OrCAD®

Electronic Design Automation Tools

Schematic Design Tools 386+

Reference Guide

Copyright © 1993, 1994 OrCAD, Inc. All rights reserved.

No part of this publication may be reproduced, translated into another language, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written consent of OrCAD, Inc.

Every precaution has been taken in the preparation of this publication. OrCAD assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

OrCAD® is a registered trademark of OrCAD, Inc.

IBM® is a registered trademark of International Business Machines Corporation.

HP-GL® is a registered trademark of Hewlett-Packard Company.

VersaCad® is a registered trademark of VersaCad Corporation.

Postscript® is a registered trademark of Adobe Systems Incorporated.

All other brand and product names mentioned herein are used for identification purposes only, and are trademarks or registered trademarks of their respective holders.

Eighth Edition 1 November 94

OrCAD®

9300 SW Nimbus Avenue
Beaverton, Oregon 97008-7137
U.S.A.

General Product Information & Non-Technical Customer Support	(503) 671-9500
Fax	(503) 671-9501
Technical Support Hotline	(503) 671-9400
24-Hour Bulletin Board System	(503) 671-9401

Preface xxvii
Tools and tool sets xxvii
Editors xxviii
Processors xxviii
Librarians xxix
Reporters xxix
Transfers xxix
User buttons xxx
Configuration screens xxxi
Prefix/Wildcard entry boxes xxxi
List boxes xxxi
Source and Destination entry boxes xxxi
Mouse techniques xxxii
Left and right mouse buttons xxxii
Keyboard equivalents xxxiii
"Enter" and "Type" xxxiii
About this guide xxxiv
Conventions xxxiv

Part I: Configuration 1

Chapter 1: Configure Schematic Tools 3
Minimum configuration 4
Display the Configure Schematic Design Tools screen 5
Driver Options 6
Printer/Plotter Output Options 11
Library Options 12
Name Table Location and Symbolic Data Location 15
Initial memory allocated 17
Worksheet Options 18
Macro Options 22
Hierarchy Options 24
Color and Pen Plotter Table 25
Template Table 29
Key Fields 37
Check Electrical Rules matrix 52

Part II: Editors 55

Chapter 2: Draft 57
Execution 57
Local Configuration 58
Command reference 61
Selecting commands 61
Locating commands 61
AGAIN 64
BLOCK 65
BLOCK Move 66
BLOCK Drag 67
BLOCK Fixup 67
BLOCK Get 68
BLOCK Save 69
BLOCK Import 70
BLOCK Export 71
BLOCK ASCII Import 72
BLOCK Text Export 73
BLOCK Copy 74
CONDITIONS 75
DELETE 76
DELETE Object 76
DELETE Block 77
DELETE Undo 77
EDIT 78
Editing techniques 78
Editing labels 79
Editing module ports 80
Editing power objects 81
Editing sheet symbols 82
Editing parts 85
Editing the title block 93
Editing stimulus objects 95
Editing trace objects 95
Editing vector objects 95
Editing layout objects 95

Chapter 2: Draft (continued)
FIND 96
GET 98
Getting a part by entering a part suffix 99
The outline symbol 99
Rotating and placing parts 100
HARDCOPY 102
HARDCOPY Print Worksheet 102
HARDCOPY Block Print 103
HARDCOPY Destination 103
HARDCOPY File Mode 104
HARDCOPY Width of Paper 104
HARDCOPY Scale 105
INQUIRE 105
JUMP 106
JUMP A, B, C, D, E, F, G, H Tag 106
JUMP Reference 106
JUMP X-Location 107
JUMP Y-Location 108
LIBRARY 109
LIBRARY Directory 109
LIBRARY Browse 110
MACRO 111
MACRO Capture 112
Valid macro keys 112
Nesting macros 114
Pause 114
Debugging macros 114
Initial macros 115
MACRO Delete 115
MACRO Initialize 115
MACRO List 115
MACRO Read 115
MACRO Write 116

Chapter 2: Draft (continued)

Using macros 116
Calling a macro 116
Macro buffer 116
Macro text files 117
Macro syntax 117
Macro comments 119
Middle mouse button macros 120
Assignment macros 120
Individual macros 121
Creating efficient macros 122
PLACE 123
PLACE Wire 123
PLACE Bus 125
PLACE Junction 126
PLACE Entry (Bus) 127
PLACE Label 128
PLACE Module Port 131
PLACE Power 133
PLACE Sheet 135
PLACE Text 137
PLACE Dashed Line 139
PLACE Trace Name 140
PLACE Vector 141
PLACE Stimulus 142
PLACE NoConnect 144
PLACE Layout 145
QUIT 146
QUIT Enter Sheet 146
QUIT Leave Sheet 146
QUIT Update File 147
QUIT Write to File 147
QUIT Initialize 147
QUIT Suspend to System 148
QUIT Abandon Edits 148
QUIT Run User Commands 149

Chapter 2: Draft (continued)
REPEAT ... 150
SET ... 151
SET Auto Pan ... 152
SET Backup File ... 152
SET Drag Buses ... 153
SET Error Bell ... 153
SET Left Button ... 153
SET Macro Prompts ... 154
SET Orthogonal ... 154
SET Show Pins ... 155
SET Title Block ... 155
SET Worksheet Size ... 156
SET X,Y Display ... 156
SET Grid Parameters ... 157
SET Repeat Parameters ... 158
SET Visible Lettering ... 159
SET Cursor Style ... 160
TAG ... 161
ZOOM ... 162
ZOOM Center ... 162
ZOOM In ... 162
ZOOM Out ... 162
ZOOM Select ... 162

Chapter 3: Guidelines for creating designs ... 163
Label names ... 163
Intersheet connections ... 168
Splitting buses ... 170
Bus mapping ... 171
Named properties ... 172
Handling and isolating power ... 173
Connecting power objects with different names ... 175
Isolating power and ground ... 176
Handling power in a hierarchy ... 177
Handling physical connectors ... 181

Chapter 4: Edit File 183
Execution 183

Chapter 5: View Reference 185
Execution 185

Part III: Processors 187

Chapter 6: Annotate Schematic 189
Execution 189
Key Fields 190
Before annotation and after annotation 190
Local Configuration 192

Chapter 7: Back Annotate 195
Execution 195
Was/Is file format 195
Running Back Annotate 195
Local Configuration 196

Chapter 8: Cleanup Schematic 199
Execution 199
Local Configuration 201

Chapter 9: Check Design Integrity 203
Execution 203
Local Configuration 204
Configure CLEANUP, CROSSREF, and ERC 204

Chapter 10: Creating a netlist 205
Incremental design 205
Compile: INET 206
Link: ILINK 206
Format: IFORM or HFORM 206
Creating a netlist 206
The compiler: INET 208
The incremental connectivity database 208
Creating a multiple-sheet design using the ILINK command 209

Chapter 10: Creating a netlist (continued)
Caveats 210
The linker: ILINK 211
Intermediate netlist structure 212
The flat formatter: IFORM 213
Unnamed nets 213
The hierarchical formatter: HFORM 214

Chapter 11: Create Netlist 215
Linked format 215
Creating linked and flattened netlists 216
Execution 217
Local Configuration of Create Netlist 218
Configure INET 219
Configure ILINK 225
Configure IFORM 227

Chapter 12: Create Hierarchical Netlist 231
Hierarchical format 231
Execution 232
Local Configuration of Create Hierarchical Netlist 233
Configure INET 233
Configure HFORM 233

Chapter 13: Select Field View 235
Execution 235
Local Configuration 235

Chapter 14: Update Field Contents 239
Before you run Update Field Contents 240
Configuring Key Fields 240
Configuring Update Field Contents 240
Creating an update file 240
Execution 243
During the updating process 243
After the updating process 243
Local Configuration 244

Chapter 15: Decompile Schematic 249
Execution 249
Local Configuration 250
Output file format 253

Chapter 16: Compile Schematic 257
Execution 257
Local Configuration 258

Part IV: Librarians 261

Chapter 17: About libraries 263
Library files 263
Library source file 264
Compiled library file 264
List parts in a library 265
Creating library files 265
Edit Library 266
Text editor 266
Components of a library part 268
Body 268
Block 268
Graphic 269
IEEE 269
Pins 270
Pin type 270
Pin shape 270
Pin number 270
Pin name 271
Names 271
Sheetpath designator 271
Reference designator 271

Chapter 18: List Library 273
Execution 273
Local Configuration 274

Chapter 19: Archive Parts in Schematic 277
Execution 277
Local Configuration 278
Configure LIBARCH 279
Configure COMPOSER 282

Chapter 20: Edit Library 283
About Edit Library 283
Editing a part with Edit Library 284
Editing existing parts to create new parts 285
Execution 286
Local Configuration 286
Command reference 289
Selecting commands 289
AGAIN 292
BODY 293
BODY Kind of Part? 294
BODY Kind of Part? Block 295
BODY Kind of Part? Graphic 295
BODY Kind of Part? IEEE 296
BODY command reference 297
BODY <Block> commands 297
Body <Block> Size of Body 297
Body <Block> Kind of Part 297
BODY <Graphic> commands 298
BODY <Graphic> Line 298
BODY <Graphic> Circle 299
BODY <Graphic> Arc 300
BODY <Graphic> Text 301
BODY <Graphic> IEEE Symbol 302
BODY <Graphic> Fill 304
BODY <Graphic> Delete 304
BODY <Graphic> Erase Body 305
BODY <Graphic> Size of Body 305
BODY <Graphic> Kind of Part 305
BODY <IEEE> commands 306
BODY <IEEE> Line 306

Chapter 20: Edit Library (continued)

BODY <IEEE> Circle 307
BODY <IEEE> Text 308
BODY <IEEE> IEEE Symbol 309
BODY <IEEE> Delete 310
BODY <IEEE> Erase Body 310
BODY <IEEE> Size of Body 310
BODY <IEEE> Kind of Part 311
CONDITIONS 312
Library 313
Current Part 313
EXPORT 315
GET PART 317
Getting a part by entering a part suffix 317
IMPORT 319
JUMP 320
JUMP A, B, C, D, E, F, G, H Tag 320
JUMP X-Location 320
JUMP Y-Location 321
LIBRARY 322
LIBRARY Update Current 322
LIBRARY List Directory 323
LIBRARY Browse 324
LIBRARY Delete Part 325
LIBRARY Prefix 326
About prefix definitions 326
0 through 16 327
Prefix and Short Prefix 327
MACRO 328
Initial Macro 328
NAME 329
NAME Add 330
NAME Delete 330
NAME Edit 330
NAME Prefix 330
ORIGIN 331

Chapter 20: Edit Library (continued)

PIN 332
PIN Add 332
PIN Delete 332
PIN Name 332
PIN Pin-number 332
PIN Type 333
PIN Shape 334
PIN Move 334
PIN Repeat 335
QUIT 336
QUIT Update File 336
QUIT Write to File 337
QUIT Initialize 337
QUIT Suspend to System 338
QUIT Abandon Edits 338
REFERENCE 339
SET 340
SET Auto Pan 340
SET Backup File 341
SET Error Bell 341
SET Left Button 342
SET Macro Prompts 342
SET Power Pins Visible 343
SET Show Body Outline 343
SET Visible Grid Dots 343
SET Repeat Parameters 344
SET Cursor Style 344
TAG 345
ZOOM 346
ZOOM Center 346
ZOOM In 346
ZOOM Out 346
ZOOM Select 346

Chapter 21: Decompile Library 347
Execution 347
Local Configuration 348

Chapter 22: Creating a library source file with a text editor 351
Library source file 351
Block part definitions 351
Graphic part definitions 351
IEEE part definitions 351
Prefix Definition 352
Use of the prefix definition 352
Constructing a prefix definition 353
Part definition 355
Three types of part definitions 355
Components of a part definition 355
Defining a block symbol 358
Part name string 359
Sheetpath keyword 359
Reference keyword 360
Grid unit size and parts/package 361
Pin definitions 362
Pin type 363
Selectively displaying pins 364
Pin-grid array 366
Defining a graphic symbol 369
Defining a vector 369
Graphic symbol considerations 370
Converted form graphic symbol 371
Defining an IEEE symbol 373
Part name string 373
Size and type definitions 375
Pin definitions 375
Vector definitions 375
Defining a vector 376
IEEE standards 376
Pin placement 377
Building the IEEE body outline 377

Chapter 22: Creating a library source file with a text editor (continued)
IEEE Vector Objects 377
Placing ACTIVE_LOWs 378

Chapter 23: Symbol Description Language 379
Syntax diagram 379
Identifiers 380
Tokens 380
How syntax is described in this chapter 382
Prefix definition 384
Part definition 386
Pin definition 389
Vector definition 395
Converted form definition 397

Chapter 24: Compile Library 399
Creating a custom library with Compile Library 399
Execution 400
Local Configuration 400

Part V: Reporters 403

Chapter 25: Check Electrical Rules 405
Checking for electrical errors 405
Find and repair errors 406
Discard error markers 406
Execution 407
Typical messages and resolutions 407
How to specify conditions to check 407
Local Configuration 412

Chapter 26: Cross Reference Parts 415
Execution 415
Sample Output 416
Local Configuration 416
Processing Options 418

Chapter 27: Convert Plot to IGES 421
Plot the file 421
Execution 421
Sample output 422
Local Configuration 423

Chapter 28: Plot Schematic 425
Execution 426
Suppressing the title block, border, and text 426
Sample output 427
Local Configuration 428

Chapter 29: Print Schematic 437
Execution 438
Sample output 438
Output size 439
Local Configuration 440

Chapter 30: Create Bill of Materials 443
Execution 443
Sample output 444
Key fields 445
Local Configuration 446

Chapter 31: Show Design Structure 451
Execution 451
Sample output 451
Local Configuration 452

Part VI: Transfers 453

Chapter 32: To PLD 455
Execution 456
Running To PLD 456
Local Configuration of To PLD 457
Configure FLDSTUFF 458
Configure ANNOTATE 458
Local Configuration of EXTRACT 459
About EXTRACT 461
Key fields 461
Unified documentation 461
Make a custom symbol 462
Defining the PLD's internal logic 462
Select a device 463
Record part type and value on the schematic 463

Chapter 33: To Digital Simulation 467
Execution 468
Local Configuration of To Digital Simulation 469
Configure ANNOTATE 470
Configure INET 470
Configure IBUILD 471
Configure ASCTOVST 472

Chapter 34: To Layout 475
Execution 476
Local Configuration of To Layout 476
Configure INET 477
Configure ILINK 477
Configure IFORM 477

Chapter 35: To Main 479
Execution 479

Appendixes 481

Appendix A: Command line controls 483
- Syntax 484
- AIM32 485
- AEX32 485
- ANNOTATE 486
- BACKANNO 487
- CLEANUP 487
- COMPOSER 488
- CROSSREF 488
- DECOMP 488
- DRAFT 489
- ERC 489
- EXTRACT 490
- FLDATTRB 490
- FLDSTUFF 491
- HFORM 492
- IFORM 493
- IFORM 494
- ILINK 495
- INET 496
- INET 497
- LIBARCH 498
- LIBEDIT 499
- LIBLIST 499
- PARTLIST 500
- PARTLIST 501
- PLOTALL 502
- PRINTALL 503
- SIMPLE 504
- TREELIST 504

Appendix B: Netlist formats 505
Usage 505
Types of netlist format files 506
Configuring for netlists 507
Flat netlists 507
Example schematics 507
Algorex (FALGOREX.EXE) 514
Allegro (FALLEGRO.EXE) 515
AlteraADF (FALTERAA.EXE) 516
AppliconBRAVO (FAPPLBRA.EXE) 520
AppliconLEAP (FAPPLLEA.EXE) 521
Cadnetix (FCADNETI.EXE) 522
Calay (FCALAY.EXE) 524
Calay (FCALAY90.EXE) 526
Case (FCASE.EXE) 527
CBDS (FCBDS.EXE) 529
ComputerVision (FCOMPVIS.EXE) 530
DUMP (FDUMP.EXE) 531
EDIF (FEDIF.EXE) 532
EEDesigner (FEEDESIG.EXE) 538
FutureNet (FFUTURE.EXE) 539
HiLo (FHILO.EXE) 544
IntelADF (FINTELAD.EXE) 546
Intergraph (FINTERGR.EXE) 549
Mentor (FMENTOR.EXE) 550
MultiWire (FMULTIWI.EXE) 552
OrCAD Hardware Description Language (FOHDLNET.EXE) 554
OrCAD/PCB II (FPCBII.EXE) 557
OrCAD Programmable Logic Design Tools (FPLDNET.EXE) 559
OrCAD Digital Simulation Tools Model (FVSTMODE.EXE) 561
PADS 2000 (FPADS2K.EXE) 563
PADS ASCII (FPADSASC.EXE) 565
PADS ASCII (FPADSASØ.EXE) 567
PADS-PCB (FPADSPCB.EXE) 568
PCAD (FPCAD.EXE) 570
PCADnlt (FPCADNLT.EXE) 572

Appendix B: Netlist formats (continued)
PDUMP (FPDUMP.EXE) 574
RacalRedac (FRACALRE.EXE) 575
Scicards (FSCICARD.EXE) 577
SPICE (FSPICE.EXE) 579
Tango (FTANGO.EXE) 583
Tango1 (FTANGO1.EXE) 585
Telesis (FTELESIS.EXE) 587
Vectron (FVECTRON.EXE) 588
WireList (FWIRELIS.EXE) 590
Hierarchical netlists 592
Example Schematics 592
EDIF (HEDIF.EXE) 597
HDUMP (HDUMP.EXE) 603
SPICE (HSPICE.EXE) 603

Appendix C: Interpreting connectivity databases 607
Overview of connectivity databases 607
About the incremental connectivity database 608
Typographical conventions 609
Terminology 610
Token 610
White space 610
Quoted token 610
String 610
Delimiter 611
Command 611
Character 611
Number 611
Sub-part code 611
Statement 612
Parameter 612
.INF format specification 612
Sample .INF file 638

Appendix D: Creating a custom netlist format 641
About netlist formats 642
Flat formats 642
Hierarchical formats 642
Part and net orientations 642
OrCAD-supplied formats 642
Customer-contributed formats 642
Directory structure 643
How to create a new format 645
Compiling and linking a format file 646
MKNETFMT.BAT command syntax 646
How MKNETFMT.BAT works 648
If you have problems 653
Environment variables 653
Compiler and linker messages 653
Combined library file 653
About format files 653
File names 653
Language 653
Functions 654
Standard symbols 654
User-defined symbols 654
Flat format 655
Hierarchical format 658
Required functions 659
Data functions 660
Data structures 660
Instance files 662
Traversal functions 663
Pipe file functions 663
General functions 664
Switches 665
Standard symbol reference 666
Type definition reference 672
Function reference 676
Error and warning messages 694

Appendix E: Plotter information 697
Plotter cable wiring 698
Plotter problems 700
General plotter tips 702
Plot language for HP plotters and printers 703
HI plotters 704
Calcomp plotters 705
Notes on plotter and printer drivers 708
HP.DRV 708
HPLASERx.DRV 709
DXF.DRV 709
PostScript plotter drivers 710
Encapsulated PostScript 710
Other PostScript 711

Appendix F: Files and file extensions 713
Design files 713
Other files 718
#ESP_OUT.TXT 718
HARDCOPY.PRN 718
ORCADESP.DAT 718
SDT.BCF 718
SDT.CFG 718
Reference files 718
Tutorial files 719
Update file 719
Was/Is file 720
Netlist format files 720

Appendix G: Converting designs, worksheets, and libraries 721
Converting designs 721
CONVERT command syntax 722
Converting individual worksheets 726
16TO32 and 32TO16 command syntax 726
Converting libraries 727
DECOMP16 and COMP16 728
Converting sheetpath parts 728

Appendix G: Converting designs, worksheets, and libraries (continued)
Upgrading from SDT IV 729
Custom netlist format files 730
Custom libraries 732
Installing OrCAD/PCB II 733

Glossary 735

Index 743

OrCAD's **Schematic Design Tools 386+** operates within the OrCAD ESP design environment. This environment provides many features that make it easier to access and use OrCAD's electronic design automation (EDA) tool sets.

