value on current line. If all property values match, then each property named on the right of the colon on the **DATA** line is assigned the corresponding assignment value on the current line.

In the example, the first line tests the **DEVICE** property for the value RES and the **VALUE** property for the value 1k; if both match then the **COST** property is assigned a value of 0.01, the **TOLERANCE** property a value of 1% and the **STOCKCODE** property a value of 100-1001. Similarly the third line tests the **DEVICE** property for the value CAP ELEC (because the device name contains a space it must be enclosed in double quotes) and the **VALUE** property for the value 1n; if both match then the **COST** property takes a value of 0.03, the **TOLERANCE** property a value of 5% and the **STOCKCODE** property a value of 300-1001.

Note that the CAP ELEC device name is enclosed in double quotes. You must do this for any of the match values to the left of the colon as these are space delimited. If this were not done, ISIS would have complained it was expecting a colon since it would have assumed you wanted to test **DEVICE** property against the value CAP and **VALUE** property against the value ELEC; the 1n would then be unexpected. Similarly, the assignment values to the right of the colon are comma delimited, so if you wanted to assign a new value that contains a comma, then you would have to enclose the value in double quotes.

There are two unusual features of the property values that ADI tests for.

- The values can contain the wild-card characters: '?' and '\*'. The question mark indicates that any single character may appear in the property value being tested in same position as the question mark. The asterisk indicates that any number of characters may appear in the property value being tested between the sets of characters to the left and right of the asterisk. In our example above, we test for a **DEVICE** property with the value SWITCH but choose not to test such a component's **VALUE** property by specifying a test value of a single asterisk.
- Numerical values (including values that contain exponent suffixes such as 'k' or 'u'). Such values are converted by ISIS in to a textual representation with six decimal places and an exponent and the textual representations are then compared character by character. In our example, component placed from the library device RES and assigned the values 1k, 1000 and 1.0k would all match the test condition in the first line.

Whilst the list to the right of the colon normally consists of a new value to be assigned to the respective property named on the **DATA** line, you can instead specify a command, as follows:

[ NULL ]	Assign the respective property an empty string. Only the property name and the following equals sign will appear in the component's properties text block.
[ REMOVE ]	Remove the property; the property name and its value are removed from the component's text property block.
[ SKIP ]	Skip assignment to the respective property. This keyword is required as an empty assignment (nothing between the commas) will cause an error. Any existing property in the component's text property block is left unchanged.

The second line of the example specified that the **STOCKCODE** property of any component placed from a RES device with a **VALUE** of 1k2 should be removed. Similarly, line four, specifies that there should be no assignment to the **TOLERANCE** property of a component placed from a SWITCH device.

## ***ELECTRICAL RULES CHECK***

ISIS can check for simple errors in a design by examining the various pin types connected to each net. Obvious examples of errors are outputs connected together or several input pins connected together without a driving source. Terminals are also regarded as having an electrical type - an input terminal connected to an input pin provides a driving source for it.

### ***Generating the Report***

You can generate the ERC report by invoking the *Electrical Rules Check* command on the *Tools* menu. The results are displayed in the *Text Viewer* from where they can be saved or printed as required.

Do bear in mind that not all entries that appear in the report may actually be mistakes (e.g. some input pins might deliberately be left NC) and that more subtle mistakes such as wrong part values will not be detected. Nevertheless, many silly mistakes can be detected at an early stage.

### ***ERC Error Messages***

The first part of the ERC process involves compiling the netlist, and this in itself can result in warning or error messages.

The actual ERC process itself detects two basic categories of error:

- Nets so wired as to be likely to result in contentions - i.e. two outputs trying to drive in different directions, resulting in a large flow of current.



- Nets so wired such that there is no driving source. A net containing only INPUT pins will result in an **UNDRIVEN** error.

The detection of the first type of error is determined by the following table:

	PS	IP	OP	IO	TS	PU	PD	PP	GT	IT	OT	BT	PR
PS	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok
IP	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok
OP	ok	ok	er	w n	er	er	er	ok	ok	er	ok	w n	er
IO	ok	ok	w n	ok	ok	ok	ok	w n	ok	w n	ok	ok	w n
TS	ok	ok	er	ok	ok	ok	ok	er	ok	er	ok	ok	er
PU	ok	ok	er	ok	ok	ok	ok	w n	ok	er	ok	ok	er
PD	ok	ok	er	ok	ok	ok	ok	w n	ok	er	ok	ok	er
PP	ok	ok	ok	w n	er	w n	w n	ok	ok	ok	ok	w n	ok
GT	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok
IT	ok	ok	er	w n	er	er	er	ok	ok	er	ok	w n	er
OT	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok
BT	ok	ok	w n	ok	ok	ok	ok	w n	ok	w n	ok	ok	w n
PR	ok	ok	er	w n	er	er	er	ok	ok	ok	ok	w n	w n

**KEY:**

- |                    |                              |                           |
|--------------------|------------------------------|---------------------------|
| PS : Passive Pin   | GT : Generic Terminal        | ok : No warning or error. |
| IP : Input Pin     | IT : Input Terminal          | w n : Warning issued.     |
| OP : Output Pin    | OT : Output Terminal         | er : Error issued.        |
| IO : I/O Pin       | BT : Bi-directional Terminal |                           |
| TS : Tri-State Pin | PT : Passive Terminal        |                           |
| PU : Pull Up Pin   | PR : Power Rail Terminal     |                           |
| PD : Pull Down Pin |                              |                           |
| PP : Power Pin     |                              |                           |



# HARD COPY GENERATION

## ***PRINTER OUTPUT***

Output through standard Windows device drivers is performed using the *Print* command on the *File* menu whilst the device to print to may be selected using the *Set Printer* command. This command also allows you access to the device driver specific setup dialogue form. The *Print* dialogue itself has a number of options which are covered in the Context Sensitive Help.