This book is a reference guide to **Schematic Design Tools 386+**, the tool set used to create schematic designs. For detailed information about the ESP design environment, see the *ESP Design Environment User's Guide*.

Tools and tool sets

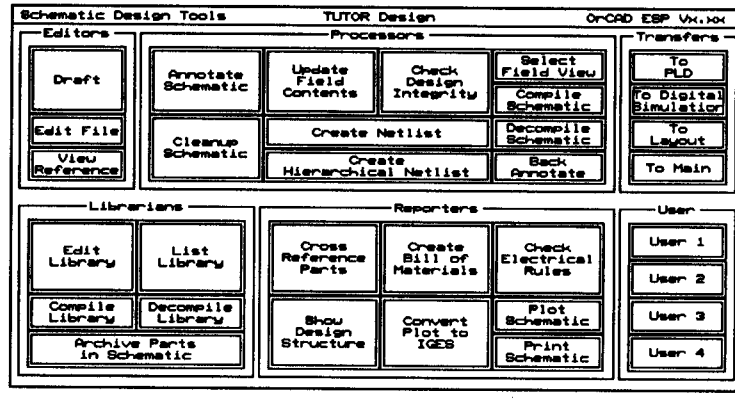
A *tool set* is a collection of tools designed to perform a set of electronic design automation tasks. There are currently four OrCAD tool sets. They are:

- **Schematic Design Tools**
- **Programmable Logic Design Tools**
- **Digital Simulation Tools**
- **Printed Circuit (PC) Board Layout Tools**

The tool sets allow you to access the same design in different ways.

Buttons for the OrCAD design tool sets appear on the main ESP design environment screen, even if you have only one tool set installed on your system.

To select the **Schematic Design Tools** tool set from the main ESP design environment screen, point to the **Schematic Design Tools** button and double-click. In a moment, you will see the screen for **Schematic Design Tools 386+** as shown in the figure on the next page.



Schematic Design Tools 386+ screen.

In tool sets, tools are grouped according to function. The six categories are:

- Editors
- Processors
- Librarians
- Reporters
- Transfers
- User buttons

These functions are described in the following paragraphs.

Editors Editors modify or create some part of the design database. An example of an editor is the schematic editor, **Draft**. Another editor is **Edit File**, which uses a text editor to view reports and enter text.

Processors Processors read, modify, then rewrite the design database. For example, **Annotate Schematic** is a processor. Processors generally do not create human-readable reports, but rather create or modify database information. Processors may create data that will be used by tools outside the ESP design environment.

Librarians Librarians are tools for managing and creating library objects that can be used by all designs, not just the current design. **Edit Library** is an example of a librarian. It is used to create a new schematic symbol for a component. This component should be available in all future design work, so it is stored in the library database.

Reporters Reporters create human-readable reports, but do not modify design data in any way. For example, a reporter creates the Bill of Materials report, a list of all the components used in the design. The tools for printing and plotting are also reporters. Reporters may create reports that will be used by tools outside the ESP design environment.

Transfers Transfer tools manage the steps needed to move design information from one tool set to another. Transfers have two parts. The first updates the database used by the current tool set so that it is current and up-to-date in every respect. The second part changes to the new tool set used to view the design. The transfer tools take care of intermediate steps so that you don't have to.

For example, the **To Digital Simulation** transfer tool performs these steps:

- Annotates the reference designators in the design.
- Builds the connectivity database.
- Builds the link between the schematic and the simulator, so that simulation directives inserted in the schematic can be accessed by the simulator.
- Transfers control to **Digital Simulation Tools**.

User buttons

A user button is the most basic way in which the ESP design environment can be extended to fit your particular requirements and make your work easier and more convenient.

A user button can be set up to run any system command. You can set up a user button to run a spreadsheet program, which you can use to analyze design information. Or, you can program user buttons to run utility programs, communications programs, other graphical user interfaces and their programs—almost any program you like. You can also write batch files and program user buttons to run them.

The ESP design environment places four user buttons inside every tool set. Chapter 4 of the *ESP Design Environment User's Guide* explains how to define a user button.

Configuration screens

Schematic Design Tools 386+ and the ESP design environment have many configuration screens. Some configuration screens apply only to a specific tool. These are called *local configuration* screens. Other configuration screens—such as the **Configure Schematic Design Tools** screens—are global in nature.

Prefix/Wildcard entry boxes

Many configuration screens have a **Prefix/Wildcard** entry box. These entry boxes contain a pathname and possibly a filename with a wildcard to indicate which files to display in a list box (described below). The asterisk can be used as a wildcard in a filename. This example lists all files in the C:\ORCADESP\SDT\LIBRARY path that have a .LIB extension:

Prefix/Wildcard `C:\ORCADESP\SDT\LIBRARY\*.LIB`

List boxes

Many configuration screens have *list boxes* containing lists of files from which to choose. Be sure you know how to select a file from a list box and how to use the scroll bars to scroll the file lists. Files preceded by ".\" are found in the current design directory. Files not preceded by ".\" are found in the path given in the **Prefix/Wildcard** entry box. When you click the left mouse button on a filename in a list box, the filename automatically displays in the **Source** entry box.

Source and Destination entry boxes

Most local configuration screens have a **Source** entry box. Many also have a **Destination** entry box.

The first time you display a local configuration screen, its **Source** and **Destination** entry boxes contain—where appropriate—the design name followed by a default extension. However, you can change this to suit your needs.

If you change the filename extension in the **Source** entry box, when you select **OK** to leave the configuration screen and save the changes, the extension in the **Prefix/Wildcard** entry box also automatically changes to the same extension.

To have the ESP design environment configure a **Source** or **Destination** entry box for you, enter a "?" followed by the file extension. For example, if you enter ?.LIB, the ESP design environment changes the "?" to the name of the current design when you select **OK** to leave the configuration screen and save the changes.

Mouse techniques



You can do all your work in **Schematic Design Tools 386+** (except typing text and numbers) using the mouse.

You *point* to an object by moving the pointer until the tip of the pointer touches the object. Do this by moving the mouse.

You *click* by pointing to an object and then pressing and releasing the left mouse button once. When you click on a button, it becomes *highlighted* and a menu pops up in the upper left corner of the screen.

In this guide, the words "click," "highlight," and "select" all mean the same thing. In every case the action you take is the same: position the pointer, press the left mouse button, and quickly release it.

You *double-click* by first pointing to an object and then clicking the left mouse button twice. Don't move the mouse while you double-click.

Left and right mouse buttons

- Clicking the left mouse button is the same as pressing the <Enter> key. In this guide, when you are instructed to "press <Enter>," you can use either the keyboard or the mouse, whichever you prefer.
- Clicking the right mouse button is the same as pressing the <Esc> key. In this guide, when you are instructed to "press <Esc>," you can use either the keyboard or the mouse, whichever you prefer.

Keyboard equivalents



Many of the explanations and instructions in this book use the mouse terminology explained on the previous page. If you prefer to use the keyboard, however, there are keyboard equivalents to nearly every mouse operation. Instead of moving the mouse to move the pointer from button to button, you can:

- Press <Tab> to move from one tool category to the next.
- Press <Space bar> to move from button to button within a category.
- Press <Shift><Tab> to move the pointer backwards to the previous category.
- Press <Enter> to select the button the pointer rests on.
- Press <Home> to move the pointer to the first button in the area nearest the upper-left corner of the screen or, on configuration screens, to the **OK** button.
- Press <Esc> to close a menu without selecting any of the commands or to cancel any changes to a text entry box.
- Press <End> to move the pointer to the first user button or, on configuration screens, to the last button in the last area.
- Press <Page Up> and <Page Down> to pan up and down on configuration screens.
- In **Draft** and **Edit Library**, press <Ctrl> and an arrow key to move the pointer five grid units in the direction of the arrow key, or press <Alt> and an arrow key to move the pointer to the edge of the window.

"Enter" and "Type"

The instructions in this guide use the terms "enter" and "type" to mean two different things. When the instructions tell you to enter something, you press the appropriate keys and end by pressing <Enter>. When the instructions tell you to type something, you press the appropriate keys, but do *not* press <Enter>.

About this guide

This guide is organized according to function. The basic parts of this guide are:

- Part I: Configuration
- Part II: Editors
- Part III: Processors
- Part IV: Librarians
- Part V: Reporters
- Part VI: Transfers

Each tool is described in a chapter in the appropriate part of this guide. For example, to find information about **Draft**, look in *Part II: Editors*.

Conventions

The notation conventions used in this guide are as follows:

BOLD CAPS	Used for main menu commands.
Bold	Used for other commands.
`Courier bold`	Used for text you enter.
Italics	Used for references to other sections or chapters of this guide, other guides, or other publications.
"Prompt"	Quotation marks show program prompts.
<B>	Brackets <> show a key (or keys) that you press. For example, ❖ <Esc> means to press the escape key. ❖ <Ctrl><S> means to press the control key, and while holding it down, press the "S" key. ❖ <S> <T> (notice the space between these two characters) means to press the "S" key and let it go, then press the "T" key.

Boxes The shadow box shown below shows a program or system prompt. Any bold type following the prompt shows text that you enter. For example:

```
Abandon edits? Y
```

This kind of shadow box shows a program menu.

```
Hardcopy
Destination
File
```

Entry boxes like the one below can be empty or contain information you can edit. They appear on configuration screens.

Wildcard `*.*`

△ *NOTE: Notes contain important reminders or hints.*

▲ *CAUTION: Cautions contain information about preventing damage to equipment, software, or data.*

PART I: CONFIGURATION

When you install **Schematic Design Tools 386+** on your system's hard disk, it is configured and ready to run.

Part I explains how to customize **Schematic Design Tools 386+** configuration.

Chapter 1: *Configure Schematic Tools* describes how to modify:

- Driver Options
- Printer and Plotter Output Options
- Library Options
- Worksheet Options
- Macro options for both **Draft** and **Edit Library**
- Hierarchy Options
- Color and Pen Plotter Table
- Template Table
- Key Fields
- Matrix for **Check Electrical Rules**



Configure Schematic Tools

OrCAD's ESP design environment has three types of configuration, all of which customize and save information used to run OrCAD tools and tool sets.

- *ESP configuration* defines driver options, the text editor, startup design, and monitor display colors. Although the ESP design environment is already configured when installed, you can change the configuration whenever you want to change ESP design environment parameters.

  The *ESP Design Environment User's Guide* provides detailed instructions for customizing the ESP design environment.

- *Tool set configuration* defines driver, library, work area, and macro options, plus tool set specific monitor display colors and display drivers. Tool set configuration applies to all tools in a tool set and can be accessed from every button in the tool set except transfers and user buttons. It has a default configuration when installed but can also be configured anytime you want to change the tool set parameters.

  The rest of this chapter provides detailed instructions for customizing the **Schematic Design Tools 386+** configuration.

- *Local configuration* determines input and output files plus special processing options for a particular tool. If a tool runs several processes, each process can be locally configured.

  Local configuration is set up when the design is created, with input and output filenames defaulting to the design name in most cases.

  The chapter that describes a tool also provides instructions for customizing its local configuration.

Minimum configuration

To use **Schematic Design Tools 386+**, you must have an IBM PC or compatible computer with:

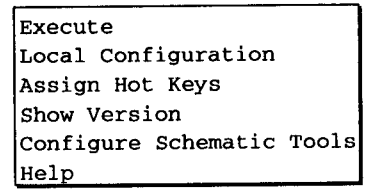
- An 80386 or 80486 microprocessor
- At least two megabytes of free RAM available to **Schematic Design Tools 386+** (four to eight megabytes are recommended).
- A hard disk with seven to ten megabytes of free storage space.

To find out how much free memory and free disk space your system has, use the DOS command CHKDSK. See your DOS manual for more information about this command.

Display the Configure Schematic Design Tools screen

With the **Schematic Design Tools** screen displayed, select any of the editors, processors, librarians, or reporters buttons. For example, select **Draft**.

This menu displays at the top of the screen. Select **Configure Schematic Tools**. The **Configure Schematic Design Tools** screen displays. Each area on this screen is shown in the sections that follow.



You can move the pointer down until it touches the lower edge of the display, and the display pans down to show more options. When you get to the bottom, the display only pans up.

If you prefer to use keyboard commands, press <Page Down> to move the window down part of a screen at a time, and <Page Up> to go up again. Press <End> to go to the bottom of the configuration screen, and <Home> to return to the top again.

In various places within the configuration screen, there are boxes or windows in which lists (usually of files) display.

Using the scroll buttons to the right of each list box, you can move these lists up and down.

When you finish making changes, select **OK** to save your changes and return to the **Schematic Design Tools** screen.

If you do not want to save your changes, select **Cancel** to return to the **Schematic Design Tools** screen.

Driver Options

The **Driver Options** (figure 1-1) area defines the driver prefix, display driver, printer driver, and plotter driver. These are described on the following pages.

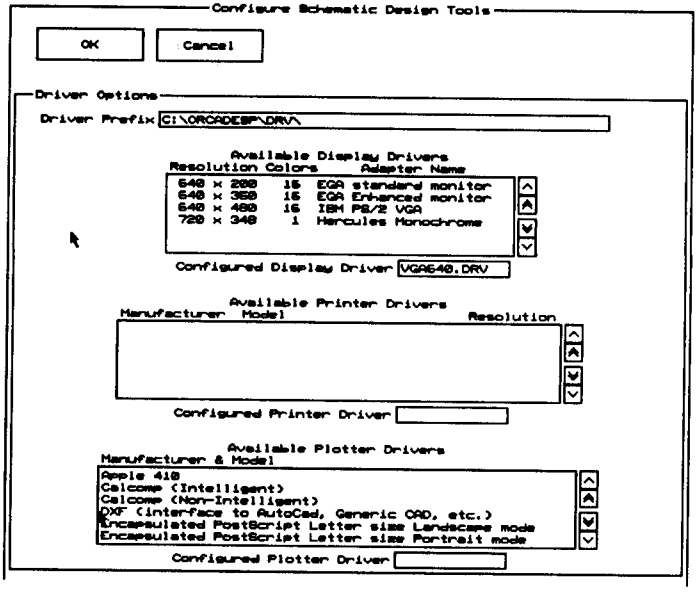


*Figure 1-1. **Driver Options** area of the **Configure Schematic Design Tools** screen.*

Driver Prefix

The **Driver Prefix** is the directory path or disk drive where **Schematic Design Tools 386+** finds and loads the display, printer, and plotter drivers.

The driver prefix is set during the installation process and does not need to change unless you move drivers to a different directory or create custom drivers in another directory.

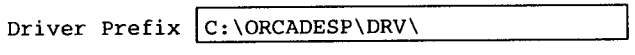
To define the driver prefix, place the cursor in the **Driver Prefix** entry box and enter the pathname of the directory containing your device drivers.

△ *NOTE: Only the drivers that are recognized by name appear in the list box. Custom drivers do not appear and need to be typed into the entry box.*

Once you enter a driver prefix, all of the drivers in that directory display in the appropriate list box: **Available Display Drivers, Available Printer Drivers,** or **Available Plotter Drivers.** Each of these list boxes is described in the sections that follow.

Example

The **Driver Prefix** is created during the installation process. If you installed **Schematic Design Tools 386+** on your C: drive, the prefix is:



This tells **Schematic Design Tools 386+** to look for the drivers in the ORCADESP\DRV directory on the C: drive.

Available Display Drivers

The **Available Display Drivers** area of the screen is where you choose which graphics display driver to load.

A list box (figure 1-2) lists the different display drivers that are available in the directory path specified in the **Driver Prefix** entry box.

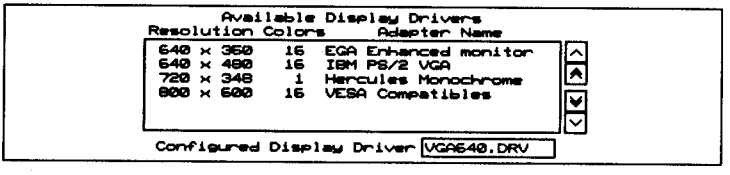


*Figure 1-2. **Available Display Drivers** list box.*

Select the driver that is appropriate for your system by clicking on it. To see other drivers not displayed in the list box, use the scroll buttons at the right of the list box to scroll the list of drivers up and down.

Once you select a display driver, its filename displays in the **Configured Display Driver** entry box.

You do not have to select a display driver from the **Available Display Drivers** list box. Instead, simply click in the **Configured Display Driver** entry box and enter the driver name. However, be sure that the driver is in the directory displayed in the **Driver Prefix** entry box.

△ *NOTE: Only the drivers that are recognized by name appear in the list box. Custom drivers do not appear, and need to be typed into the entry box.*

Example

If you select the **EGA Enhanced monitor** from the drivers displayed in figure 1-2, the following displays:

Configured Display Driver EGA16E.DRV

Available Printer Drivers

The **Available Printer Drivers** area of the screen is where you choose which printer driver to load.

A list box (figure 1-3) lists the printer drivers available in the directory path specified in the **Driver Prefix** entry box.

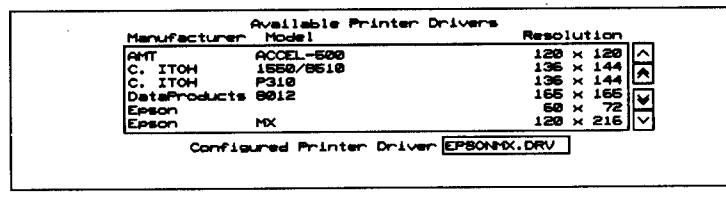


*Figure 1-3. **Available Printer Drivers** list box.*

Select the driver for your printer by clicking on it. If you need to see other drivers not displayed in the window, use the scroll buttons at the right of the list box to scroll the list of drivers up and down.

Once you select a printer driver, its filename displays in the **Configured Printer Driver** entry box.

You do not have to select a printer driver from the **Available Printer Drivers** list box. Instead, simply click in the **Configured Printer Driver** entry box and enter the driver name. However, be sure that the driver is in the directory displayed in the **Driver Prefix** entry box.

△ *NOTE: Only the drivers that are recognized by **Schematic Design Tools 386+** appear in the list box. Custom drivers do not appear, and need to be typed into the **Configured Printer Driver** entry box.*

Example

If you select the Epson MX printer from the drivers displayed in figure 1-3, the following displays:

```
Configured Printer Driver [EPSON.DRV                ]
```

Available Plotter Drivers

The **Available Plotter Drivers** area of the screen is where you choose which plotter driver to load.

A list box (figure 1-4) lists the different plotter drivers that are available in the directory path specified in the **Driver Prefix** entry box.

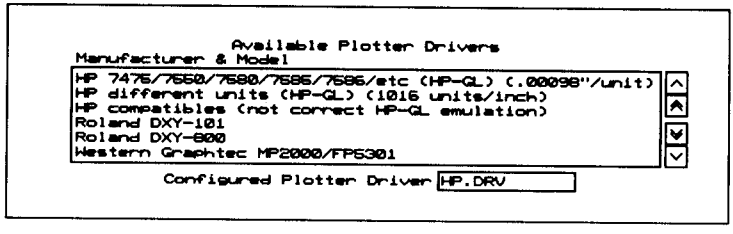


*Figure 1-4. **Available Plotter Drivers** list box.*

Select the driver for your plotter by clicking on it. If you need to see other drivers not displayed in the list box, use the scroll buttons at the right of the list box to scroll the list of drivers up and down.

Once you select a plotter driver, its filename displays in the **Configured Plotter Driver** entry box.

You do not have to select a plotter driver from the **Available Plotter Drivers** list box. Instead you can enter the name of a driver in the **Configured Plotter Driver** entry box by simply typing it and pressing <Enter>.

△ *NOTE: Only the drivers that are recognized by **Schematic Design Tools 386+** appear in the list box. Custom drivers do not appear and need to be typed into the **Configured Plotter Driver** entry box.*

Example

If you select the first HP driver from the drivers displayed in figure 1-4, the following displays:

Configured Plotter Driver [HP.DRV]

For additional information about how to plot a file, see *Chapter 28: Plot Schematic.*

Printer/Plotter Output Options

The **Printer/Plotter Output Options** area (figure 1-5) defines the ports to which your printer and plotter are connected. Select the desired output port for your printer or plotter or both.

If you select a parallel port (LPT1:, LPT2:, or LPT3:), the baud rate, parity, data bits, and stop bits options are dimmed. You do not need to define these communications parameters for parallel ports.

If you select a serial port (COM1:, COM2:, COM3:, or COM4:), the baud rate, parity, data bits, and stop bits options become available. Click on the desired settings for your printer or plotter or both. These settings are determined by the needs of your printer or plotter and the serial port to which it is connected. If necessary, see your printer or plotter documentation.

△ *NOTE: The BIOS on some computers does not support COM3 and COM4. If your computer's BIOS does not support COM3 and COM4, you cannot use these ports.*

Unavailable options

On monochrome screens and in OrCAD manuals, options that are not available are shown with a line through them. On color monitors, the options are dimmed.

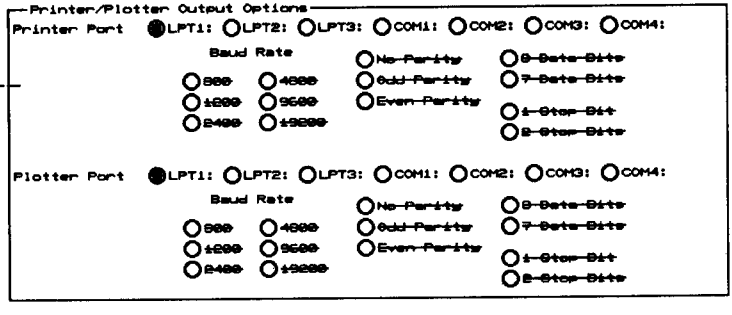


*Figure 1-5. **Printer/Plotter Output Options** area of the **Configure Schematic Design Tools** screen.*

Library Options

The **Library Options** area (figure 1-6) defines the prefix **Schematic Design Tools 386+** uses to find libraries, and the libraries that load when tools run. It also specifies the location of the reference library's name table and symbolic data table; and the active library size.

Draft and other tools load the libraries listed in the **Configured Libraries** list box when they run.

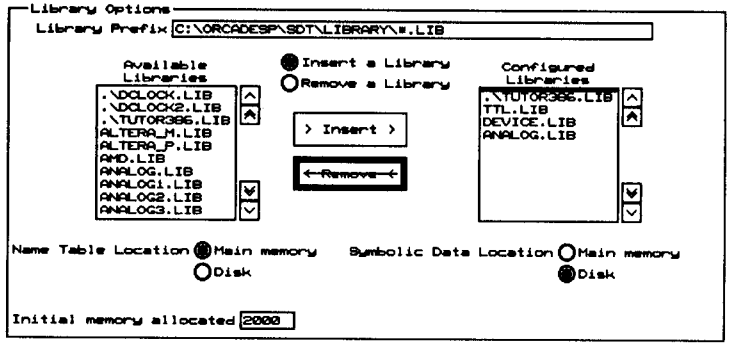


*Figure 1-6. **Library Options** area of the **Configure Schematic Design Tools** screen.*

Draft loads and maintains libraries in the order in which they are listed in the **Configured Libraries** list box. This is important when retrieving parts while creating schematics. When you ask **Draft** to get a certain part name, it searches the libraries in the order they are listed in the **Configured Libraries** window and gets the first part it finds with a matching name.

Duplicate part names can cause problems when you get parts in **Draft**. Note that OrCAD-supplied parts libraries do not have parts with duplicate names in the *same* library; however, some libraries, such as the PSPICE.LIB and SPICE.LIB libraries, do contain parts that have the same names as parts in the other library. In these cases, the order in which libraries load can be very important.

If you create your own version of an OrCAD-supplied part, save it in a custom library you create yourself. Then, configure **Schematic Design Tools 386+** to load this library before any OrCAD libraries by placing it first in the **Configured Libraries** window. Using custom libraries also makes sure your custom parts are not overwritten if OrCAD updates the original library.

To create a custom library, use **Edit Library's QUIT Write to File** command. For instructions on how to change the order of the configured libraries list, see *Changing the library order* in this chapter.

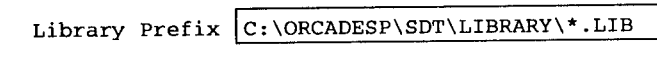
Library Prefix

The **Library Prefix** is the disk drive or directory path where **Schematic Design Tools 386+** finds and loads libraries.

To define the library prefix, place the cursor in the **Library Prefix** entry box and enter the pathname of the directory containing your libraries and a wildcard with a specific extension. Once you enter a library prefix, all of the libraries in that directory display in the **Available Libraries** list box.

Example

The example below tells **Schematic Design Tools 386+** to look for libraries with the .LIB extension in the ORCADESP\SDT\LIBRARY subdirectory on the C: hard disk.



Available Libraries and Configured Libraries

The **Available Libraries** list box displays all of the libraries available in the directory specified in the **Library Prefix** entry box. The **Configured Libraries** list box displays all of the libraries configured to load when you run **Draft** or other schematic design tool.

Inserting a library

To add a library in the **Configured Libraries** list box, select the **Insert a Library** option. The **Insert** option becomes highlighted and available for use.

Select the library that you would like to add to the **Configured Libraries** list by clicking on it. If you need to see other libraries that aren't displayed in the window, use the scroll buttons at the right of the window to scroll the list of libraries up and down.

The **Configured Libraries** window contains a bar. On color monitors, this bar is green. It shows the position where the next library will be inserted. To move this bar, point the cursor where you want it to appear and click the left mouse button.

Click the **Insert** button. The selected library is added to the **Configured Libraries** list, above the green line.

For information about the order of libraries, see *Library Options*. For information about changing the order of libraries, see *Changing the library order*.

Removing a library

To remove a library from the **Configured Libraries** list box, select the **Remove a Library** option. The **Available Libraries** window becomes dimmed. In addition, the **Remove** button becomes active and available for use.

Select the library that you would like to remove from the **Configured Libraries** list by clicking on its name. If you need to see other libraries that aren't displayed in the window, use the scroll buttons at the right of the window to scroll the list of libraries up and down.

Once you select a library, click the **Remove** button. The selected library is removed from the **Configured Libraries** list.

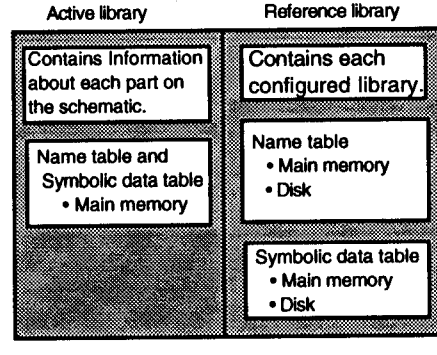
Changing the library order

Draft loads and maintains libraries in the order in which they are listed in the **Configured Libraries** list box. This is important when retrieving parts while creating schematics. When you tell **Draft** to get a certain part name, it searches the libraries in the order they are listed in the **Configured Libraries** window and gets the first part it finds with a matching name. If you want to change the order in which your libraries are listed, follow these steps:

- Libraries must be reordered one at a time. Determine which library you want to move and remove it from the **Configured Libraries** list box.
- Select the **Insert a Library** option. Move the green bar in the **Configured Libraries** list until it is positioned where you want to insert the library.
- Insert the library that you removed earlier. It appears in the **Configured Libraries** window just above the green line.

Name Table Location and Symbolic Data Location

Schematic Design Tools 386+ uses two types of libraries: the active library and the reference library. Both of these libraries contain a *name table* and a *symbolic data table.*



The active library contains information about each part *on the schematic.*

The reference library contains information about *each configured library.* You can configure **Schematic Design Tools 386+** to store it in main memory (which includes extended memory) or on disk using the options listed under **Name Table Location** and **Symbolic Data Location.** For information about the reference library, see *Reference library* in the next section.

Reference library

The reference library contains information about *each configured library*. These are the libraries listed in the **Configured Libraries** list box. The reference library contains a name table and a symbolic data table.

- The name table contains a list of all the parts in *each configured library*. It can be stored in main memory or on disk.
- The symbolic data table contains all of the symbol information for each part in *each configured library*. It can be stored in main memory or on disk. If you place the symbolic data table in main memory, **Draft's GET** and **LIBRARY Browse** commands run more quickly.

If you have limited main memory, you can configure the software to keep the symbolic data table on disk. Depending on the speed of your disk, **Draft's GET** and **LIBRARY Browse** commands will slow down a little or a lot, but **Draft** will redraw the screen as fast as always, because the information it uses for redraws is in the active library.

Depending on the performance of your disk drive and the demands of software that runs in the background, you can expect the performance impacts shown in table 1-1.

	Main memory	*Disk*	*Comments*
Name Table Location *Symbol Table Location*	✔ ✔		This is usually the most efficient configuration. **Draft's GET** and **LIBRARY Browse** commands work fastest under this configuration.
Name Table Location *Symbol Table Location*	✔	 ✔	This is slower yet, but is still tolerable. This is the best option for PCs with limited main memory.
Name Table Location *Symbol Table Location*		✔ ✔	This is the slowest configuration. It should only be used with computers that have minimal memory (4M or less). It is tolerable for long use only if your hard disk is fast.

Table 1-1. Performance impacts based on the location of the reference library.

Active library

The active library contains information about each part *on the schematic*. It always resides in main memory. Like the reference library, it has a name table and a symbolic data table.

- The name table contains a list of the parts found *on the schematic*.
- The symbolic data table contains all of the symbol information for each part *on the schematic*.

Draft builds this library by copying information from the other libraries as it loads a schematic or when you get a new part using **Draft's GET** command, and discards it when you exit **Draft**. Because all of the needed information is in one library, redraws and panning are very fast.

Initial memory allocated

This is the space allocated for libraries and design files. The range is 2000K to 31,000K. If your design grows larger than this initial allocation, additional memory, if available, is allocated.

Worksheet Options

The **Worksheet Options** area (figure 1-7) defines the worksheet prefix, the default worksheet file extension, and default title block information.

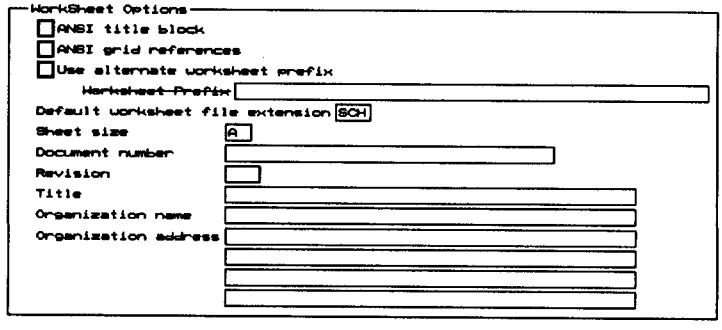


*Figure 1-7. **Worksheet Options** area of the **Configure Schematic Design Tools** screen.*

Select any combination of the following options:

- ANSI title block

  Causes **Schematic Design Tools 386+** to use the ANSI Standard Y14.1-1980 title block on worksheets, instead of the default.

  The default title block is shown below.

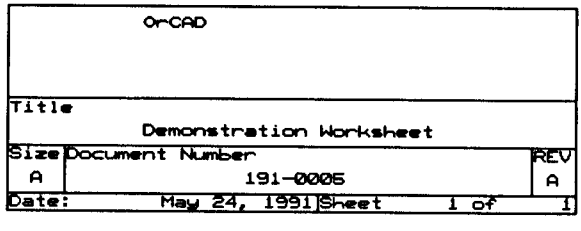


Figure 1-8. Sample OrCAD title block.

The alternative, an ANSI Standard title block, is shown in figure 1-9. ANSI title blocks are larger than the default OrCAD title blocks. On an A-size drawing, they take up a large amount of the drawing area.

△ *NOTE: You may use extended ASCII characters in any of the **Worksheet Options** entry boxes except **Sheet Size**.*

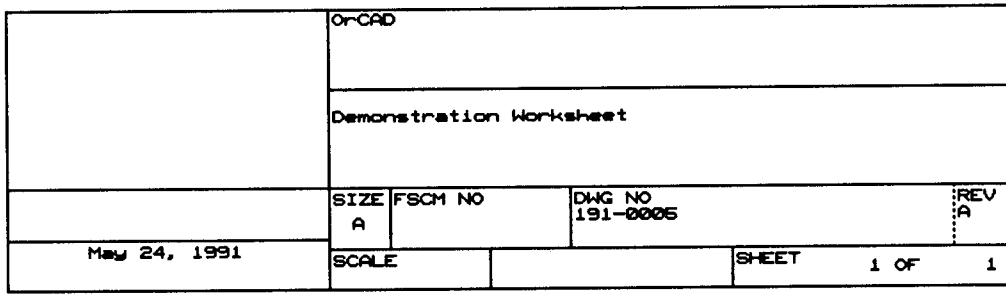


Figure 1-9. ANSI title block for sheet sizes A through C.

The ANSI title blocks for sheet sizes A, B, and C are different from ANSI title blocks for sheet sizes D and E. See the ANSI Y14.1-1980 specification for more information.

❑ `ANSI grid references`

Causes **Schematic Design Tools 386+** to use ANSI Standard Y14.1-1980 grid references on worksheets. Table 1-2 shows both ANSI and common grid references.

Sheet Size	*ANSI References*		*Common References*	
	X Grid Range	*Y Grid Range*	*X Grid Range*	*Y Grid Range*
A	N/A	N/A	1..8	A..D
B	N/A	N/A	1..8	A..D
C	1..4	A..D	1..8	A..D
D	1..8	A..D	1..8	A..D
E	1..8	A..H	1..8	A..D

Table 1-2. X and Y grid references. N/A indicates that the value is not applicable because the sheet size does not have grid references per ANSI Y14.1-1980.

❑ Use alternate worksheet prefix

Do not select this option if you are using the ESP design environment. The ESP design environment manages all paths and prefixes for you. This option is provided for compatibility with older versions of OrCAD software. When you select this option, the **Worksheet Prefix** entry box becomes available:



The **Worksheet Prefix** is the disk drive and directory path where **Draft** finds worksheet files containing your schematic designs.

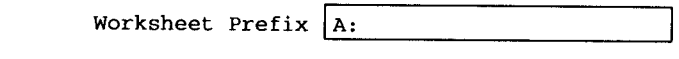
If you make an entry that falls into any of the categories listed below, **Draft** looks for the worksheet in the location specified in the **Worksheet** entry box:

- Drive name. For example, A:, B:, C:, or D:.
- Drive name followed by a backslash (\). For example, A:\, B:\, C:\, or D:\.
- A backslash (\).
- A pathname that begins with any of the categories listed above.

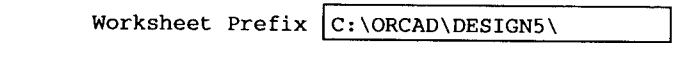
If you add a directory path that doesn't begin like one of the examples outlined above, **Draft** treats the directory path as a subdirectory in the current design.

Examples

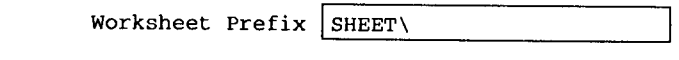
Use this prefix to find files on the A: floppy disk:



Use this prefix to find files in the \ORCAD\DESIGN5 subdirectory on the C: hard disk:



Use this prefix to find files in the subdirectory called SHEET in the current design directory:



Default worksheet file extension

If you enter a filename that doesn't end with a filename extension or a period in response to one of Draft's prompts, **Draft** appends the extension specified in **Default worksheet file extension** to the name. For example, if you enter "SHEET" while in **Draft**, **Draft** appends the extension specified in **Default worksheet file extension**.

If you enter a filename that ends with an extension or ends with a period (.), this field is ignored. For example, if you enter "SHEET." or "SHEET.EXT" while in **Draft**, **Draft** does not append an extension.

△ *NOTE: The information entered in the remaining fields is used as the default information on a schematic worksheet. For this information to appear on a worksheet, it must be entered* before the worksheet is created.

*Once a worksheet is created, any changes made here are not reflected on an existing worksheet. To change sheet size, use **Draft's SET** command. To change the worksheet's title block information, use **Draft's EDIT** command.*

Sheet size

This field must be set to A, B, C, D, or E (American), or to A4, A3, A2, A1, or A0 (International Standards Organization). You specify sheet dimensions on the **Template Table**.

Document number

The document number may contain up to 36 characters.

Revision

The revision code may contain up to three characters.

Title

The title may contain up to 44 characters.

Organization name

The organization name may contain up to 44 characters.

Organization address

Each organization address line may contain up to 44 characters.

Macro Options

A macro is a series of commands that runs automatically at the touch of a single key or key combination. You can construct macros and store them in a file for later use with **Draft** or **Edit Library.**

Macro Options (figure 1-10) defines the macro buffer size, the macro file (a file that can contain many macros), and the name of an initial macro (a macro within the macro file) that runs each time you run **Draft** or **Edit Library.**

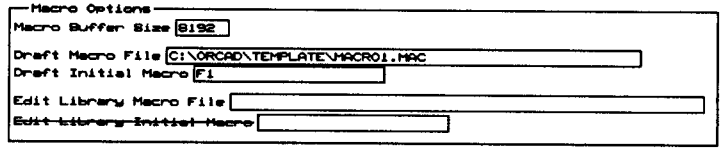


*Figure 1-10. **Macro Options** area of the **Configure Schematic Design Tools** screen.*

With macros you can dramatically reduce the number of keystrokes required to perform complex or repetitive actions. Macro files can contain many macros. Macro file size is limited by the size of the macro buffer (defined in the **Macro Buffer Size** entry field, described below).

To help you use macros right away, the **Schematic Design Tools 386+** master disks contain two OrCAD-supplied macro files, MACRO1.MAC and MACRO2.MAC. These macro files are a useful starter set for **Draft**. Note that they are not designed for **Edit Library**. MACRO1.MAC and MACRO2.MAC reside in the TEMPLATE directory. They are copied to each new design you create.

Macro Buffer Size

The macro buffer is the storage location for macros read in from a file or created but not written to a file. The amount of memory allocated to the macro buffer is entered in the **Macro Buffer Size** entry box. The minimum memory size for the macro buffer is 8192 bytes; the maximum is 65,535 bytes.

Example

In the example below, the macro buffer is set at its minimum size.

```
Macro Buffer Size [8192        ]
```

Draft Macro File and Edit Library Macro File

The macro file is the path and filename of the macro file to load automatically whenever you run **Draft** or **Edit Library**.

To define a macro file, enter the path and filename of the macro file in the **Draft Macro File** or **Edit Library Macro File** entry box. If the macro file is not in the design directory, you must specify a full pathname to the macro file. After you create and save your own macro files, its name can be entered here for automatic loading.