## ***PLOTTER OUTPUT***

Windows support for pen plotters is, unfortunately, very poor. Although drivers are supplied for HPGL and other plotters, the implementation of these drivers is very sketchy. Better drivers may be available for particular plotters, but we have not relied on this in designing the plotter support within ISIS for Windows. Instead, we rely on the Windows plotter driver only to draw straight lines and then ISIS itself does the rest.

Plotter output is generated as for bitmap printer output using the *Print* command on the *File* menu. Where a plotter is selected as the output device, the *Labcenter Plotter Driver* option is enabled on the *Print Design* dialogue form. If this option is checked then ISIS only relies on the plotter driver's ability to draw a line on the plotter with ISIS linearising arcs, circles, etc. and rendering text in a vector font. If this option is not checked then ISIS treats the plotter driver just like any other driver and expects it to sensibly handle all Windows GDI (Graphics Device Interface) calls, including translation of functions the plotter doesn't support (for example, Bezier curves in to

### ***Plotter Pen Colours***

The plotter drivers are able to generate a multi-coloured plot according to the colours selected on the *Set Colours* command.

Unfortunately, the Windows plotter drivers do not (as far as we have been able to find out) support the direct selection of plotter pens by number. Therefore it is up to you to find out how your plotter driver maps colours onto pens and to select appropriate colours on the *Set Colours* dialogue.

## ***CLIPBOARD AND GRAPHICS FILE GENERATION***

As well as printing directly to Windows print devices, ISIS for Windows can generate output for use by other graphics applications. You have the choice of generating this output as

either a Bitmap or a Windows Metafile, and you can transfer the output to the other applications either through the clipboard, or by saving it to a disk file.

### ***Bitmap Generation***

The *Export Graphics Bitmap* command on the *File* menu will create a bitmap of the board and place it either on the clipboard or in a disk file. The dialogue form offers several additional options, each of which has an associated help topic.

### ***Metafile Generation***

The Windows Metafile format has the advantage of being truly scaleable where a bitmap is not. However, not all Windows applications (e.g. *Paintbrush*) can read a metafile.

The *Export Graphics Metafile* command on the *File* menu will create a bitmap of the board and place it either on the clipboard or in a disk file.

### ***DXF File Generation***

The DXF format can be used to transfer output to DOS based mechanical CAD applications (it is better to use a clipboard metafile to transfer to Windows based CAD programs). The file is generated by a Labcenter output formatter, rather than by Windows, and as such many of the Windows based drawing appearance attributes will be lost.

Our current experience is of considerable incompatibility and disagreement between applications on what constitutes a valid DXF file. To put this another way, given six applications supporting DXF, only about 30% of file exchange pairings seem to work! This not withstanding the fact that our DXF has been tested with the official Autodesk applications (AutoCAD, AutoSketch etc.). For Windows work, the Clipboard provides a much more reliable transfer medium.

### ***EPS File Generation***

An EPS file is a form of Postscript file that can be embedded in another document. Although popular in the world of DTP, for Windows based DTP work you are much better off transferring graphics using a clipboard metafile.

EPS file generation shares the same output options as *Printer Output* on page 143.

# ISIS AND ARES

## INTRODUCTION

ARES is Labcenter Electronics' high performance PCB design system offering the same user interface as ISIS and full netlist based integration with it. Using ISIS and ARES together you are guaranteed to produce a PCB layout which matches the schematic exactly and the use of a netlist in PCB design also saves you from worrying about the details of IC pin numbering when laying out the board. In addition, a netlist is a more or less essential where autorouting is to be used.

In ISIS, the *Netlist to ARES* command is located on the *Tools* menu and when invoked causes one of two things to happen:

- If no copy of ARES is running, then a copy will be started with command line arguments causing it to load the appropriate PCB file and read the netlist from ISIS.
- If a copy of ARES is already running, then ISIS locates it and sends it a message telling it to read the new netlist.

Whilst much of the discussion relates specifically to ARES, users of alternative PCB design systems will find much of interest in the following sections.

## PACKAGING

In order that ARES knows which library packages must be used for given components, this information must be entered somewhere in the design process. With ISIS and ARES, the best time to enter it is either whilst creating new library parts or when editing the schematic. The property to use for this purpose is, not improbably, called **PACKAGE**. Most of the library parts in the supplied libraries have this property embedded.

### Default Packaging

Most of the library parts supplied with ISIS have suitable packages for PCB layout associated with them, and these packages can be viewed when you browse the libraries. Note that some parts - e.g. simulator primitive models do not represent real world components and so do not have packages.

ISIS supports multiple packagings for a single schematic symbol, and where appropriate we have utilized this to offer both through hole (PTH) and surface mount (SMT) packagings for the parts in the libraries.

*Note, however, that the responsibility lies wholly with yourself to check that the defaults we have used are suitable for your particular application. We cannot and will not accept any liability for losses due to boards being wasted as a result of unsuitable PCB packaging, however this may have arisen. We recommend most strongly that you make a one off prototype before commissioning the manufacture of large numbers of PCBs.*

### **Manual Packaging**

User properties including **PACKAGE** can be edited in a variety of ways. The simplest is to select the *Instant Edit* icon, click on the component whose field you want to edit, and then type in the new property information into the *PCB Package* text field. Where the components have already been given references, you can use the *Edit* command to access them by name. For example, the sequence 'E', Q1, ENTER will bring up the dialogue form for transistor Q1.