Example

The example below tells **Draft** to look for the macro file named MACRO1.MAC on the C: hard disk in the \ORCAD\TEMPLATE directory.

```
Draft Macro File [C:\ORCAD\TEMPLATE\MACRO1.MAC]
```

Draft Initial Macro and Edit Library Initial Macro

The initial macro is a keystroke that **Draft** or **Edit Library** issues automatically whenever you run the respective program. This keystroke should be one defined to run a particular macro. For the initial macro to work, a filename (containing the desired macro definition) must be defined in the **Macro File** entry box.

If a filename is not entered in the **Macro File** entry box, the **Initial Macro** entry box is dim.

Examples

The example below tells **Draft** to execute the macro assigned to function key <F1> automatically whenever you run **Draft**. To enter <F1>, type <F> and then <1>.

```
Draft Initial Macro [F1                    ]
```

If the macro you want for the initial macro runs when you press the keys <Ctrl> and <A> simultaneously, represent the <Ctrl> key with the *caret* character (^), which you type by pressing <Shift> <6>. Enter the characters <^> <A> in the **Draft Initial Macro** entry box:

```
Draft Initial Macro [^A                    ]
```

For more information, see the **MACRO** command description in *Chapter 2: Draft.*

Hierarchy Options

Hierarchy Options (figure 1-11) defines the size of the hierarchy buffer.

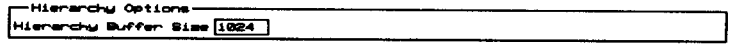


*Figure 1-11. **Hierarchy Options** area of the **Configure Schematic Design Tools** screen.*

Hierarchy Buffer Size

All hierarchical sheet and pathnames are stored in the hierarchy buffer. The amount of memory allocated to the hierarchy buffer is entered in the **Hierarchy Buffer Size** entry box.

The minimum memory size is 1024 bytes, which allows you to create a hierarchical depth of about 75 to 100 worksheets (depending on sheet and pathname character lengths).

You can increase the size of the hierarchy buffer to 65,535 bytes, large enough for a hierarchical depth of over 200 worksheets.

Example The hierarchy buffer is set at its minimum size.

Hierarchy Buffer Size [1024]

Color and Pen Plotter Table

The **Color and Pen Plotter Table** (figure 1-12) selects the screen display and plotter pen colors for library parts, pin numbers and names, wires, buses, junctions, connectors, and other objects in the worksheet. The objects are listed in the left column of the table.

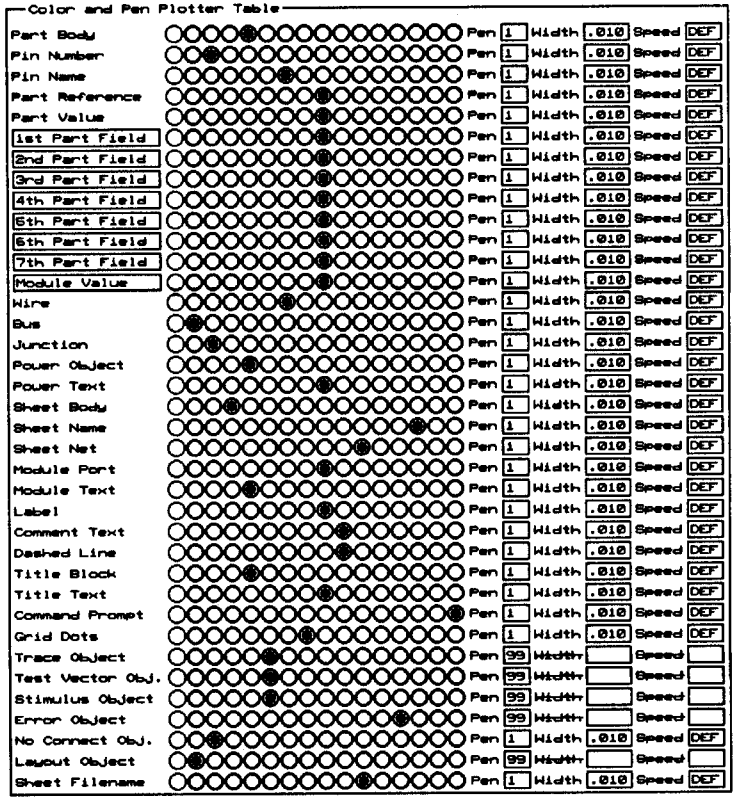


Figure 1-12. Color and Pen Plotter Table area of the Configure Schematic Design Tools screen.

Color To change an item's color, simply select the desired color.

Pen To select a pen, enter its number in the **Pen** entry box for the desired object. Valid entries are:

1-16 Valid pen numbers.

0 Causes the plotter to pause so you can change pens.

99 Causes the plotter to not plot the object and the **Width** and **Speed** entry boxes to become dimmed.

Examples Here are some examples showing how to use the **Pen** entry box to suppress plotting the title block's lines, text, or both.

Suppressing title block lines

To suppress title block lines and leave title block text on the worksheet, enter **99** in the **Pen** entry box to the right of **Title Block**.

When you open a worksheet in **Draft**, the title block lines still display. However, they do not appear on the plot, or on **Plot Schematic** output sent to a printer, as shown in the figure below.

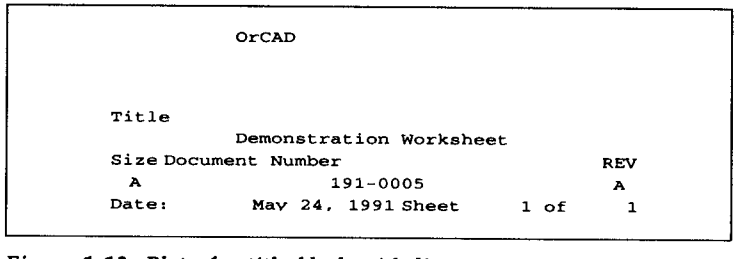


Figure 1-13. Plot of a title block with lines suppressed.

△ *NOTE: This also turns off the border around the drawing area.*

Suppressing title block text

To suppress title block text and leave title block lines on the worksheet, enter **99** in the **Pen** entry box to the right of **Title Text**.

When you open a worksheet in **Draft**, the title block text still displays. However, it does not appear on the plot, as shown in the figure below.

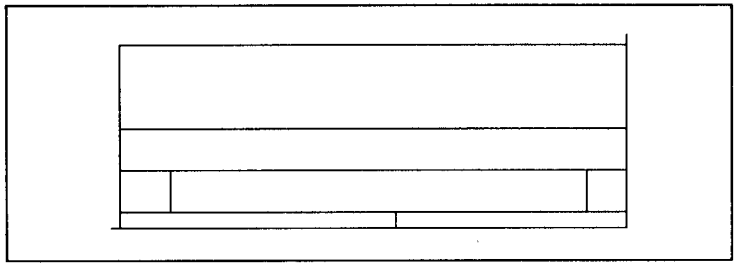


Figure 1-14. Plot of a title block with text suppressed.

Suppressing title block lines and *text*

There are two ways to suppress title block lines *and* text:

- Set both the title block and title text to a pen of **99**. Using this method, the title block and its text display on the screen, but do not appear on a plot.
- Use **Draft's SET Title Block No** command. If you use this method, the title block and its text do not display on the screen or appear on a print or a plot.

➤ *Helpful hint . . .*

If you are plotting on paper pre-printed with your title block, suppress the title block and its text as described above. Use **Draft's PLACE** command to place text in the correct position so that when you plot your schematic, the text prints in the correct place in the pre-printed title block.

Width Pen width is the actual width of the pen the plotter uses to draw an object. **Plot Schematic** uses this value to calculate the number of pen strokes needed to create an object. The value is expressed in inches (0.010 is 1/100 of an inch).

Speed

Enter the pen's velocity in the **Speed** entry box for the desired object. See your plotter manual to correctly set the speed. The value is expressed in units, a measurement determined arbitrarily by the plotter manufacturer.

Entering **DEF** in the **Speed** entry box tells **Plot Schematic** to use to your plotter's default speed. **Plot Schematic** makes no change to the pen speed.

1st through 7th Part Field and Module Value

The names used to label the **1st Part Field** through the **7th Part Field** and **Module Value** can be changed. Look at the **Color and Pen Plotter Table**, and notice that the names **1st Part Field** through **7th Part Field** and **Module Value** are contained in entry boxes. To change any of these names, simply click inside the entry box and edit the name. The new name appears in **Draft** when you use the **EDIT Part Name** command.

Look at the **Template Table** on the **Configure Schematic Design Tools** screen and notice that it also contains the **1st Part Field** through **7th Part Field** and **Module Value** entry fields. If you change these headings in the **Color and Pen Plotter Table**, they are also changed in the **Template Table**.

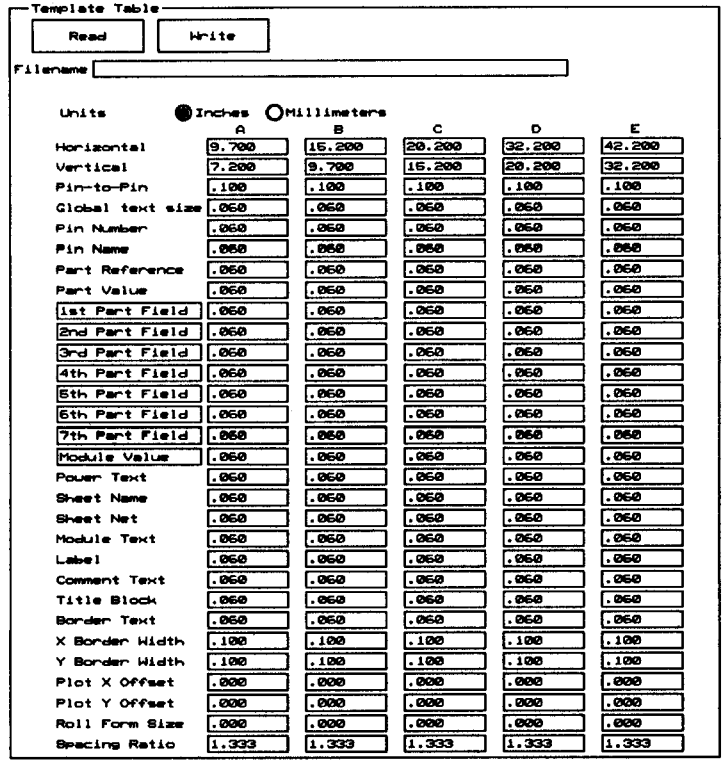
Template Table

The **Template Table** (figure 1-15) contains the global settings for the size (in inches or millimeters) of various parameters in schematics. This includes worksheet dimensions, text size, border size, pin-to-pin spacing, and so on.

Draft has five sheet sizes built in: A through E (US) or A4 through A0 (International Standards Organization). Using the **Template Table**, you can tailor the dimensions and many characteristics of each to match your requirements.

The maximum worksheet dimensions are 65 inches by 65 inches, including the border. Object scaling is controlled by the **Pin to Pin** values.

To change one of the values, simply click inside the appropriate entry box and edit the value.



	A	B	C	D	E
Horizontal	9.700	15.200	20.200	32.200	42.200
Vertical	7.200	9.700	15.200	20.200	32.200
Pin-to-Pin	.100	.100	.100	.100	.100
Global text size	.060	.060	.060	.060	.060
Pin Number	.060	.060	.060	.060	.060
Pin Name	.060	.060	.060	.060	.060
Part Reference	.060	.060	.060	.060	.060
Part Value	.060	.060	.060	.060	.060
1st Part Field	.060	.060	.060	.060	.060
2nd Part Field	.060	.060	.060	.060	.060
3rd Part Field	.060	.060	.060	.060	.060
4th Part Field	.060	.060	.060	.060	.060
5th Part Field	.060	.060	.060	.060	.060
6th Part Field	.060	.060	.060	.060	.060
7th Part Field	.060	.060	.060	.060	.060
Module Value	.060	.060	.060	.060	.060
Power Text	.060	.060	.060	.060	.060
Sheet Name	.060	.060	.060	.060	.060
Sheet Net	.060	.060	.060	.060	.060
Module Text	.060	.060	.060	.060	.060
Label	.060	.060	.060	.060	.060
Comment Text	.060	.060	.060	.060	.060
Title Block	.060	.060	.060	.060	.060
Border Text	.060	.060	.060	.060	.060
X Border Width	.100	.100	.100	.100	.100
Y Border Width	.100	.100	.100	.100	.100
Plot X Offset	.000	.000	.000	.000	.000
Plot Y Offset	.000	.000	.000	.000	.000
Roll Form Size	.000	.000	.000	.000	.000
Spacing Ratio	1.333	1.333	1.333	1.333	1.333

Figure 1-15. Template Table area of the Configure Schematic Design Tools screen.

Filename You can create your own template table files using the **Read** and **Write** buttons in the **Template Table** area of the **Configure Schematic Tools** screen. To save a template table, enter a path and filename in the **Filename** entry box and then select **Write**. To load a template table, enter a path and filename and select **Read**. The template table reads in the new values and over-writes the present values.

Units **Draft** has five worksheet sizes built in: A through E (US) or A4 through A0 (International Standards Organization). Using the **Template Table**, you can tailor the dimensions and many characteristics of each to match your requirements.

Select either the **Inches** or **Millimeters** option to choose the unit of measure. If you select **Inches**, all measurements are shown in inches. Note also that the headings above each column are A, B, C, D, and E.

If you select **Millimeters**, all measurements are shown in millimeters. Note also that the headings above each column are A4, A3, A2, A1, and A0.

▲ *CAUTION: If you change from inches to millimeters or vice versa, all custom settings you may have made are lost, and the default settings for each item are restored.*

Horizontal Width (horizontal dimension) of the schematic work-sheet working area. The maximum width allowed for a schematic worksheet is 65 inches.

Vertical Height (vertical dimension) of the schematic worksheet working area. The maximum height allowed for a schematic worksheet is 65 inches.

About worksheet size The border, if there is to be one, is drawn inside the rectangle defined by the **Horizontal** and **Vertical** dimensions.

Tables 1-3 and 1-4 list the sizes of ANSI (A–E) and ISO (A4–A0) sheet sizes and drawing areas within the specified borders. Unfortunately, most, if not all, PC-compatible printers and plotters are unable to print as close to the edge of the page as specified in the ANSI and ISO standards.

Tables 1-5 and 1-6 list the reduced dimensions that will work with most printers and plotters. It is possible that your printer or plotter can print closer to the edge of the paper than allowed by these values. Hence, you may wish to adjust the sizes in the **Horizontal** and **Vertical** entry boxes.

△ *NOTE: The maximum worksheet dimensions are 65 inches by 65 inches, including the border.* ***Plot Schematic*** *will only print schematics up to 32 inches by 32 inches in size.* ***Plot Schematic*** *scales anything larger to fit in these dimensions.*

Sheet Size	*Outside Border*		*Inside Border*	
	Horizontal	*Vertical*	*Horizontal*	*Vertical*
A	11.0	8.5	10.5	7.7
B	17.0	11.0	15.7	10.2
C	22.0	17.0	21.0	15.5
D	34.0	22.0	32.0	21.0
E	44.0	34.0	43.0	32.0

Table 1-3. ANSI horizontal and vertical dimensions of worksheets in inches.

Sheet Size	Outside Border		Inside Border	
	Horizontal	Vertical	Horizontal	Vertical
A4	297	210	276.8	187.9
A3	420	297	388.6	266.7
A2	594	420	563.8	388.6
A1	841	594	800.1	553.7
A0	1189	841	1137.9	789.9

Table 1-4. ISO horizontal and vertical dimensions of worksheets in millimeters.

Sheet Size	Outside Border		Inside Border	
	Horizontal	Vertical	Horizontal	Vertical
A	9.7	7.2	9.5	7.0
B	15.2	9.7	15.0	9.5
C	20.2	15.2	20.0	15.0
D	32.2	20.2	32.0	20.0
E	42.2	32.2	42.0	32.0

Table 1-5. OrCAD default horizontal and vertical dimensions of worksheets in inches.

Sheet Size	Outside Border		Inside Border	
	Horizontal	Vertical	Horizontal	Vertical
A	246.4	182.9	241.3	177.8
B	386.1	246.4	381	241.3
C	513.1	386.1	508	381
D	817.9	513.1	812.8	508
E	1071.9	817.9	1066.8	812.8

Table 1-6. OrCAD horizontal and vertical dimensions of worksheets in millimeters.

Pin-to-Pin Determines the minimum distance between two pins on a part and also controls template size (as defined in ANSI and object scaling. The default value (0.1 inch) is a half-size template; a full-size template (based on the ANSI standard) results when the pin-to-pin spacing is set to 0.2 inch. It also changes the distance between grid dots.

Global text size Height of all text in a schematic. You can use the **Global text size** field to globally change the text height for all text fields and then change any single field if you wish.

This value is not saved when you select **OK** or when you write the template table to a file. When you read in a new template table, the **Global text size** is set to the value entered in the **Pin number** field.

Pin Number Height of a part's pin numbers. Because pin numbers appear between pins, do not make the pin number size greater than the pin-to-pin spacing.

Pin Name Height of a part's **Pin Name** characters.

Part Reference Height of a part's **Part Reference** characters.

Part Value Height of a part's **Part Value** characters.

1st through 7th Part Field and Module Value Height of the characters used to show the information contained in the part fields.

Notice that the names **1st Part Field** through **7th Part Field** and **Module Value** are contained in entry boxes, indicating that you can change these names—just like in the **Color and Pen Table Plotter Table.**

To change a name, simply click inside the box and edit the name. The new name appears in **Draft** when using the **EDIT Part Name** command.

The **Color and Pen Plotter Table** on the **Configure Schematic Design Tools** screen also lists the **1st Part Field** through **7th Part Field** and **Module Value** entry fields. Changing these headings in the **Template Table** also changes them in the **Color and Pen Plotter Table.**

Power Text Height of power object characters.

Sheet Name Height of sheet name characters.

Sheet Net Height of sheet net name characters.

Module Text Height of module port name characters.

Label Height of label characters.

Comment Text Height of comment text characters.

Title Block Height of title block characters.

Border Text Height of the characters used to show text in the border. For readable results, do not make border text larger than the border width.

X Border Width and Y Border Width Width of the border around the worksheet. **Draft** draws the border inside the rectangle defined by the horizontal (**X Border Width**) and vertical (**Y Border Width**) dimensions. Grid references are shown inside the border.

Plot X Offset Moves the origin horizontally. **Plot Schematic** uses the lower left corner of a sheet as its origin point when plotting. Some plotters (such as Hewlett-Packard C, D, and E size plotters) place the origin in the page's center, which results in the worksheet not plotting properly: the lower left corner of the worksheet is plotted in the center of the page. Use this entry box to compensate for this. The correct X offset can be calculated as shown below:

$$Plot\ X\ Offset = -\frac{1}{2}\ page\ width$$

△ *NOTE: This is a negative number.*

Plot Y Offset Moves the origin vertically. Use it to compensate for plotters that do not use the lower left corner as the origin point. The correct Y offset can be calculated as shown below:

$$Plot\ Y\ Offset = -\frac{1}{2}\ page\ height$$

△ *NOTE: This is a negative number.*

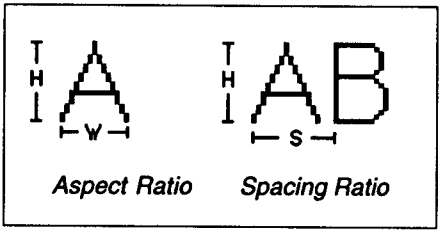
Roll Form Size Amount of paper to unroll from the spool when done plotting.

Spacing Ratio Determines the relationship between a character's vertical height and horizontal distance between the start of one character and the start of the next character:

$$Spacing\ Ratio = \frac{Pin\text{-}to\text{-}pin\ spacing * 0.08}{Height\ (H) * 0.10}$$

This is not the same as the character's aspect ratio, which is the relationship between a character's vertical height and its horizontal width.

$$Aspect\ Ratio = \frac{Height\ (H)}{Width\ (W)}$$



You can change the spacing ratio to fine-tune the appearance of plots when **Pin-to-Pin** spacing changes. For example, if **Pin-to-Pin** spacing is set to 0.2 inch (a full-size template), and text height set to 0.156 inch, the spacing ratio is 1.026. This gives a spacing ratio of 1.026 inch.

$$Spacing\ Ratio = \frac{0.2*0.08}{0.156*0.10} = 1.026$$

Default settings

Draft's default is a half-size template (with pin-to-pin spacing at 0.1 inch), all text sizes of 0.060 inch, and a spacing ratio of 1.333.

$$Spacing\ Ratio = \frac{0.1*0.08}{0.06*0.10} = 1.333$$

Key Fields

To understand *key fields* and how to use them, you need to understand *part fields.*

Associated with every library part are ten *part fields.* A part field is a slot for holding text or data associated with that part. These fields can be accessed and edited after a part is placed in a design.

```
Reference
Part Value
1st Part Field
2nd Part Field
3rd Part Field
4th Part Field
5th Part Field
6th Part Field
7th Part Field
Module Value
```

The ten part fields.

Two part fields are reserved for particular types of data:

- The **Reference** field is reserved for holding reference designator values, such as "U1A" or "Q1."
- The **Part Value** field is reserved for holding part names, such as "74LS04" or values relevant for the part, such as Ohm (Ω) values for resistors.

To be processed correctly by **Schematic Design Tools 386+**, every part (including parts not supplied by OrCAD) *must* have data in the **Reference** field and in the **Part Value** field. The only exceptions to this rule are objects whose only pin is of type POWER (ground symbols, for example).

The other eight fields are named **1st Part Field** through **7th Part Field** and **Module Value.**

△ *NOTE: You can change the names used to identify* ***1st Part Field*** *through* ***7th Part Field*** *and* ***Module Value.*** *See the sections in this chapter about the* ***Color and Pen Plotter Table*** *and the* ***Template Table*** *for instructions on how to do this.*

You can store any useful information in the eight part fields: tolerance, vendor name, part number, and so on. The last part field is configured for the Module Value. Each field can be up to 127 characters long. You can edit the contents of these fields and make them visible or invisible on the schematic using **Draft's EDIT** command. You can also use the **Select Field View** processor to make a part field visible or invisible for all parts in a design.

To perform special processing, the tools that affect or use part field values can combine and compare the information in these part fields. For example, the way **Annotate Schematic** designates parts for grouping in component packages can be changed, or **Create Bill of Materials** can list parts grouped by tolerance *and* value, rather than by value alone.

To tell the tools which fields you want to combine and compare, *key fields* are used (figure 1-16). A key field lists the part fields to combine and compare. Key fields are specific to a particular tool. **Schematic Design Tools 386+** has sixteen key fields: **Annotate Schematic** uses one key field, **Create Netlist** uses two, **Create Bill of Materials** uses two, **Update Field Contents** uses nine, and **Extract Source from Schematic** uses two.

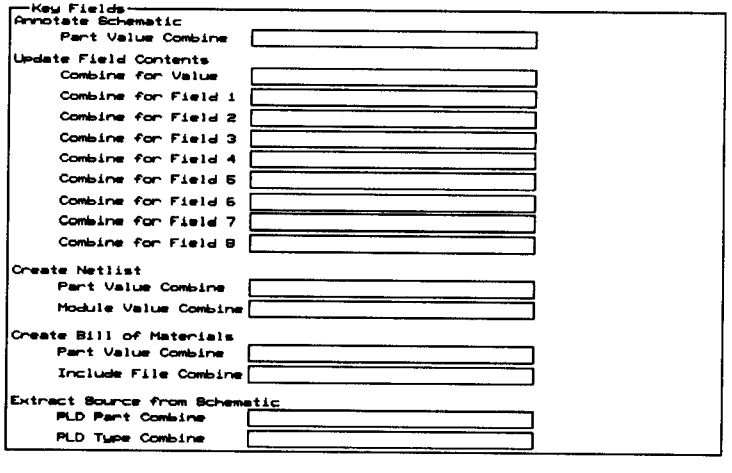


Figure 1-16. Key Fields area of the Configure Schematic Design Tools screen.

Key fields can contain:

- The character **R** to represent the value found in the **Reference** field.
- The character **V** to represent the value found in the **Part Value** field.
- The numerals **1** through **8** for the values found in **Part Fields 1** through **7** and **Module Value**. Note that the **Module Value** field is also known as **Part Field 8**.
- Any combination of text (including blank spaces, commas, and other punctuation). Note that you cannot include **V** and **R** in a text string, as those are the characters used to represent the values found in the **Reference** and **Part Value** fields. You can, however, use **v** and **r** in a text string.

The total length of a key field combination should not exceed 127 characters. For example, if **V 1** is in your key field, the total length of the data stored in the **Part Value** field, plus the data in the **1st Part Field**, plus the space character between them cannot be longer than 127 characters.

Now you know how to specify values in the key fields. The next sections explain the different types of key fields, what they do, and how to use them.

Annotate Schematic Part Value Combine

When you run **Annotate Schematic** on a schematic design, it puts a reference designator value in the **Reference** field of each part.

Some parts come in packages containing more than one part. For example, the 7439 package contains four two-input NAND buffers. When annotating schematics with parts that come in packages containing multiple parts, **Annotate Schematic** groups parts with the same value in their **Part Value** fields in the same package (until the package is full). **Annotate Schematic** creates reference designator values naming first the packages the parts come in and then the occurrence of the part within the package.

Example

Suppose a design contains six parts with values of "74LS08" in their **Part Value** fields, and the part library specifies these parts are bundled four per package.

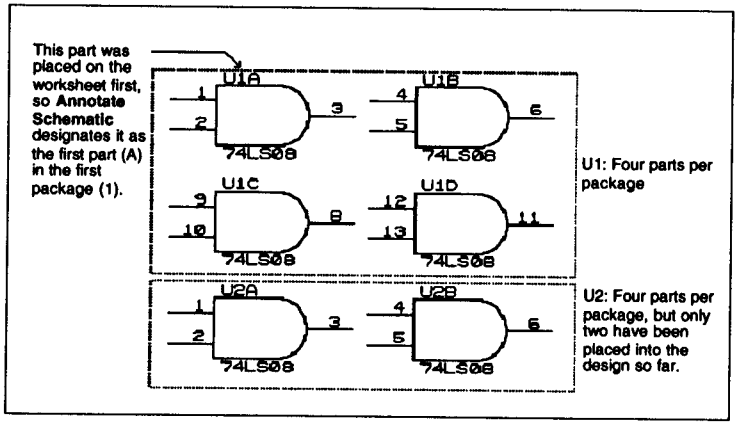


*Figure 1-17. How **Annotate Schematic** normally assigns reference designators.*

When you run **Annotate Schematic** on this design, it assigns reference designators (such as U1A, U1B, U1C, U1D, U2A and U2B) to these parts (figure 1-17). U1 names one package and U2 names a second; the letters A–D name the particular parts within each package.

Reference designators are assigned to parts in the order in which you placed them on the worksheet. In this example, the first 74LS08 placed in the worksheet is numbered U1A and the last one placed U2B.

Suppose for physical layout reasons you want to partition the circuit to group these six parts differently. You want three of them in one package, and three in another. This is where **Annotate Schematic**'s key field comes in handy.

First designate which portion of the circuit each part is to be grouped in. Assign the value "MOD_1" or "MOD_2" to the **1st Part Field** of every part in the design using **Draft's EDIT Part** command (figure 1-18). If your worksheet is crowded, hide the information from view by making the field invisible. For more information, see *Chapter 13: Select Field View*.

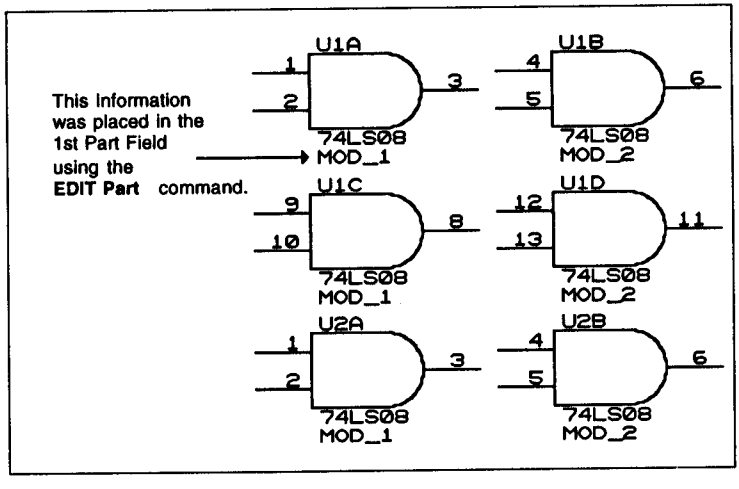


*Figure 1-18. Parts with "MOD_1" and "MOD_2" values assigned to **1st Part Field**.*

Next, configure the key field for **Annotate Schematic** so that it carries out this scheme in the way it assigns reference designators. In the **Key Field** area of the **Configure Schematic Design Tools** screen, enter the characters **V 1** into the **Annotate Part Value Combine** edit field. For example:

```
Annotate Part Value Combine [V 1                    ]
```

From now on, when **Annotate Schematic** assigns reference designators to parts in a design, it will not group parts (that come several per package) in the same package unless they have the same **Part Value** *and* the same value in **Part Field 1.**

So, if on three of the 74LS08s you place the value "MOD_1" in **Part Field 1,** and on the other three 74LS08s you placed the value "MOD_2" in **Part Field 1, Annotate Schematic** assigns each set of three its own package. The reference designators for one set would be something like U1A, U1B and U1C, and the reference designators for the other set would be something like U2A, U2B, and U2C.

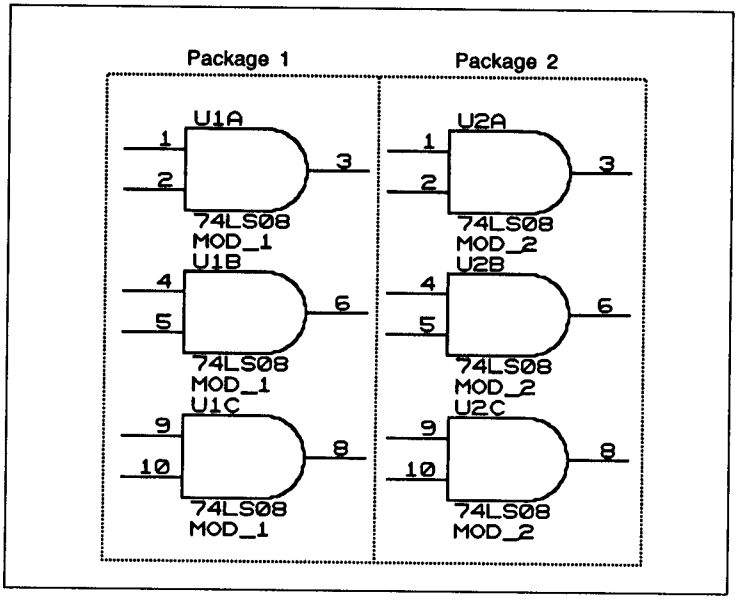


*Figure 1-19. **Annotate Schematic** assigned reference designators based on **Part Value** (74LS08) and **1st Part Field** data.*

Update Field Contents

To understand how these key fields work, you must understand how **Update Field Contents** works.

Update Field Contents is a search and replace tool. It looks for certain text in a selected part field. When it finds a match, it places new data in a selected part field.

- You define the part fields to match in the **Key Fields** area on the **Configure Schematic Design Tools** screen.
- You select the part field to update on the configuration screen for **Update Field Contents.**
- You create an update file to specify the text to match the part fields and the text to replace. The update file consists of pairs of strings. Each string in the update file is enclosed by single quotation marks, like this:

```
'74LS04'        '14 pin Inverter Package'
'74LS138'       '16 pin Inverter Package'
```

The first part of each pair is the text to match with the part field specified in the **Key Fields** area of the **Configure Schematic Design Tools** screen. The second part of each pair is the data to be placed in the part field specified on the configuration screen for **Update Field Contents.**

For details about the format of the update file, see *Chapter 14: Update Field Contents.*

Combine for Value and Combine for Fields 1 through 7 and Module Value

To specify the part field to match, you make an entry in the **Combine for Value** entry box (to update the **Part Value** field) or one of the **Combine for Field** entry boxes (to update a **Part Value** field). This entry can be an "R" (for reference), a "V" (for part value), a digit 1–8 (for 1st through 7th part field and module value, respectively), or a combination of these characters.

Update Field Contents determines which part field is selected on its configuration screen and looks at the entry for that part field here.

For example, if the field to be updated—as selected on the configuration screen for **Update Field Contents**—is **Part Field 2**, **Update Field Contents** looks in the **Combine for Field 2** entry box to determine which part fields should match with the text in the update file. Remember, an update file consists of pairs of text string. The first item in a pair is the text to match.

You can match on more than one field by making a combination entry. For example, enter **V 1** to match the **Part Value** field and **Part Field 1** with the text in the update file.

For each part in the design, **Update Field Contents** builds a text string from the value found in the part's **Part Value** field, a blank space, and the value found in the part's **1st Part Field**.

Example

Assume that the **Combine for Field 2** entry box contains `V 1`, as described above.

If a part has a value of "74LS04" in its **Part Value** field, and a value of "14DIP300" in its **1st Part Field**, **Update Field Contents** builds the string:

```
74LS04 14DIP300
```

Then, **Update Field Contents** compares this match string to the first text string of each pair of text strings it finds in the update file.

For this example, the first item in each pair of text strings should contain *two* items to match, as shown here:

```
'74LS04 14DIP300' '14 pin Inverter Package'
```

Update Field Contents puts the text "14 pin Inverter Package" into the **2nd Part Field** of a part whose **Part Value** field reads "74LS04" and whose **1st Part Field** reads "14DIP300."

Points to remember

When defining part fields to match, keep these points in mind:

- Only one field can be updated at a time—the one selected on the local configuration screen for **Update Field Contents**. Each field can have its own separate search criteria, so there are nine key fields for **Update Field Contents** on the **Configure Schematic Design Tools** screen.
- The **Reference** field is protected and cannot be updated with **Update Field Contents**. If you wish to change the **Reference** field, use the **EDIT Reference** command in **Draft**.
- For more information on **Update Field Contents**, see *Chapter 14: Update Field Contents*.

Create Netlist Part Value Combine

Configuring this key field tells **Create Netlist** and **Create Hierarchical Netlist** to read the **Part Value** (or combination of values) in the **Netlist Part Value** field for each part in the design and use this as the **Part Value** for each part in the netlist. Generally, you should set this field to "V" or empty.

What value, if any, you should substitute for the **Part Value** depends on the netlist format you want to produce. Different applications require netlists with different types of values in the netlist's **Part Value** position.

Create Netlist Module Value Combine

Enter the number 8 in this entry box to indicate that the last part field (**Module Value**) holds the module value.

This key field tells **Create Netlist** and **Create Hierarchical Netlist** to create a *module value* to show the name of the module to be used for layout. Most layout products refer to the pad shape separately from the part value.

For example, a 74LS00 comes in a 14-pin 0.300-inch center DIP package or in a surface-mount package. Accordingly, the *module value*, also known as a footprint, is used to indicate the package type so the layout can have the correct pad shape.

What value, if any, you create for the *module value* depends on the particular netlist format you want to produce. Different applications require netlists with different types of module values.

If you do not specify this key field, the *module value* will be the **Part Value** field.

Create Bill of Materials Part Value Combine

To understand how this key field works, you must understand how **Create Bill of Materials** works.

Create Bill of Materials creates a list of the parts placed in a specified design. It uses the values found in the **Part Value** field of each part to determine whether parts are the same or different. **Create Bill of Materials** lists the parts it identifies as the same on the same line.

Configuring this key field instructs **Create Bill of Materials** to read the **Part Value** (or combination of values) in the **Combine Field** for each part in the design and use this as the **Part Value** for each part in the Bill of Materials. Generally, you should leave this field set to "V" (for Value) for most Bill of Materials formats.

Create Bill of Materials Include File Combine

The **Bill of Materials Include File Combine** key field is used to determine how the match string in the include file is applied to the schematic.

For example, if you want the values in the include file to be matched to the contents of **Part Field 6**, you enter **6** in this key field.

In a design, you placed several capacitors, all with the value of 50pF in their **Part Value** fields. Some of them are made of a plastic material and others of a ceramic. You want the part list to show this distinction.

To do this, edit **Part Field 1** on the plastic capacitors to hold the value "Plastic," and edit **Part Field 1** on the ceramic capacitors to hold the value "Ceramic." Then, in **Key Fields** area, enter the number **1** in the **Bill of Materials Include File Combine** entry box.

When **Create Bill of Materials** runs on the design, all capacitors with a **Part Field 1** value of "Plastic" *and* a **Part Value** of "50pF" appear on one line of the part list, and all capacitors with a **Part Field 1** value of "Ceramic" *and* a **Part Value** of "50pF" appear on a different line of the part list.

Extract Source from Schematic PLD Part Combine and PLD Type Combine

Use these key fields in conjunction with **Extract Source from Schematic**, one of the processes in the **To PLD** transfer to **Programmable Logic Design Tools. Extract Source from Schematic** extracts information from a schematic for use with OrCAD's programmable logic device compiler and places it in a .PLD file. The **PLD Part Combine** identifies which of eight part fields holds the PLD part names. The PLD part name is placed immediately after the Part keyword in the OHDL source code. The part type identifies which of eight part fields holds the part architecture. The part architecture is placed immediately after the Type keyword in the OHDL source code. For more information on Part and Type, see *Programmable Logic Design Tools Language Guide.*

Example

The next example shows how these key fields are used by **Extract Source from Schematic.** Figure 1-20 shows part of a design with a PLD.

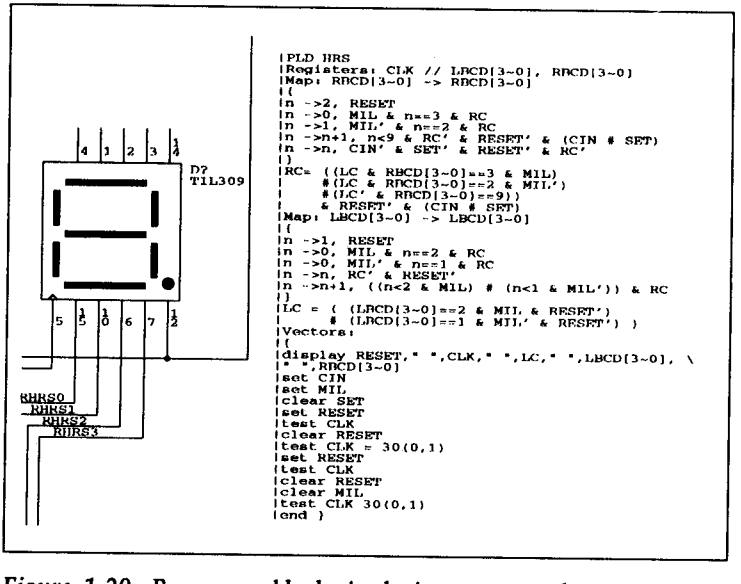


Figure 1-20. Programmable logic device source code.