Where many components have the same package - e.g. resistors are almost always packaged with RES40, the *Property Assignment Tool* can be used to load this fixed value into the desired field of each component you click on. For example, to load the string RES40 into the **PACKAGE** field of a group of components you would set the PAT string to PACKAGE=RES40.

### **Automatic Packaging**

The *ASCII Data Import* feature can be used for the automatic packaging of component types which always take the same package. As an example of this, the file DEVICE.ADI is supplied with ISIS and shows how packages and other properties were assigned to the parts in DEVICE.LIB.

The file is set up such that packages are loaded into the **PACKAGE** property which is then hidden. You can change this by editing the ADI block headers - see *Report Generation* on page 135 for more information on ADI.

Some components such as ordinary and electrolytic capacitors come in many different shapes and sizes. Furthermore, we cannot know what particular capacitor families you will want to use. Nevertheless, it is possible to build ADI files that operate on part values as well as device library names. For example, the extract below packages capacitors from our favourite suppliers, and at the same time, loads the order codes for them into property **ORDER**.

Note that packaging is only done for capacitors which are not of the default size specified in DEVICE.LIB. Running CAPS.ADI would override the default assignments to achieve the desired result.

```

;CAPACITOR PACKAGING FOR TYPES REQUIRING
;OTHER THAN CAP10 / ELEC-RAD10 PACKAGES
DATA DEVICE + VALUE : PACKAGE-
CAP 1u : CAP20
"CAP ELEC" 470u : ELEC-RAD20
"CAP ELEC" 1000u : ELEC-RAD30
"CAP ELEC" 2200u : ELEC-RAD30
END

```

```

;CAPACITOR ORDER CODES
;These are Ceramic, Polyester types and
;Electrolytics at 25V or better.
DATA DEVICE + VALUE : ORDERCODE-
CAP 100p : F146-447
CAP 680p : F146-457
CAP 1n : F149-100
CAP 1n5 : F149-101
CAP 3n3 : F149-103
.
.
.
"CAP ELEC" 47u : R11-0235
"CAP ELEC" 100u : R11-0245
"CAP ELEC" 220u : R11-0260
"CAP ELEC" 470u : R11-0280
END

```

In general, there will always be a final phase of manual packaging to do for unusual parts, or for special components used in that design only. We suggest that you make the **PACKAGE** properties of manually packaged parts visible as a reminder that they are non-standard.

### ***Using the Bill of Materials to Help with Packaging***

In the case of largish designs, it can become difficult to ascertain whether all the components have been packaged, and which packages have been assigned. This is especially true if you have made the **PACKAGE** property invisible during *ASCII Data Import*.

One way round this is to create a supplementary Bill of Materials configuration file which has the following **FIELD** records:

```

FIELD=VALUE , 15
FIELD=PACKAGE , 15

```

so that both the part value and the package appear for each component. The Bill of Materials then gives you a sorted list of all the components and their packages making for much easier checking.

Note that you can create multiple Bill of Materials configuration scripts using the *Set BOM Scripts* command on the *System* menu.

### ***The Package Verifier***

ISIS can verify that all the packages specified for components on a schematic exist in the ARES libraries, and that all the pin numbers specified in the ISIS library parts exist in the specified packages.

Whilst this facility is most useful when creating library parts *en masse*, it is also useful if you wish to verify that all the packaging is correct prior to netlisting to ARES.

The *Verify Packaging* command may be found on the *Library* menu.

### ***Packaging with ARES***

If you miss out packaging information from any of the components on the schematic, ARES will prompt for it when it loads the netlist. Indeed, you do have the option of doing all the package selections at this stage. However, packaging selections made this way will have to be re-entered - at least for unplaced components - each time the netlist is loaded into ARES.

Note that we do not recommend this method of working.

## ***NET PROPERTIES AND ROUTING STRATEGIES***

ARES associates with every net in the netlist a routing "strategy" which defines track and via styles, net type, routing layers and so forth. By default, all nets take the **SIGNAL** strategy except:

- The nets VCC and GND which take the POWER strategy.
- Nets with names such as D[0] which take the BUS strategy. It is the presence of the square brackets which is important. This syntax is really provided for use with schematics packages which could not otherwise convey a net property for a bus.

To override the default strategy assignment for a given net, you need to attach a *Net Property* to one of the wires on the net. This is achieved by placing a wire label of the form:

```
STRAT=strategy_name
```

If the named strategy does not exist, ARES will create it and add it to the strategy selector when the netlist is loaded. The strategy definition can then be edited by selecting the strategy and clicking on the strategy selector toggle.



It is perfectly legal to place an assignment such as:

```
STRAT=BUS
```

on a bus segment, though this will not work in the case of a 'cosmetic' bus. Such a bus is one which connects to no bus pins or terminals, and does not carry a bus net label.

Sometimes it is appropriate to specify a strategy for all the nets on a given sheet. For example, you might want all the nets on the power supply sheet to be assigned the POWER strategy.

This could be done by placing a script block as follows:

```
*NETPROP  
STRAT=POWER
```

on the appropriate sheet of the design. See page 72 for more information.

Wire and bus label net properties take precedence over sheet global net properties.