Assume the **Extract Source from Schematic** key fields are configured as shown below.

```
Extract Source from Schematic
PLD Part Combine  2
PLD Type Combine  1
```

When you run **Extract Source from Schematic** on the schematic containing the part, it creates a file called HRS.PLD. The contents of HRS.PLD are shown in figure 1-21. In the HRS.PLD output, the **PLD Type Combine** key field (**1st Part Field**) is shown in ***bold italics***, and the **PLD Part Combine** key field (2nd Part Field) is shown in **bold**.

```
|"HRS"    1:CLK,
|          2:-,
|          3:-,
|          4:-,
|          5:-,
|          6:-,
|          7:-,
|          8:-,
|          9:MIL,
|         10:SET,
|         11:CIN,
|         13:RESET,
|         23:LC,
|         22:LBCD3,
|         21:LBCD2,
|         20:LBCD1,
|         19:LBCD0,
|         18:RC,
|         17:RBCD3,
|         16:RBCD2,
|         15:RBCD1,
|         14:RBCD0
|
|Value:    "HRS"
|Type:     "22V10"
|Part:     "PALC22V10-35"
|Library: "EXAMPLE.LIB"
|
|Title:    "Example schematic"
|Title:    " November  5, 1990"
|
|Registers: CLK // LBCD[3~0], RBCD[3~0]
|Map: RBCD[3~0] -> RBCD[3~0]
|{
|n ->2, RESET
|n ->0, MIL & n==3 & RC
|n ->1, MIL' & n==2 & RC
|n ->n+1, n<9 & RC' & RESET' & (CIN # SET)
```

Figure 1-21. ***Extract Source from Schematic*** *output (continues).*

```
|n ->n, CIN' & SET' & RESET' & RC'
|}
|RC= ((LC & RBCD[3~0]==3 & MIL)
|     #(LC & RBCD[3~0]==2 & MIL')
|     #(LC' & RBCD[3~0]==9))
|     & RESET' & (CIN # SET)
|Map: LBCD[3~0] -> LBCD[3~0]
|{
|n ->1, RESET
|n ->0, MIL & n==2 & RC
|n ->0, MIL' & n==1 & RC
|n ->n, RC' & RESET'
|n ->n+1, ((n<2 & MIL) # (n<1 & MIL')) & RC
|}
|LC = ( (LBCD[3~0]==2 & MIL & RESET')
|      # (LBCD[3~0]==1 & MIL' & RESET') )
|Vectors:
|{
|display RESET," ",CLK," ",LC," ",LBCD[3~0], \
|" ",RBCD[3~0]
|set CIN
|set MIL
|clear SET
|set RESET
|test CLK
|clear RESET
|test CLK = 30(0,1)
|set RESET
|test CLK
|clear RESET
|clear MIL
|test CLK 30(0,1)
|end }
```

*Figure 1-21. **Extract Source from Schematic output (continued).***

Check Electrical Rules matrix

The **Check Electrical Rules** matrix summarizes the rules that **Check Electrical Rules** uses when testing connections between pins, module ports, and sheet nets. This matrix is shown in figure 1-22.

The pins, module ports, and sheet nets are listed in columns and rows in the table. A test is represented by the intersection of a row and column. The intersection of a row and column is either empty, or contains a "W" or an "E." An empty intersection represents a valid connection, a W is a warning, and an E represents an error. You can toggle between these three settings by pointing to an intersection and clicking the mouse until the desired setting appears. To return all intersections to their default settings, select the **Set to Defaults** option.

Connections prefixed with an "m" are module ports. There are four types of module ports: input (mI), output (mO), bidirectional (mB), and unspecified (mU).

Connections prefixed with an "s" are sheet nets. As with module ports, there are four types of sheet nets: input (sI), output (sO), bidirectional, (sB), and unspecified (sU).

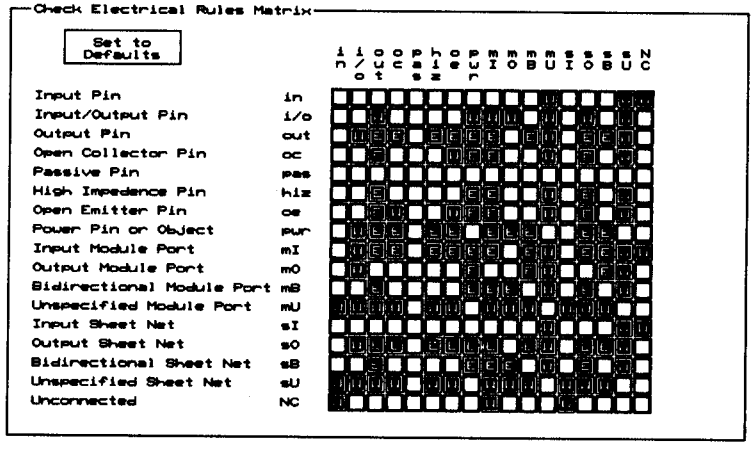


*Figure 1-22. **Check Electrical Rules Matrix** area on the **Configure Schematic Design Tools** screen.*

For definitions of pin types, see the **PIN Type** command description in *Chapter 2: Draft.*

Example To find the test result value when an output pin is connected to an input pin, use the OUT column (the third column in figure 1-20) and the IN row (the first row). This intersection is empty, which means this type of connection is acceptable. However, if you look at the intersection of the OUT column and the OUT row (the third row), you see an E, which indicates an output pin connected to an output pin is considered an error.

PART II: EDITORS

The heart of **Schematic Design Tools 386+** is **Draft**, an editor you use to create or modify schematics. You also use editors to edit or view text files and to access files containing reference information.

Part II describes editor tools and provides instructions for their use.

Chapter 2: *Draft* explains how to configure and run **Draft**. An alphabetical command reference gives detailed descriptions of each of **Draft**'s commands.

Chapter 3: *Guidelines for creating designs* explains how to name signals, place labels, handle module ports, buses, and power objects correctly so that when a netlist is produced (using **Create Netlist** or **Create Hierarchical Netlist**) satisfactory results are achieved.

Chapter 4: *Edit File* explains how to use **Edit File** to run the text editor of your choice.

Chapter 5: *View Reference* describes how to use **View Reference** to read supplemental reference material provided by OrCAD.



Draft

This chapter contains information needed to run **Draft**, the schematic editor at the heart of **Schematic Design Tools 386+**.

Within this chapter you will find execution and local configuration information, and a complete command reference. In the command reference section, commands are described in the order in which they appear in **Draft**'s main menu.

Execution

With the **Schematic Design Tools** screen displayed, select **Draft**. Select **Execute** from the menu that displays.

- If you have not specified a source file on **Draft**'s local configuration screen, **Draft** prompts:

```
load file?
```

  Enter the name of the worksheet to create or edit, or press <Enter> and select a name from the list that displays.

- If you have specified a source file in **Draft**'s local configuration, **Draft** loads the worksheet.

For instructions on how to specify a source file on **Draft**'s local configuration screen, see the next section of this chapter. For information about **Draft**'s commands, see the *Command reference* section in this chapter.

△ ***NOTE: Schematic Design Tools 386+** sets the root schematic of a design to the same name as the design name.*

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Draft**. Select **Local Configuration** from the menu that displays.

Select **Configure Draft**. Draft's configuration screen displays (figure 2-1).

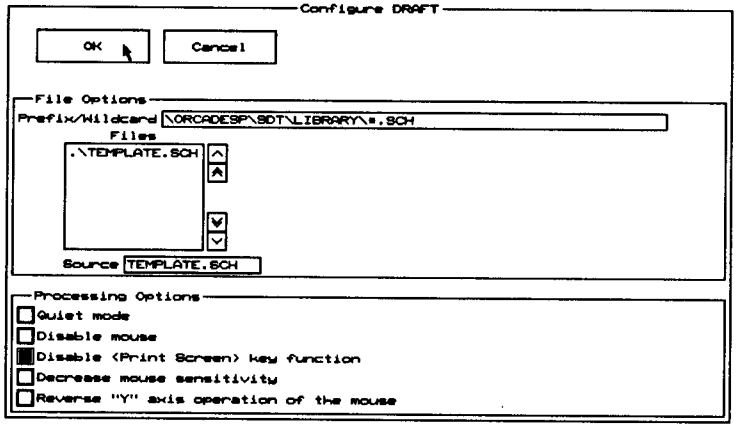


Figure 2-1. Draft's local configuration screen.

File Options

File Options specifies a worksheet for **Draft** to load.

Prefix/Wildcard

Enter a prefix and wildcard. These are described in the *ESP Design Environment User's Guide*. A list of files that match the selection criteria in this entry box displays in the **Files** list box.

➤ *Helpful hint . . .*

If you change the extension in the **Source** entry box and select **OK**, the extension in the **Prefix/Wildcard** text field changes to match the extension in the **Source** entry box when you again display **Draft**'s local configuration screen.

Files This box contains a list of the files matching the search filter entered in the **Prefix/Wildcard** entry box and those in the current design directory that match the wildcard. Files in the current directory have .\ before their names. Use the scroll buttons to scroll the list of files up and down.

When you see the file you would like to open, select it. Its filename displays in the **Source** entry box.

Source The **Source** is the filename of a worksheet to load. Either select a worksheet from the **Files** list box or enter the name of the worksheet.

△ *NOTE: If you do not enter a source name, **Draft** requests a filename when you run **Draft**.*

Processing Options You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on. Without this option, tracking information displays in the monitor box at the bottom of the screen.

- ❑ `Disable mouse`

  Disables the mouse. This option is normally used when debugging mouse problems while working with OrCAD Technical Support. It may also be required when running on older PC-compatible computers.

- ❑ `Disable <Print Screen> key function`

  Disables **Draft's** <Print Screen> key function. Use this option when you run other applications (usually RAM-resident) that use the <Print Screen> key. If this option is *not* selected, **Draft** uses the <Print Screen> key to capture hard copy output and blocks other uses.

- ❑ `Decrease mouse sensitivity`

  Slows the mouse down. Used for mouse devices that are too sensitive. For example, if you move your mouse a small distance and the pointer moves a large distance on the screen, select this option to make the pointer movement respond more closely to the mouse movement.

- ❑ `Reverse "Y" axis operation of mouse`

  Causes the mouse to respond differently. If this option is selected, the pointer moves up when you pull the mouse toward you, and moves down when you push the mouse away from you.

Command reference

The remainder of this chapter is a command reference for **Draft**, the schematic editor.

Commands are described in alphabetical order, the order in which they appear in **Draft**'s main menu. Main menu commands appear at the top of the first page describing the command. Many commands display other menus. Commands on these menus are described under the main menu command.

Some commands in the main menu appear in several other menus. These commands (such as **FIND**, **JUMP**, and **ZOOM**) are described at the main menu level only. When a command occurs in another menu, see the main menu description of the command for an explanation of its use.

Selecting commands

Select **Draft** commands in one of two ways:

- Press the first letter of the command name. Occasionally a menu has more than one command beginning with the same letter. When this happens, use the method of selecting commands explained below.
- Move the highlight bar over the command name, and press <Enter> or click the left mouse button.

Press <Esc> to return to the previous menu.

Locating commands

If you are not sure of the name of a command, use table 2-1 to quickly look up the command alphabetically or use table 2-2 to identify the task you want to accomplish and then identify the command.

Once you know the name of the command, look it up in the command reference.

Command	*Menu commands*		
AGAIN	*None*		
BLOCK	Move Drag Fixup Get	Save Import Export ASCII Import	Text Export Copy
CONDITIONS	*None*		
DELETE	Object	Block	Undo
EDIT	Edit Find	Jump Zoom	
FIND	*None*		
GET	*None*		
HARDCOPY	Destination File Mode	Make Hardcopy Width of Paper	
INQUIRE	*None*		
JUMP	A–H tags Reference	X location Y location	
LIBRARY	Directory	Browse	
MACRO	Capture Delete	Initialize List	Read Write
PLACE	Wire Bus Junction Entry (Bus) Label	Module Port Power Sheet Text Dashed Line	Trace Name Vector Stimulus NoConnect Layout
QUIT	Enter Sheet Leave Sheet Update File	Write to File Initialize Suspend to System	Abandon Edits Run User Commands
REPEAT	*None*		
SET	Auto Pan Backup File Drag Buses Error Bell Left Button	Macro Prompts Orthogonal Show Pins Title Block Worksheet size	X, Y Display Grid Parameters Repeat Parameters Visible Lettering Cursor Style
TAG	A–H tags		
ZOOM	Center In	Out Select	

*Table 2-1. **Draft** menu commands.*

Category	*Task*	*Select*
Entering and editing objects and data	Put parts, connections, and hierarchical sheets on your worksheet.	**PLACE**
	Get parts from loaded libraries.	**GET**
	Edit part attributes on the worksheet.	**EDIT**
	Identify a section of the worksheet.	**BLOCK**
	Erase objects or blocks of objects.	**DELETE**
Setting locations and conditions	Set the status of **Draft** parameters.	**SET**
	Identify and remember locations on the worksheet.	**TAG**
Navigating on the screen	Move the pointer to a specified location on the worksheet.	**JUMP**
	Locate a string of text characters.	**FIND**
	Change the location and amount of detail you see on the screen.	**ZOOM**
Repeating repetitive or complex tasks	Repeat the last main menu command.	**AGAIN**
	Define macros (recorded commands).	**MACRO**
	Duplicate the last entered object, label, or text string.	**REPEAT**
Showing status or other information	Display part list directories and parts in loaded libraries.	**LIBRARY**
	Display memory available for the worksheet, hierarchy buffer, and macro buffer.	**CONDITIONS**
	Display text associated with worksheet objects.	**INQUIRE**
Printing your schematic	Send a schematic to the printer or a file.	**HARDCOPY**
Leaving the program	Save your worksheet, enter or leave hierarchical worksheets, leave the root sheet, or leave **Draft**.	**QUIT**

*Table 2-2. **Draft** commands by function.*

AGAIN

AGAIN repeats the last *main menu* command executed. For example, if the last command you selected is **PLACE**, you may repeat **PLACE** by selecting **AGAIN**.

AGAIN repeats commands only from the main menu. For example, if you execute a command in the **PLACE** menu and then select **AGAIN**, the **PLACE** menu displays, ready for you to select another **PLACE** command.

BLOCK

Use **BLOCK** and its commands to manipulate specific areas of your worksheet. Select **BLOCK** to move, rubberband (stretch), make orthogonal (perpendicular to each other), duplicate, import, or export a section of a worksheet.

When you select **BLOCK**, the menu shown at right displays. The commands on this menu are described on the following pages.



BLOCK Move

BLOCK Move moves selected objects to another location on the worksheet.

Select **BLOCK Move**. **Draft** displays:

```
Begin  Find  Jump  Zoom
```

To select the objects to move, draw a box around them. Place the pointer where you want the corner of the box to start and select **Begin**. The **Begin** command in the command line is replaced with **End**:

```
End  Find  Jump  Zoom
```

Move the pointer. As you do, **Draft** draws a box. When the box encloses or intersects all of the objects you wish to move, select **End**. **Draft** selects the objects and displays:

```
Place  Find  Jump  Zoom
```

You can now move objects within or intersected by the box. The objects are drawn as outlined symbol shapes so that **Draft** can move them quickly. If you selected an area that contains many objects, only the box enclosing the area appears to move. **Draft** keeps the details of the objects in the box and displays them again when you place them in the new location.

Move the box to the new location. The objects within the area being moved still show at their original location. Only the outlined symbols within the selected area move.

Select **Place** to place the moved objects at the new location. **Draft** redraws the screen, placing the objects in their new location. **Draft** returns to the main menu level.

△ *NOTE: You may move and place a single object by selecting **BLOCK Move**, positioning the pointer inside the object, and then selecting **Begin** and **End**. You do not have to enclose the object in a box. The box can also be drawn as a horizontal line, a vertical line, or a single point, as long as at least part of the box intersects the object you want to move.*

BLOCK Drag

BLOCK Drag moves objects while maintaining connectivity. When you use this command, wires and buses appear to stretch as you move the block of objects on the worksheet. This effect is aptly called "rubberbanding." When you finally place the block of objects, the wires and buses are extended to maintain their original connectivity.

BLOCK Drag works the same as **BLOCK Move.**

To drag a bus, first turn on the **SET Drag Buses** option (see the **SET** command later in this chapter) to maintain bus connectivity.

BLOCK Fixup

Use **BLOCK Fixup** to "fix up" wires and buses, making them orthogonal (perpendicular to each other) by adding new wire or bus segments.

Select **BLOCK Fixup. Draft** displays the command line:

```
Pick  Find  Jump  Zoom
```

Use **Pick** to select the wire or bus to make orthogonal. Place the pointer so it touches either end of the wire or bus you want to make orthogonal and select **Pick** to select it. **Draft** displays the command line:

```
Drop  End  Find  Jump  Zoom
```

Move the pointer to the new endpoint. The end of the wire or bus moves with the pointer. In addition, **Draft** adds a new wire or bus segment that extends from the original wire or bus position to the new position.

Select **Drop** to "drop" a new wire or bus segment when you have it where you want it. You can continue dropping segments by selecting **Drop** as many times as desired.

Select **End** to drop the last segments when you have finished fixing up the wire or bus. **Draft** displays the **BLOCK Fixup** command line, so you can "fix up" another wire or bus. Press <Esc> to return to the main menu level.

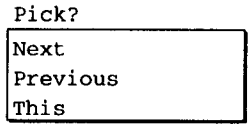
△ *NOTE: Use **BLOCK Fixup** only for adding segments when straightening non-orthogonal wires and buses. Use **BLOCK Drag** for cleanups that do not need additional segments.*

Multiple wires or buses

If a node has more than one wire or bus connected, a menu displays that you use to select either **Drag All** or **Pick One** wire or bus.

Select **Drag All** to drag all the wires or buses attached to a common point.

Select **Pick One** to choose one wire or bus from those connected to the node. **Draft** displays the menu shown at right.

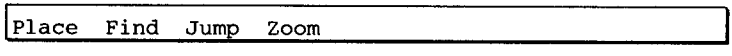


The wire or bus currently selected with **Pick One** is highlighted in another color. Select **Next** or **Previous** to select one of the other wires or buses. When the desired wire or bus is selected, select **This** to begin the "fix up."

BLOCK Get

BLOCK Get retrieves objects saved in a buffer (using **BLOCK Save**, described later) and places them on the worksheet.

Select **BLOCK Get**. **Draft** displays:



A box containing the previously saved objects displays. The pointer is attached to this box. Move the box to the desired location. Place the objects on the worksheet by selecting **Place**.

The box containing the objects remains on the screen. You can place as many copies of the objects as desired. Simply move the box and select **Place**.

BLOCK Save

BLOCK Save stores a copy of a group of objects in a buffer so that they can be duplicated in another area of the worksheet.

Select **BLOCK Save. Draft** prompts:

```
Begin  Find  Jump  Zoom
```

To select the objects to be saved, draw a box around them. Place the pointer where you want the corner of the box to start and select **Begin.** The command line changes to:

```
End  Find  Jump  Zoom
```

Notice that the **Begin** command has changed to **End.**

Move the pointer. As you do this, **Draft** draws a box enclosing objects on the worksheet. When the box encloses all of the objects to save, select **End. Draft** saves a copy of the objects within the box in a buffer and returns to the main menu level.

Saved objects are placed on the worksheet using **BLOCK Get** (described previously).