## **FORWARD ANNOTATION - ENGINEERING CHANGES**

The term 'Engineering Change' refers to a situation in which an existing schematic is modified, and the resulting netlist re-loaded into ARES. This can occur both during the development of an original design, and also at a later date where a 'Mark II' version of the product is being produced. The PROTEUS system fully supports engineering changes, but it is important that you appreciate what the software does in the various circumstances that can arise.

### **Adding New Components**

Adding new components and associated wiring to a design poses little problem *provided that you use the Auto-Annotator in incremental mode*. If you add components to the schematic and totally re-annotate the schematic (such that the part IDs of *existing* components get changed or exchanged) then ARES will not in general be able to make sense of the incoming netlist.

#### **To add new components to a design**

1. Place and wire up the components in ISIS in the usual way.
2. Use the Auto-Annotator in *incremental* mode to give the parts new unique component IDs. Alternatively, you can do this manually. Under no circumstances should you change the component IDs of existing components.
3. Transfer the changes to ARES using the *Netlist->ARES* command on the *Tools* menu. ARES will display the new components in the *Component Selector*.
4. Place the components in ARES in the usual way. ARES will then display ratsnest lines to indicate the required connections to the new components.

5. Route tracking to connect up the new components using automatic or manual routing as appropriate.

### ***Removing Existing Components***

Again, this is fairly straightforward to manage. ARES will tag the components that have been removed from the netlist so that you can see what has been removed before it is actually deleted from the PCB.

#### **To remove components from a design**

1. Delete the components from the schematic in the usual way. ISIS will automatically remove any wiring to them.
2. Transfer the changes to ARES using the *Netlist->ARES* command on the *Tools* menu. ARES will tag & highlight the components on the PCB that no longer appear in the netlist.
3. Examine the tagged components to ensure that you really do want to delete them. For each component that you want to delete, point at it and click right.
4. Remove residual tracking to the old components by using the usual route-editing tools in ARES.

### ***Changing the connectivity***

Where existing wiring is changed, ARES analyses the current connectivity of the board and marks sections of tracking that make connections no longer in the netlist as VOID. This is indicated by them appearing flashing yellow. Connections appearing in the netlist but not present on the PCB are shown as ratsnest lines in the usual way.

#### **To change the connectivity of a design**

1. Make changes to the wiring and connectivity of the schematic in ISIS. The change procedure is not affected by whether the changes involve drawn wires or syntactical changes such as wire and terminal labels.
2. Transfer the changes to ARES using the *Netlist->ARES* command on the *Tools* menu. ARES will mark tracks that make connections that are not in the netlist as VOID and they will appear in flashing yellow. Connections present in the netlist but not made on the PCB will appear as new ratsnest lines.
3. Inspect the void tracking to ensure that you really do want to remove it. If so, use the *Tidy* command to do so.

4. Form tracking to make the new missing connections using automatic or manual routing as you see fit.

## ***Re-Annotating Components, and Re-Packaging Gates***

This area is potentially the most confusing. There is a tendency to think that if you have a particular component in ISIS, say 'U35', that if you edit it in ISIS and call it 'U34' that this will simply be reflected in ARES when you next compile and load the netlist. *This is not the case.* What such a re-annotation means, in the context of PROTEUS, is that 'U35' has been removed, and that 'U34' has been added. Consequently, ARES will tag 'U35', and add a new part 'U34' to the parts selector.

Similarly, if two 7400 gates, say 'U1:A' and 'U2:B' are re-annotated on the schematic such that their names are exchanged, this will be seen by ARES as a change to the connectivity of the design, in that wires going to pins 1,2,3 will now go to pins 4,5,6 and vice-versa.

The key concept is that PROTEUS uses the part IDs in ISIS as the cross reference between the schematic and the PCB. If you change these IDs, then you will change the connectivity of the design rather than its annotation.

Changes to the annotation of the design must be made in ARES, from where they can be back-annotated into ISIS - see *Re-Annotation* on page 156.

## ***PIN-SWAP/GATE-SWAP***

Many types of component have either interchangeable pins - for example the two enable pins on a 74138, or interchangeable elements - for example the six inverters in a 7404. Some components such as the 7400 have both - there are four interchangeable gates each of which has two identical inputs. This phenomenon can be exploited when routing complex PCBs in that it may turn out to be easier to route to an alternate pin or gate to the one specified in the netlist. Clearly, making such a 'spur of the moment' decision will affect the connectivity of the design, and this change needs to be recorded and eventually reflected back on the schematic.

PROTEUS provides an extremely comprehensive - probably unrivalled - degree of support for this aspect of the EDA problem. Specifically, the following features are provided:

- A scheme for specifying swappable pins and gates within ISIS library parts.
- The ability to mark a pin or gate in ARES and see a graphical display of which other pins and gates it can be swapped with based on the data from the ISIS library parts.
- A gate-swap optimiser which will attempt to find the optimum gate allocations based on achieving the shortest possible ratsnest length.
- Automatic back-annotation of both manual and automatic changes into ISIS.

- A locking mechanism which prevents changes being made to both schematic and PCB simultaneously, thus preventing conflicts and ambiguities.

### **Specifying Pin-Swaps and Gate-Swaps for ISIS Library Parts**

*The following discussion assumes a detailed understanding of the creation of ISIS library parts. Please refer to the chapter entitled Library Facilities on page 91 if you need further information.*

The first thing to appreciate is that pin-swaps can occur for both single element and multi-element devices whereas gate-swaps require the device to have multiple elements more or less by definition. Thus there are three cases.

### **Specifying Pin-Swaps in Single Element Devices**

This is achieved through the use of the **PINSWAP** property; the *pin-names* (not numbers) of the interchangeable pins are listed. Thus, a resistor for which the two pins named '1' and '2' are interchangeable can be given the property

PINSWAP=1 , 2

whilst the 74138, which has identical input pins 'E2' and 'E3' carries the property:

PINSWAP=E2 , E3

Where more than one set of pins is interchangeable, a semicolon may be used to separate the pin sets. For example:

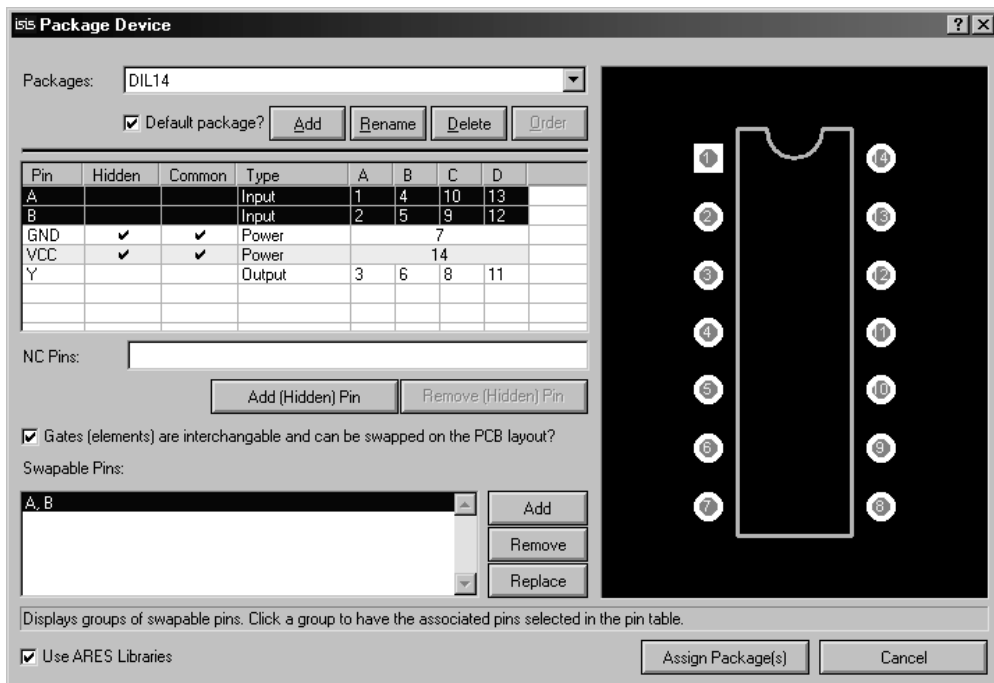
PINSWAP=A , B ; C , D

means that A can be swapped with B, and C can be swapped with D, but that the swaps A-C, A-D, B-C and B-D are still illegal.

Only one **PINSWAP** property may be used.

## Specifying Pin-Swaps in Multi-Element Devices

For multi-element parts such as the 7400 shown below, you must use the *Visual Packaging Tool* to specify pinswap groups. This is achieved using the *Pinswap* section of the dialogue form.



### To add a pinswap group to a packaging:

1. Highlight the pins to be swapped in the *Pin Grid* by holding down the CTRL key and clicking on each row in turn. In the above case, pins A and B are swappable, so you would highlight their rows as shown.
2. Click the *Add* button to create the pinswap group.

You can define multiple swap groups by repeating the above procedure.

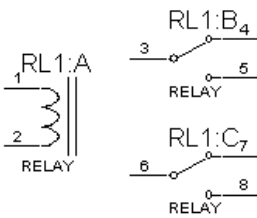
## Specifying Gate-Swaps in Multi-Element Devices

It is a happy fact that ISIS can work out the much of the gate-swap information for a multi-element device entirely by itself. For example, in the case of the 7400 (show overleaf), the

packaging script contains sufficient information to tell that the swappable element consists of pins A,B and Y and that there are 4 of them.

The only requirement is that we tell ISIS to allow gate swaps using the *Gateswap* checkbox in the *Visual Packaging Tool*. Note that the power pins VCC and GND are automatically excluded from the swappable element because they are hidden. They are also required to be on the same nets if a swap is to be made between different packages, since they are marked as being common.

Our final example is of a heterogeneous multi-element part - in this case a DPDT relay consisting of a coil and two pairs of contacts:



Pin	Hidden	Common	Type	A	B	C
C1		✓	Passive	1	---	---
C2		✓	Passive	2	---	---
COM			Passive	---	3	6
NC			Passive	---	4	7
NO			Passive	---	5	8

NC Pins:

Gates (elements) are interchangeable and can be swapped on the PCB layout?

Swappable Pins:

C1, C2

In this case, both pins of the coil are both *common* and swappable. Omitting the *common* would, of course, be disastrous as it would allow swaps of contacts between different relays!

### Performing Manual Pin-Swaps and Gate-Swaps in ARES

Pin-swaps and gate-swaps are both carried out in exact the same way using the *Ratsnest* mode in ARES.

#### To swap two pins or gates:

1. Load the netlist for the design from ISIS. ARES must have a copy of the netlist that is synchronized with the schematic before this feature may be used.
2. Select the *Ratsnest* icons in ARES.