△ *NOTE: The buffer used to save objects is also used by* ***BLOCK Move*** *and* ***BLOCK Drag****. Objects saved with* ***BLOCK Save*** *are lost after using either* ***BLOCK Move*** *or* ***BLOCK Drag****.*

To save objects and still use ***BLOCK Move*** *or* ***BLOCK Drag*** *in your editing session, use* ***BLOCK Export*** *rather than* ***BLOCK Save*** *to store worksheet objects.*

BLOCK Import

BLOCK Import retrieves objects stored in other files (with **BLOCK Export**, described on the next page) and places them in your current worksheet.

Select **BLOCK Import**. **Draft** displays:

```
File to Import?
```

Enter the path and filename of the file to import.

△ *NOTE: If a pathname is entered in the **Worksheet Prefix** entry box on the **Configure Schematic Design Tools** screen, **Draft** uses that pathname. If you do enter a pathname here, **Draft** ignores the pathname specified on the **Configure Schematic Design Tools** screen.*

Draft displays:

```
Place  Find  Jump  Zoom
```

An outline of the block and objects displays. Position the pointer on the worksheet where you want to place the contents of the file. Select **Place** to put the contents of the imported file on the worksheet. **Draft** places the imported objects on the worksheet. **Draft** returns to the main menu level.

△ *NOTE: When you position the pointer, be sure to allow adequate room below and to the right of the pointer to prevent placing parts off the edge of the worksheet.*

BLOCK Export

BLOCK Export saves a copy of a group of objects in a file.

Select **BLOCK Export. Draft** displays:

```
Begin  Find  Jump  Zoom
```

To select the objects to be saved, draw a box around them. Place the pointer where you want the corner of the box to start and select **Begin.** The starting corner is also the starting corner of the block when it is imported with **BLOCK Import.**

Draft displays:

```
End  Find  Jump  Zoom
```

Notice that the **Begin** command has changed to **End.**

Move the pointer. As you do this, **Draft** draws a box. When the box encloses all of the objects you wish to save, select **End. Draft** displays:

```
Export filename?
```

Enter the path and filename to which to export the objects.

△ *NOTES: If a pathname is entered in the* ***Worksheet Prefix*** *entry box on the* ***Configure Schematic Design Tools*** *screen,* ***Draft*** *uses that pathname unless you enter one here. When you enter a pathname here,* ***Draft*** *ignores the pathname specified on the* ***Configure Schematic Design Tools*** *screen.*

When you position the pointer, begin in the upper left corner and drag down and to the right.

Draft saves a copy of the objects enclosed and intersected by the box in this file and returns to the main menu level. The objects in this file can be placed back on the worksheet using **BLOCK Import** (described on the previous page).

BLOCK ASCII Import

BLOCK ASCII Import retrieves text stored in a text file and places it on your worksheet. You can create the text file with any editor that can read and write DOS text files.

Select **BLOCK ASCII Import**. **Draft** displays:

```
File to Import?
```

Enter the path and filename of the file to import. **Draft** displays the command line:

```
Place  Find  Jump  Zoom
```

Position the pointer on the worksheet where you want to place the text. Select **Place**. The text file's contents are placed on the worksheet just to the right of the pointer. Each new line is placed below the previous line.

Draft returns to the main menu level.

△ *NOTES: If your text editor supports special formatting and layout of documents, you should make sure these special features are not used to create the text file.*

***BLOCK ASCII Import** expects data in the format created with the **BLOCK Text Export** command described on the following page.* Do not *use **BLOCK ASCII Import** to import a file created by the **Decompile Schematic** tool.*

BLOCK Text Export

BLOCK Text Export saves a copy of selected text in a text file. You can then edit this file with any editor that can read and write ASCII text. When you finish editing the text, you can place the contents of the text file back on the worksheet with the **BLOCK ASCII Import** command (described earlier).

Select **BLOCK Text Export. Draft** displays:

```
Begin  Find  Jump  Zoom
```

To define the text to export, draw a box around it. Place the pointer where you want the corner of the box to start and select **Begin. Draft** displays:

```
End  Find  Jump  Zoom
```

Notice that the **Begin** command has changed to **End.** Move the pointer. As you do this, **Draft** draws a box enclosing the text. When the box encloses all of the text you wish to export, select **End. Draft** displays:

```
Text Export Filename?
```

Enter the path and filename in which to export the text. A copy of the text enclosed and intersected by the box is saved in this file.

Draft returns to the main menu level.

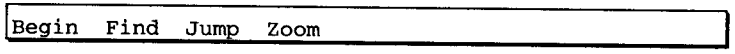
△ *NOTES: Only text objects can be exported. Labels or other characters associated with other objects cannot be exported.*

BLOCK Text Export writes data in the format expected by the BLOCK ASCII Import command described on the preceding page. Do not *use Compile Schematic with a file created with the BLOCK Text Export command.*

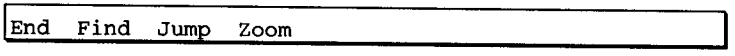
BLOCK Copy

BLOCK Copy duplicates a portion of your schematic.

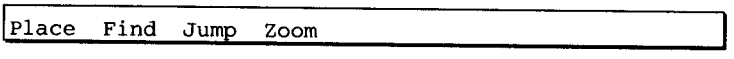
Select **BLOCK Copy**. **Draft** displays:



To define the area of the schematic to duplicate, draw a box around it. Place the pointer where you want the corner of the box to start and select **Begin**. **Draft** displays:



Notice that the **Begin** command has changed to **End**. Move the pointer. As you do this, **Draft** draws a box enclosing the area. When the box encloses all of the area you wish to copy, select **End**. **Draft** displays:



Notice that the enclosed area is highlighted. Move the pointer to the location where you want the copy and select **Place**.

Draft places the copied objects at the new location and returns to the main menu level.

CONDITIONS

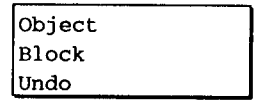
CONDITIONS monitors the amount of memory used by the current worksheet. When you select **CONDITIONS, Draft** displays the worksheet memory size, as shown in this example:

```
::: press any key to continue :::
Worksheet Memory Size = 2008 byte
```

Note that even a blank worksheet uses memory for border and title block information.

DELETE

DELETE and its commands erase objects or blocks of objects. Use the **Undo** command in case you accidentally delete an object and would like to restore it to its original position.

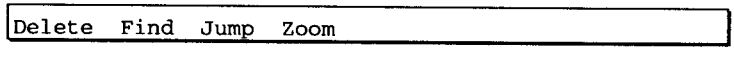


When you select **DELETE**, the menu shown above displays. Each of the commands on this menu are described on the following pages.

DELETE Object

DELETE Object erases an object from the worksheet.

Select **DELETE Object**. **Draft** displays:

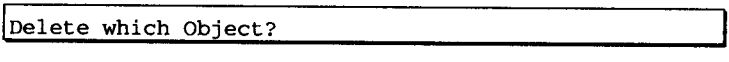


To delete an object, place the pointer on the object you want to delete and select **Delete**.

△ *NOTE: You must place the pointer on or within the body of a part to delete it.*

If you want to delete one of two intersecting wires and you have placed the pointer at their intersection, the first wire drawn is the first deleted. To delete the last wire drawn, move the pointer away from the intersection along the wire to delete and select **Delete**.

If the pointer is pointing to more than one type of object, **Draft** displays:



Draft displays a menu listing objects to delete. Select the object from the menu to delete it.

After deleting an object from the worksheet, **Draft** returns to the **DELETE Object** command line, where you may continue to delete objects.

To return to the main menu level and redraw the worksheet, press <Esc>.

DELETE Block

DELETE Block deletes a block of objects on a worksheet.

Select **DELETE Block. Draft** prompts:

```
Begin  Find  Jump  Zoom
```

To select the objects to delete, draw a box around them. Place the pointer where you want the corner of the box to start and select **Begin.** The command line changes to:

```
End  Find  Jump  Zoom
```

Notice that the **Begin** command has changed to **End.**

Move the pointer. As you do this, **Draft** draws a box enclosing objects on the worksheet. When the box encloses or intersects all of the objects you wish to delete, select **End. Draft** deletes objects within or intersected by the box. After deleting the block of objects, **Draft** returns to the main menu level.

DELETE Undo

DELETE Undo restores your last deletion.

Select **DELETE Undo.** The object or block of objects deleted with the last **DELETE Object** or **DELETE Block** command is restored on your schematic.

EDIT

Use **EDIT** to:

- Edit labels, text, module ports, power objects, sheets, part reference designators, part values, part fields, and the title block.
- Change pin names and numbers on devices with multiple parts per package.
- Move part reference designators, part values, and part fields to new locations on the worksheet.
- Make part reference designators, part values, or part fields visible or invisible.
- Change the style of parts, power objects, module ports, labels, and text.
- Change the orientation of parts, text, and labels.
- Change the size of text and labels.
- Edit sheets, sheet names, sheet nets, sheet net types, and sheet filenames. You can use **EDIT** to add sheet nets or to delete sheet nets.

Select **EDIT**. **Draft** displays:

```
Edit  Find  Jump  Zoom
```

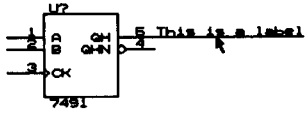
To edit an object, place the pointer on the object you want to edit and select **Edit**. **Draft** displays a menu that's appropriate to the object you are editing. These menus are described on the following pages.

Editing techniques

When editing text fields, use these techniques:

- Position the cursor with the <←> and <→> keys, or the <Home> and <End> keys, or the mouse
- Erase characters with <Backspace> or <Delete>
- Add new characters with the alphanumeric keys

Editing labels



To edit a label, place the pointer immediately below and within the label, as shown in the figure at the left. Select **Edit. Draft** displays the menu shown at right.

```
Edit Label
Name
Orientation
Larger
Smaller
```

Name

Select **Name** to edit the name of a label. After selecting **Name**, the prompt "Name?" displays followed by the current label name.

Edit the name using the editing techniques described at the beginning of the **EDIT** section. When you finish, press <Enter>. The edited name displays on the worksheet.

Orientation

Select **Orientation** to specify the direction of the baseline of the label's text.

Horizontal and **Vertical** orient the text to the bottom of the screen.

Up rotates the label 90° counterclockwise, orienting the text to the right edge of the schematic.

Over rotates the label 180° counterclockwise, orienting the text to the top edge of the schematic.

Down rotates the label 270° counterclockwise, orienting the text to the left edge of the schematic.

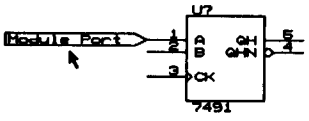
Larger

Select **Larger** to make the label's character size larger. You may select **Larger** more than once until the text is as large as you desire.

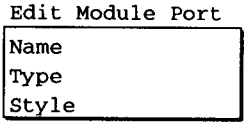
Smaller

Select **Smaller** to make the label's character size smaller. As with **Larger**, you may select **Smaller** multiple times, but the default text size is the smallest text size.

Editing module ports



To edit a module port, place the pointer within the module port symbol and select **Edit. Draft** displays the menu shown at right.



Name Select **Name** to edit a module port name. **Draft** displays the prompt "Module Port Name?" followed by the current module port name.

Edit the module port name using the editing techniques described at the beginning of the **EDIT** section. When you finish editing, press <Enter>. The module port name displays on the worksheet.

Type Select **Type** to change the type of module port (figure 2-2) Select the type of module port: **Input, Output, Bidirectional,** or **Unspecified.**

Style Select **Style** to change how the module port looks (figure 2-2). Choose from: **Right** pointing, **Left** pointing, **Both** pointing, or **Neither** pointing.

Figure 2-3 shows the different types of module ports and their default styles. A module port's style is independent from its type. For example, a "both pointing" style does not mean that the module port is bidirectional.

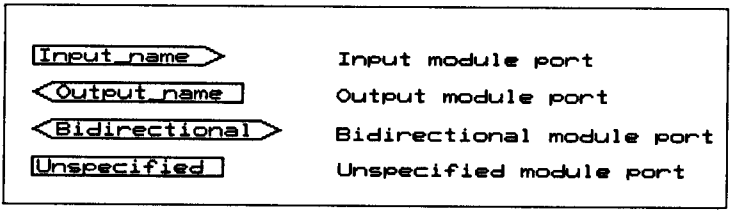
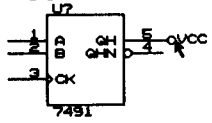


Figure 2-2. Types of module ports and their default styles.

Editing power objects



A power object is a schematic symbol that represents some type of power supply connected to the design.

```
Edit Power
Name
Type
Orientation
```

To edit a power object, place the pointer on a power object and select **Edit**. **Draft** displays the menu shown above.

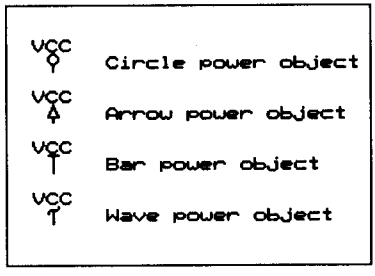
Name

Select **Name** to edit the name of the power object. **Draft** displays the prompt "Power Name?" followed by the object's current name.

Edit the name using the editing techniques described at the beginning of the **EDIT** section. When you finish, press <Enter>. The name displays on the worksheet.

Type

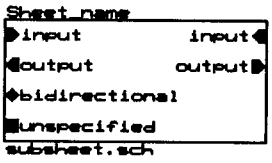
Select **Type** to change the type of power object. The different types (shown in the figure at the right) are **Circle**, **Arrow**, **Bar**, or **Wave**.



Orientation

Select **Orientation** to change the orientation to **Top**, **Bottom**, **Left**, or **Right**.

Editing sheet symbols



Sheet symbols are used in hierarchical designs to represent references to other schematic worksheets.

To edit sheet symbols, place the pointer within a sheet symbol boundary and select **Edit**. **Draft** displays:

```
Add-Net Delete Edit Name Filename Move Size Zoom
```

After you select **Edit**, pointer movement is restricted to the border of the sheet symbol. This helps you place the pointer at sheet net name locations.

Add-Net

Select **Add-Net** to add net connections between worksheets. To add a net connection to a sheet symbol, move the pointer to an appropriate position and select **Add-Net**.

Draft displays the prompt "Net Name?" Enter the desired net name. A menu displays listing the types of net symbols: **Input**, **Output**, **Bidirectional**, and **Unspecified**. Figure 2–3 shows the types of net symbols. Select the appropriate type.

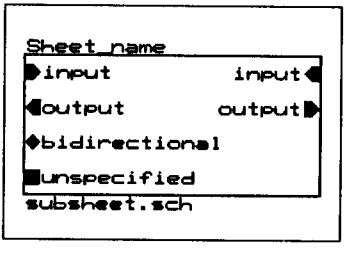


Figure 2-3. Net symbol types.

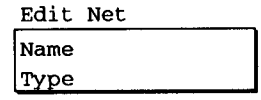
Notice the input net symbol points into the sheet symbol, and the output net symbol points out of the sheet symbol.

Delete

To delete a net name, place the pointer on the net name and select **Delete**.

Edit

To edit a net's name or type, put the pointer on the net name and select **Edit**. The menu shown at right displays.



- Select **Name** to change the name of the net. The net name displays on the prompt line. Edit it using the editing techniques described at the beginning of the **EDIT** section. When you finish, press <Enter>.
- Select **Type** to change the type of the net. A menu displays listing the types of net symbols: **Input, Output, Bidirectional,** and **Unspecified**. Figure 2–3 shows each of these types of net symbols. Select the appropriate type.

Name

Use **Name** to edit the name of a sheet symbol, located at the top of the sheet. Initially, the sheet name is a question mark. Typically, names identify the function of the worksheet represented by the sheet symbol (such as "Memory Array" or "Dynamic RAM Refresh Circuitry").

To edit the sheet symbol's name, select **Name. Draft** displays "Sheet name?" and shows the current name.

Edit the name using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the name of the sheet symbol, press <Enter>.

Press <Esc> to cancel any changes to the sheet name.

Filename

Use **Filename** to edit the name of the worksheet represented by a sheet symbol.

Draft automatically creates a filename based on the date and time of day, ensuring no two filenames will be alike. This name displays at the bottom of the sheet symbol as soon as it is created.

To edit the filename, select **Filename. Draft** displays the prompt "Filename?" followed by the current filename.

Edit the filename using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the filename, press <Enter>.

Move Use **Move** to move a net connection. Move the pointer to the net connection you want to move, select **Move**, move to the new location, and select **Place** or press <Enter>.

Size Select **Size** to change the size of the sheet symbol displayed on the screen. **Draft** displays:

```
End  Jump  Zoom
```

Draft moves the pointer to the lower right corner of the sheet symbol. To change its size, move the pointer until the symbol's size is satisfactory, then select **End**.

Editing parts

Use **Edit Part** to:

- Edit and move part reference designators, values, and fields
- Select other parts in library parts with multiple parts per package
- Change the orientation of the symbol.

Figure 2-4 illustrates a library part with its default reference designator and part value.

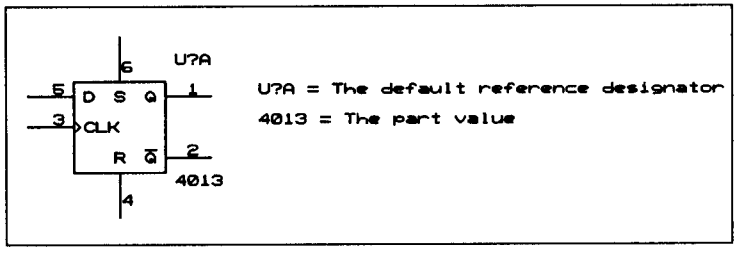


Figure 2-4. CMOS 4013 library part.

To edit a part, place the pointer within the part symbol boundary and select **Edit**. For a description of a symbol boundary, see *The outline symbol* under the **GET** command description in this chapter.

When you select **Edit, Draft** displays the menu shown at right.

Which Device only displays when you edit a device with multiple parts per package.

```
Reference
Part Value
1st Part Field
2nd Part Field
3rd Part Field
4th Part Field
5th Part Field
6th Part Field
7th Part Field
Module Value
SheetPart Name
Orientation
Which Device
```

Reference

Select **Reference** to edit or move the reference designator values of library parts placed on the worksheet. **Draft** displays the menu shown at right.

```
Reference
Name
Location
Visible
```

Examples of reference designators are: U1, U2A, Q6, R1, R2, and C12.

See *Chapter 6: Annotate Schematic* for a method of automatically incrementing reference designators and changing the corresponding pin numbers of parts placed on the worksheet.

Name

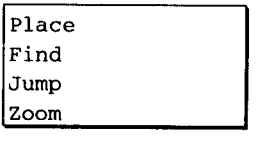
Select **Name** to edit the name of a reference designator. The prompt "Reference?" displays. If the part already has a reference designator name assigned, it also displays.

Edit the name using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the name, press <Enter>.

△ *NOTE: Some library parts contain more than one part per package. For example, a 74LS04 hex inverter contains six parts. The reference designators for the six parts may be U?A, U?B, U?C, U?D, U?E, and U?F. The last letter in the reference designator tells which one of the parts in the package you are editing. You cannot edit this last letter with the* ***EDIT Edit Name*** *command. To edit a different part in a multiple-part package, use the* ***EDIT Edit Which Device*** *command.*

Location

Select **Location** to change the reference designator's location. **Draft** highlights the part's reference designator, value, and part fields, and displays the menu shown at right.



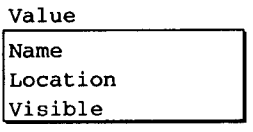
You may move the reference designator anywhere in the worksheet using the arrow keys or the mouse. Select **Place** to place the reference designator at the new location.