3. Click right on the source pin. The ratsnest lines connected to it will highlight.
4. Click and hold down the left button. All pins with which the source pin can be swapped will highlight. This set includes all the valid pin-swaps within the source gate and all the same pin-sets in any valid gate-swaps.
5. Drag the source ratsnest lines to the destination pin you want to swap with. If this pin is in the same gate then just a pin swap will be performed. If it is in another gate, then a gate-swap and possible also a pin-swap will occur.

There is little more to it than that. Note, however, that ARES will not allow swaps where any of the source or destination pins have tracking attached. In the case of a gate swap, all pins of both gates must be unrouted.

### **WARNING**

*Pin-swaps and Gate-swaps constitute changes to the connectivity of your design. ARES uses the pin-swap and gate-swap data specified in the ISIS libraries to decide what is, and is not, a valid swap. If there are errors in this data, then ARES may well suggest illegal swaps. We will not, under any circumstances, be held liable for any costs incurred or losses arising as result of such mishaps, whether the error be in your library parts or ours or in the software itself. We strongly recommend that you check that the swaps you make are legal, and that you prototype your PCB prior to manufacturing large quantities.*

## **The Gate-Swap Optimizer**

In cases where a design has large numbers of swappable gates, it can be very hard to see what the best gate-allocation should be. The number of possible combinations can become literally astronomical even for small numbers of gates. The sample design SHIFT16 contains many more combinations than there are particles in the universe!

To help you find a near optimum solution, ARES incorporates an automatic gate-swap optimizer which will perform thousands of trial swaps in such a way as to find what is technically called a 'local minima' for a given board placement. In many cases this is very near optimum, and almost invariably gives some reduction in the total ratsnest length.

### **To use the Gate-Swap Optimizer:**

1. Load the netlist for the design from ISIS. ARES must have a copy of the netlist that is synchronized with the schematic before this command may be used.
2. Place all the components in the usual way, aiming to achieve as low a ratsnest length as possible. The swap optimizer is no excuse for bad placement!
3. Invoke the *Gate-Swap Optimizer* command from the *Tools* menu.

The algorithm makes repeated passes until such time as it can make no further improvement. Run times can be quite long (30 minutes) if there lots of possible swaps.

### **WARNING**

*The Gate-Swap Optimizer relies entirely on the gate-swap data specified in the ISIS component libraries to decide what is, and is not, a valid swap. If there are errors in this data, then the swap-optimizer is likely to make erroneous changes to the connectivity of your design. We will not, under any circumstances, be held liable for any costs incurred or losses arising as result of such mishaps, whether the error be in your library parts or ours or in the software itself. We strongly recommend that this command be used only if you are going to prototype your PCB prior to manufacture.*

## **RE-ANNOTATION**

In some circumstances you may wish to renumber components on the PCB to suit manufacturing or assembly activities. This can be done either manually or automatically.

### **To re-annotate manually:**

1. Ensure that an up-to-date copy of the netlist is loaded into ARES.
1. Select the *Edit Mode* icon in ARES, and the correct layer for the component(s) you wish to annotate.
2. Click on the component labels to edit them.

You will get an error message if you attempt to make changes which cannot be back-annotated to the schematic. Connectors made with physical terminals are the most significant example - because there is no named carrying entity in ISIS, there is nowhere for the change to be applied.

### **To re-annotate automatically:**

1. Specify any components you do not wish to be renumbered by giving them a **NOANNOTATE=TRUE** property on the schematic.
2. Ensure that an up-to-date copy of the netlist is loaded into ARES.
3. Invoke the *Component Re-Annotator* command from the *Tools* menu in ARES. This will renumber the components based on their existing prefixes and their physical positions on the PCB.

Either way, any changes made will be reflected on the schematic the next time you save the PCB.



## **BACK-ANNOTATION WITH ISIS**

Once pin-swaps, gate-swaps or annotation changes have been made, you will want to transfer the effects of these change back to ISIS so that the schematic accurately reflects both the annotation and the connectivity of the PCB. There are two ways of working this:

### ***Semi-Automatic Back-Annotation***

By default, you will be prevented from making changes to the schematic whilst open with unsaved changes. The menu bar in ISIS will contain the text '(Locked)' to indicate this state of affairs.

To re-synchronise and unlock the schematic, use the *Back-Annotate from ARES* command on the *Tools* menu in ISIS. This command will cause ARES to save its changes.

### ***Fully-Automatic Back-Annotation***

In this case, the schematic will update whenever you bring ISIS to the foreground. However, a consequence of this is that ARES will automatically save its changes to disk.

To select fully automatic operation, check the *Auto Sync/Save checkbox* on the *Set Environment* dialogue from the *System* menu in ISIS. The default mode is semi-automatic.

In both cases, back-annotation is automatic if the PCB is saved after the changes have been made.



## @

@PROPERTIES 38

## A

Active Components 101  
 Annotating components 11  
 Apply Default Template Command 46  
 Apply Template From Design Command 46  
 Arc graphic 86  
 ARES 2, 145  
 ASCII Data Import  
   in PCB design 146  
   reference 136  
 ATTRIB program 91  
 Auto Dot Removal 67  
 Automatic Annotator  
   reference 37  
   tutorial 12  
 Automatic dot placement & removal 67  
 Automatic Wire Routing 35

## B

Backup files 29  
 Bill of Materials  
   and parameterized circuits 53  
   configuring 135  
   generating 135  
   tutorial 21  
   use in PCB design 147  
 Bitmap 144  
 Block  
   commands tutorial 10  
   copy 33  
   delete 34  
   move 34  
 BMP file 144  
 Boardmaker (netlist format) 133  
 BOM Command 135  
 Border 15, 37  
 Box graphic 85  
 Bring to Front command 40  
 Bus labels

deleting 76  
 in netlist generation 128  
 placing and editing 76

Bus pins 105, 111

### Buses

connecting to individual bits 77, 129  
 connectivity rules for 127  
 definition 75  
 placing 75  
 BUSNODE marker 89, 111

## C

CadStar 134  
 Capacitors  
   packaging for PCB design 60, 146  
 Changing Schematic Appearance 42  
 Child sheet 118  
 Circle graphic 85  
 Circuits  
   automatic creation of 121  
   definition 118  
   sharing in hierarchical design 122  
 Clipboard 143  
 Common pins 96, 107  
 Component Properties 102  
 Components  
   annotating 11, 37  
   assigning packages for PCB design 145  
   definition 63  
   editing 11, 65  
   editing by name 32  
   finding 32  
   in hierarchical design 120  
   placing 7, 64  
   properties 66  
   replacing 20, 65  
   replacing from library 91  
   selecting from libraries 63  
 Configuring  
   Bill of Materials 135  
 Connection points 34  
 Connectors 81  
 Coordinate Display 26

Co-ordinates	6, 26
Copy icon	33
Copying	
between designs	29
tagged objects	33
Creating	
graphics symbols	92
multi-element heterogeneous devices	110
multi-element homogenous devices	109
new design	28
single element device	97
single-element devices	108
user defined device pins	95
user defined module ports	94
user defined terminals	93
Cursor types	26

## D

---

Data Sheets	103
DATA...END (ADI command)	138
Default Font	43
Default Properties	48
Default properties for device	112
DEFINE keyword	49, 72
Delete icon	34
Deleting	
Bus Labels	76
general procedure	30
tagged objects	34
Wire Labels	68
Design	
properties	51
Design Global Annotation	118, 122
Device pins	
creating	95
Device Properties	101
Devices	96
advanced tutorial	16
default properties	112
editing	114
multi-element heterogeneous	110
multi-element homogenous	109
single element	108
tutorial	13
with bus pins	111

Displaying design information on the schematic	38
Dots	66
Dragging	
general procedure	30
labels	31
objects	8
wires	36
Duplicate pin names	124
DXF	
file generation	144

## E

---

E12,E24 keywords	55
Eagle PCB software	133
EDIF (netlist format)	123
Edit BOM Scripts command	135
Editing	
components	65
devices	114
general procedure	32
Pin objects	84
pinout	115
sub-circuits	80
symbols	96
terminals	82
Wire Labels	67, 68
Editing Global Styles	42
Editing Local Styles	44
Editing Window	5, 24
EEDesigner (netlist format)	133
Electrical Rules Check	
error messages	140
generating	140
tutorial	21
Electrical types	
of device pins	14, 99
of terminals	94
Encapsulated Postscript	144
EPS file	144
Example files	5
Exit to Parent command	22, 121
Export	
Bitmap	144
DXF	144
EPS file	144

Windows Metafile	144
Export Section command	29
External module files	121
External text editor	32

## F

False origin	27
FIELD keyword	
for packaging connectors	81
reference	71
File types	28
Files	
loading	28
saving	16, 29
Find and Edit command	32
Fonts	43
Footprint	105
Futurenet (netlist format)	133

## G

Gate swap	108
Global net names	125
Global Styles	
See Styles	41
Goto Sheet command	117
Graphics	
placing	85
reference	85
resizing	87
text	87
tutorial	15
Graphics files	143
Graphics Styles	
See Styles	41
Graphs	3
Grid dots	6, 27

## H

Hard copy	
tutorial	16
Header block	15, 20, 38
HEADER symbol	38
Help Topics	103
Heterogeneous multi-element devices	96, 110
Hidden pins	66, 107
Hiding	

power pins	113
text	48
Hierarchical design	51, 117
external modules	121
navigating	121
sub-circuits	80
tutorial	22
with module components	120
with sub-circuits	119
Homogenous multi-element devices	109
Homogenous multi-element devices	96

## I

Icons	6
IF...END (ADI command)	136
Import Section command	29
Incremental annotation	12
ISIS II	
converting files	28

## J

Junction Dots	See Dots
Justifying	
labels	33

## K

Keyboard shortcuts	5
--------------------	---

## L

LABEL marker	89, 93
Labels	
dragging	9, 31
editing	33
resizing	57
Library	
read only	91
selector	7
supplied files	91
Library parts	See Devices
Like bit connection rule	127
Line graphic	85
LISA	
introduction	2
List of @PROPERTIES	38
Load design command	28

## LABCENTER ELECTRONICS

---

Loading files	28
Locked Design	157
Logical netlist	132
Logical terminals	80

### M

---

Make Device command	100
Make Symbol command	93, 95
MAP ON keyword	49, 73
Markers	88
Menu bar	5, 23
Metafile	144
Mirror icon	8, 24, 31
MOD files	121
Mode selector toolbar	23
MODELS keyword	73
Module components	120
Module ports	
creating	94
placing	79
Modules	118
Move icon	34
Moving	
tagged objects	34
Multi-element devices	16
Multi-element parts	96, 105
Multi-sheet designs	22, 117
Multewire (netlist format)	133

### N

---

Named scripts	74
Navigating a hierarchical design	121
NC pins	108
Net names	
assigning with logical terminals	81
assigning with wire labels	69
definition	123
design global	125
in hierarchical design	122
legal characters in	124
Net properties	
assigning with NETPROP keyword	72
assigning with wire labels	69
use for routing strategies	148
Netlist	
compiler command	132

definition	123
errors	133
generating	132
NETPROP keyword	72, 149
Network compatibility features	4
New Design command	28
New Sheet command	117
NODE marker	89, 93
Non –Physical Sheets	122

### O

---

Object selector	26
Object Selector	63
Objects	
deleting	30
dragging	30
editing	32
Editing Local Styles	44
editing properties	48
placing	29
reorienting	31
resizing	31
shuffling database order	40
tagging	30
Ordinal number of sheets	117
Orientation toolbar	24
Origin command	27
ORIGIN marker	88, 93, 98
Overbars	99
Overview Window	5, 25

### P

---

PACKAGE property	145
Package Verifier	148
PACKAGE.ADI file	146
Packages for PCB design	145
Packaging	
with hidden power pins	114
Packaging Tool	105
Panning	5, 24
Parameter Mapping Tables	73
Parameterized circuits	51, 119
Parent sheet	118
Parts bin	26
PCB design	2, 145
PCB Package	105

PDF files	103	in netlist generation	124
Pen colours	143	Prefix	101
Physical netlist	132	Printing	143
Physical terminals	80, 81	Properties	
Pick command	10, 64, 91	in PCB design	145
Picking library parts	7	Property Assignment Tool	55
Pin		examples	59
NC	108	tutorial	11
Pin grid		Property definitions	102
in packaging tool	107	Property Definitions	48
Pin objects	83	Property Expression Evaluation	54
Pin swap	108	Property Substitution	53
PINNAME marker	89, 95	ProSPICE	2
PINNAME property	99	Protel (netlist format)	134
PINNUM marker	89, 95	PSPICE (netlist format)	134
PINNUM property	99		
Pinout		<b>Q</b>	
editing	115	Quick keys	5
Pins		Quit command	29
common	107		
hidden	66	<b>R</b>	
hiding	107	Racal (netlist format)	134
placing for new device	98	Real Time Annotation	9, 11
with duplicate names	124	Real Time Snap	9, 27
with overbars	99	Re-annotation	156
Place preview	25	RedBoard	134
Placing		Redraw command	24
bus entries	77	Reference prefix	101
buses	75	Remove Sheet command	117
components	7, 64	Reorienting objects	31
dots	67	Replacing components	20, 65
general procedure	29	Resizing	
graphics	15, 85	general procedure	31
header block	15	graphics	87
markers	89	labels	33, 57
scripts	70	Roaming a hierarchical design	121
sheet border	15	Root sheets	117, 118
sub-circuits	78	Rotation icon	8, 24, 31
terminals	9, 81		
text	15	<b>S</b>	
wires	9	Sample files	5, 21
Plotter output	143	Save As command	29
Plotters		Save Default Template Command	46
pen colours	143	Save Design command	29
Postscript	144	Saving files	29
Power nets	124	SCRIPT keyword	74
Power pins	66, 113		

Scripts	70	Symbols	
placing	70	definition	92
SDF	123, 133	editing	96
Search & Tag Commands	58	placing	87
Sections	29	tutorial	20
Send to Back command	40	System properties	47
Set Sheet Sizes command	37		
Sheet global net properties	149	<b>T</b>	
Sheet names		Tagging	
in hierarchical design	122	tutorial	8
Sheet properties	49, 119	with mouse	30
Sheets		with Search & Tag commands	58
adding, removing	117	Tango (netlist format)	134
definition	118	Tapping a bus	77
name and title	16, 117	Templates	46
ordering in design	117	Terminals	
placing border	15, 37	creating	93
selecting sizes	37	definition	80
Shift Pan	5	editing	82
Shift Zoom	6	in netlist generation	123
Shift-Pan	24	placing	9, 81
Shift-Zoom	25	properties	83
Simulation	2, 134	Text	
Single element device	108	graphic	87
Single element parts	96	hiding	48
Snapping	27	placing	15
SPICE	3	scripts	70
SPICE (netlist format)	134	Text Styles	
SPICE keyword	75	See Styles	41
SPICE-AGE (netlist format)	134	Toolbars	23
STRAT net property	148	TrueType Fonts	43
Strategies	148	TYPE property	99
Styes			
Component creation	97	<b>U</b>	
Styles		Units	26
Benefits Of	42	Untagging	30
Component creation	13	User properties	48
Editing Global Styles	42		
Editing Local Styles	44	<b>V</b>	
Introduction	41	Valid (netlist format)	134
Symbol creation	92, 93, 94, 95	Value annotation	37
Tutorial	42	Variable zoom	25
Sub-Circuits		Vector Font	43
editing	80	Verify packaging	148
placing	78	Visual Packaging Tool	105
use in hierarchical design	119	VSM	2
Symbol libraries	92		



file types	28
gadgets	85
Inter-element net names	126
models	73
netlist generation for	132
parameter mapping	73
Vutrax (netlist format)	134

**W**

Wire Auto Router	30, 35
Wire labels	
deleting	68
in netlist generation	123
properties	69

Wire Labels	
Editing	68
placing and editing	67
Wires	
and block move	34
auto repeat placement	35
dragging	9, 36
routing	9, 34
WMF file	144

**Z**

Zoom	25
Zoom to Child command	22, 121