Visible

Select **Visible** to choose whether the reference designator appears on the screen and on paper prints and plots. When you select **Visible**, a menu displays **Yes** and **No**. **Yes** makes the reference designator visible; **No** makes it invisible.

Part Value

Select **Part Value** to edit or move the part values of components on the worksheet. Examples of part values are: 100K, 1N4004, .01 uf, 2N2222, and 80386.



When you select **Part Value**, **Draft** displays the menu shown above.

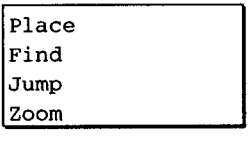
Name

Select **Name** to edit a part value. The prompt "Value?" displays. If the part already has a value assigned, it also displays.

Edit the value using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the part value, press <Enter>.

Location

Select **Location** to change the part value's location. **Draft** highlights the part's reference designator, value, and part fields and displays the menu shown at right.



You may move the part value anywhere in the worksheet using the arrow keys or mouse. Select **Place** to place the part value in the new worksheet location.

Visible

Use **Visible** to choose whether the part value appears on the screen and on paper prints and plots. When you select **Visible**, **Draft** a menu displays **Yes** and **No**. **Yes** makes the part value visible; **No** makes it invisible.

1st Part Field through 7th Part Field, Module Value

Eight part fields may be placed on the schematic worksheet by selecting them from the menu. Use **1st Part Field** through **7th Part Field** and the **Module Value** to add information about the part (such as tolerance, part number, vendor information, and so on) to the schematic. When you select one of the eight part fields, **Draft** displays the menu shown above.

```
1st Part Field
Name
Location
Visible
```

Name

Select **Name** to edit the information in a part field. The prompt "xx Part Field?" displays. If the part field already contains information, it also displays here.

Edit the information using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the part field, press <Enter>.

△ ***NOTE:*** *If you changed the part field names on the* ***Configure Schematic Design Tools*** *screen,* ***Draft*** *displays their new names in menus and prompts.*

Location

Select **Location** to change the location of the part field on the worksheet. **Draft** highlights the part's reference designator, value, and part fields and displays the menu shown at right. Use the arrow keys or the mouse to move the part field anywhere on the worksheet. Select **Place** to place the part field in the new location.

```
Place
Find
Jump
Zoom
```

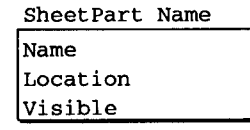
Visible

Select **Visible** to choose whether the part field appears on the screen and on paper prints and plots. When you select **Visible**, a menu displays **Yes** and **No**. **Yes** makes the part field visible; **No** makes it invisible.

SheetPart Name

A sheet part name is a filename. Once you assign a part a sheet part name, it is no longer a primitive part, but functions as a subsheet, which is a symbolic representation of another worksheet. The sheet part name can also be used to control whether **Draft** and other tools descend into the sheet part.

Select **SheetPart Name** to edit or move an object's sheet part name or change its visibility. When you select **SheetPart Name, Draft** displays the menu shown at right.



Using the sheet part name, you can tell INET that a netlist exists for the sheet, and prevent INET from creating a netlist for that worksheet. This is useful for building a hierarchy of netlists from different sources and for bottom-up or parallel design involving externally created incremental netlist files.

A hierarchical reference to a netlist is referred to as an *external view*. To create an external view, enter a sheet part name of the form `EXTERNALVIEW=`*fileName.ext*.

External view example

Suppose that you have created an FPGA design, optimized it using place and route software, and created a post-route simulation netlist file (MY_FPGA.INF). You want to simulate the FPGA in conjunction with the components of a PC board, but the original FPGA design schematic no longer represents the optimized netlist. To prevent INET from ignoring the optimized netlist, you edit the sheet part name to be `EXTERNALVIEW=MY_FPGA.INF`. This tells INET that the FPGA part refers to a netlist rather than the underlying schematic. It is important to note that the netlist file MY_FPGA.INF must be located in the same directory as the schematic.

For further information on using external views, see the *Digital Simulation Tools 386+ Reference Guide Supplement*.

For more information about sheet parts and how they differ from the sheet symbol and sheetpath parts, see *Chapter 9: Tips and Techniques* in the *Schematic Design Tools 386+ User's Guide.*

Name

Select **Name** to edit a sheet part name. The prompt "SheetPart Name?" displays. If the part already has a sheetpart name assigned, it also displays.

The sheet part name is the name of a schematic file, or the EXTERNALVIEW keyword followed by an equal sign (=) and the name of a schematic file. If the file is not in the current design directory, specify the full pathname.

Edit the sheet part name using the editing techniques described at the beginning of the **EDIT** section. When you finish editing the sheet part name, press <Enter>.

Location

Select **Location** to change the location of the sheet part name. **Draft** highlights the part's reference designator, value, part fields, and sheet part name and displays the menu shown at right. You may move the sheetpart name anywhere in the worksheet using the arrow keys or mouse. Select **Place** to place the sheetpart name in the new location.

```
Place
Find
Jump
Zoom
```

Visible

Use **Visible** to choose whether the sheet part name appears on the screen and on paper prints and plots. When you select **Visible, Draft** displays **Yes** and **No. Yes** makes the sheetpart name visible; **No** makes it invisible.

Orientation

Select **Orientation** to change the view of a part. **Draft** displays the following command line:

```
Rotate Convert Normal Up Over Down Mirror Zoom
```

Rotate

Rotate turns the part 90° counterclockwise from its current position.

Convert

Use **Convert** to change the form of the part.

Convert only displays when editing a part that contains its normal representation as well as a DeMorgan form. Parts with DeMorgan form actually have two versions of the part in the library. For example, the 74LS02 contains its standard form (a NOR) gate and a DeMorgan form (an inverted AND gate).

See *Chapter 20: Edit Library* for an explanation of how a part is given a DeMorgan form.

Normal

Normal returns a rotated part to its original orientation, as created in the part library. **Normal** also cancels the effect of the **Convert** and **Mirror** commands.

Up

Up rotates a part 90° counterclockwise from its normal position, equivalent to rotating it once from its normal position.

Over

Over rotates a part 180° counterclockwise, equivalent to rotating it twice from its normal position.

Down

Down rotates a part 270° counterclockwise, the equivalent of rotating it three times from its normal position.

Mirror

Mirror displays a mirror-image of a part. Mirroring is along the horizontal axis.

Which Device

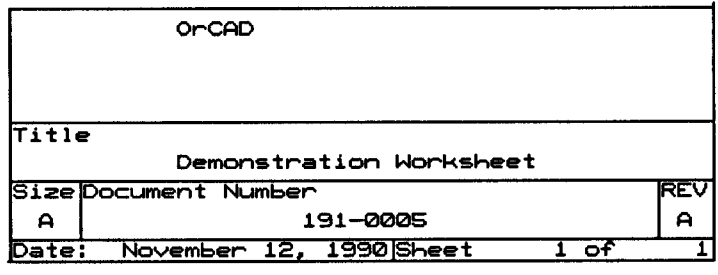
Which Device only displays when you edit a part containing more than one part per package. An example would be a 74LS04 hex inverter, which contains six inverters. To select a different part in the package, select **Which Device** and then select the device letter of the part to edit.

Editing the title block

To edit title block information, place the pointer inside the title block and select **Edit**. The title block is in the lower right worksheet corner. **Draft** displays the menu shown at right.

```
Edit title block
Revision code
Title of sheet
Document number
Sheet number
Number of sheets
Organization name
1st Address Line
2nd Address Line
3rd Address Line
4th Address Line
```

If a field is empty, simply enter the appropriate information. If the field already contains information, edit the information using the editing techniques described at the beginning of the **EDIT** section.



For more information about the title block, see:

- *Worksheet Options* in *Chapter 1: Configure Schematic Tools*
- *Title block tips* in *Chapter 9: Tips and Techniques* in the *Schematic Design Tools 386+ User's Guide*

Revision Code

Select **Revision Code** to add or edit the revision code (three characters maximum). **Draft** displays the prompt "Revision Code?"

Title of Sheet

Select **Title of Sheet** to add or edit a title (44 characters maximum). **Draft** displays the prompt "Title of Sheet?"

Document Number

Select **Document Number** to edit or add a document number (36 characters maximum). **Draft** displays the prompt "Document Number?"

Sheet Number Select **Sheet Number** to add or edit a sheet number (any number up to 32767). **Draft** displays the prompt "Sheet Number?" This can also be calculated automatically by **Annotate Schematic.**

Number of Sheets Select **Number of Sheets** to add to the number of worksheets (any number up to 32767). **Draft** displays the "Number of Sheets?" prompt. This can also be calculated automatically by **Annotate Schematic.**

Organization Name Select **Organization Name** to add or edit an organization name (up to 44 characters). **Draft** displays the prompt "Organization Name?"

Address Lines Select the address line to edit. Each line can be up to 44 characters long. **Draft** displays the prompt "*xxx* Address Line?," where *xxx* is either 1st, 2nd, 3rd, or 4th, depending on the address line you are editing.

△ *NOTE: If you are using an ANSI title block (defined on the* ***Configure Schematic Design Tools*** *screen), two additional areas are included in the title block:* ***CAGE CODE*** *and* ***SCALE****. You must use the* ***PLACE text*** *command to make an entry in these areas.*

△ *NOTE: See the PLACE command description in this chapter for a definition of the syntax format of the objects described on this page. See* Digital Simulation Tools Reference Guide *for information on the use of the first three objects.*

Editing stimulus objects



To edit a stimulus object, place the pointer at the location where the stimulus object connects to the wire or pin on which it was placed. Select **Edit**. The prompt "Stimulus?" displays followed by the current value.

Edit the value using the editing techniques described at the beginning of the **EDIT** section. When you finish editing, press <Enter>.

Editing trace objects



To edit a trace object, place the pointer at the location where the trace object connects to the wire or pin on which it was placed. Select **Edit**. The prompt "Trace Name?" displays followed by the current name.

Edit the name using the editing techniques described at the beginning of the **EDIT** section. When you finish editing, press <Enter>.

Editing vector objects



To edit a vector column, place the pointer at the location where the vector object connects to the wire or pin on which it was placed. Select **Edit**. The prompt "Vector Column?" displays followed by the current value.

Edit the vector column using the editing techniques described at the beginning of the **EDIT** section. When you finish editing, press <Enter>.

Editing layout objects



To edit the message associated with a layout object, place the pointer below the layout symbol and select **Edit**. The prompt "Layout Directive?" displays followed by the current message.

Edit the message, and press <Enter>.

FIND

FIND locates a string of text characters anywhere in a worksheet and places the pointer at the object containing the search string. A search string can be any number of characters identifying any of the following items:

- Module ports
- Labels
- Reference designators
- Part values
- Data stored in a part field
- Sheet symbol names
- Power objects
- Text

You must specify a complete character string. For example, if the part value "74LS00" exists in the worksheet and you specify "LS" as the search string, **Draft** displays the message "ERROR: Not Found: LS." However, if you specify the complete string "74LS00," **Draft** places the pointer at the object containing the string.

Select **FIND**. **Draft** displays "Find?" Enter the character string you want to find. The character string can be entered in upper or lower case; **FIND** is not case sensitive. **Draft** searches the worksheet for the desired character string and places the pointer near it.

The next time you select **FIND**, **Draft** shows the previous string on the prompt line. To search for a new string, edit the previous entry by positioning the cursor with the <←> and <→> keys and the <Home> and <End> keys, erasing characters with <Backspace> or <Delete>, and adding new characters with the alphanumeric keys. When you finish entering the new string, press <Enter>.

If you are searching for a string identical to the last string (for example, you want to find all 200 Ω resistors on the worksheet), press <Enter> with the current string name on the prompt line. **Draft** remembers the location of the previous string and searches for the next one.

If you select **FIND** after finding the last occurrence of the string, **Draft** "wraps" to the first occurrence of the string. If there is only one occurrence of the string, **FIND** returns repeatedly to the string when the command is selected.

△ ***NOTES:*** *FIND works only in the worksheet you are editing.*

To erase the previous string, select ***FIND*** *and press <Esc>. When* ***FIND*** *is selected again, the field is cleared.*

GET

GET retrieves parts from the library database and places them in the worksheet as normal, rotated, or converted symbols. There are two methods you can use with **GET** to retrieve parts.

Method 1	Method 2
Select **GET**. **Draft** displays **"Get ?"** Enter the desired part name *exactly* as it appears in the library directory. **Draft** searches the libraries you specified when you configured **Schematic Design Tools** and finds the part you requested. Once it finds the part, **Draft** puts its outline on the screen. If there are two parts with the same name in the configured libraries, **GET** *always* retrieves the first one. If the name typed does not match any part name in the library directory, **Draft** shows an error message. To verify the spelling of a part name, use the **LIBRARY Directory** command.	Select **GET**. **Draft** displays **"Get ?"** Press <Enter>. **Draft** displays a list of libraries. These are the libraries you specified when you configured SDT. Place the highlight on the library name you want to get a part from, or begin typing the name. You can also use the <Page Up>, <Page Down>, <Home>, and <End> keys to move through the list. After you select a library, **Draft** displays a menu listing the library parts. Move the highlight to the part name you want, then press <Enter> to retrieve the part. **Draft** puts the part's outline on the screen.

Once the part outline displays, you need to place it on the worksheet. For information on placing parts on the worksheet, see *Rotating and placing parts* in this section.

△ *NOTE: GET searches libraries in sequence, as explained in* Library Options *of* Chapter 1: Configure Schematic Tools, *for the first part with the name you specified. To verify that the part on the schematic is from the library you intend, Inquire on the part.*

Getting a part by entering a part suffix

If you are using Method 1 (described on the previous page), you can select library part numbers created with a prefix and shorthand string (see *Prefix definition* in *Chapter 17: About libraries*) from the library by entering the suffix. For example, suppose you want to retrieve a 74LS27 from the library. After selecting **GET**, enter any of the following examples to retrieve the part:

Get? 74LS27 — In this example, the entire part name is used to retrieve the part.

Get? LS27 — In this example, the prefix "LS" (which is the shorthand string for "74LS") is combined with the suffix "27." **Draft** retrieves the part 74LS27.

Get? 27 — In this example, only the suffix "27" is entered. **Draft** looks through all of the configured libraries and displays a menu of all available parts with "27" as a suffix. Select the desired part from the menu.

The outline symbol

After getting the part from the library, the screen shows an outline of the part symbol (figure 2-5). The outline symbol shows the size and shape of the part, but no detail. Its function is to let **Draft** move the part quickly around the worksheet.

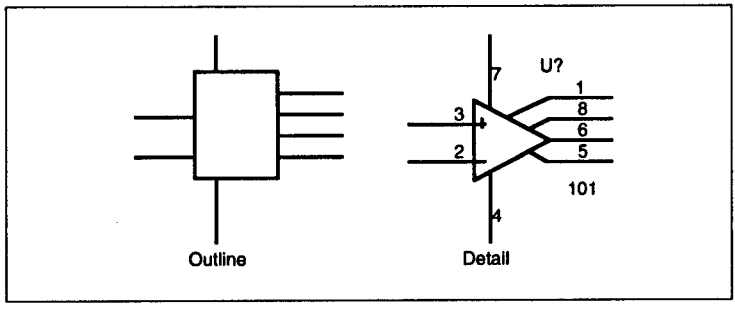


Figure 2-5. A part's outline symbol and its detailed symbol.

If the outline symbol remains stationary for a short time, Draft shows the detailed part symbol enclosed by the outline symbol. Use this feature to view the position the part will have when it is placed in the worksheet.

Rotating and placing parts

With the part selected and the outline symbol displayed, **Draft** displays this command line:

```
Place Rotate Convert Normal Up Over Down Mirror
```

Move the symbol to where you want to place it. Use these commands to rotate or place the part in the worksheet.

△ *NOTE: The **Convert** command only displays for parts that have a convert.*

Place

Select **Place** to place the part on the worksheet.

After you place a part on the worksheet, **Draft** keeps the same part selected, so that you can repetitively place duplicates of a part without repeating the selection process. When you have placed all the parts, press <Esc> to return to the main command level.

When an outline symbol is placed over a copy of the same object already placed on the worksheet, the object may disappear. Moving the outline symbol displays the placed part.

Rotate

Select **Rotate** to turn a part counterclockwise 90° (figure 26). Each time you select **Rotate**, the part rotates an additional 90°.

Convert

This command only displays for parts that have a convert.

Some library parts have a second form, usually (but not always) a DeMorgan equivalent, as well as the standard representation. If an object has a converted form, the **Convert** command displays on the command line when it is retrieved from the library.

Select **Convert** to convert the object from its normal form to its converted form. You may see the converted object by leaving the outline symbol stationary.

To return a converted object to its original form, select **Normal**.

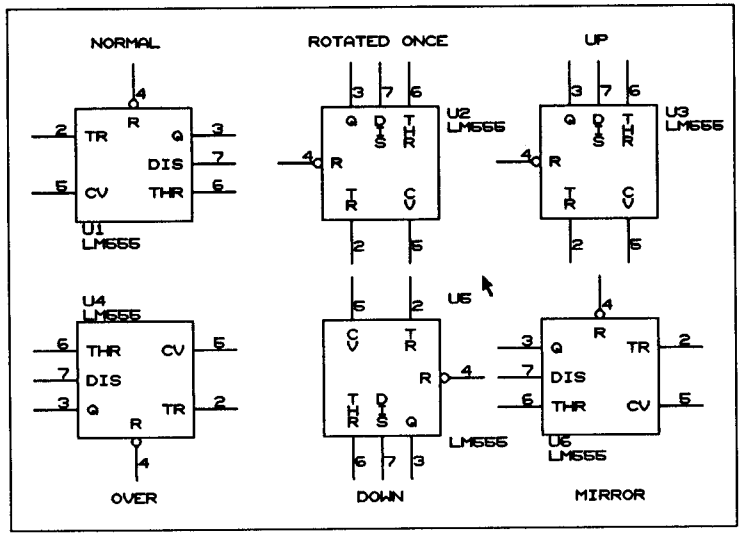


Figure 2-6. Normal, Rotated, Up, Over, Down, and Mirrored positions.

Normal Select **Normal** to rotate an object to its original position, as shown in the library (figure 2-6). This command also returns mirrored or converted parts to their original form.

Up Select **Up** to turn an object 90° counterclockwise (figure 2-6). This is equivalent to rotating it once from its normal position.

Over Select **Over** to turn an object 180° counterclockwise (figure 2-6). This is equivalent to rotating it twice from its normal position.

Down Select **Down** to turn an object 270° counterclockwise (figure 2-6). This is equivalent to rotating it three times from its normal position.

Mirror Select **Mirror** to get a mirror image of an object (figure 2-6).

HARDCOPY

HARDCOPY uses fast draft printing mode to send a worksheet to the printer or to a file.

HARDCOPY is not used to output to plotters. To plot a schematic, use the **Plot Schematic** as described in *Chapter 28: Plot Schematic.*

To print a schematic, select **HARDCOPY** from the main menu.

Draft displays the menu shown at right.

Hardcopy

Print Worksheet	
Block Print	
Destination	File
File Mode	Replaced
Width of Paper	Narrow
Scale	1.0000

△ *NOTE: For printing a multiple sheet design, use the **Print Schematic** reporter (described in* Chapter 29: Print Schematic*) or **Plot Schematic** reporter (described in* Chapter 28: Plot Schematic*).*

HARDCOPY Print Worksheet

Print Worksheet prints a hardcopy of the worksheet displayed on the screen. It either sends it to the printer attached to your printer port, or to a file, depending on the setting of **HARDCOPY Destination.**

To send a copy of your worksheet to your printer, first make sure your printer has power and is on line, and be sure that **HARDCOPY Destination** is set to your printer port. Then select **Print Worksheet. Draft** displays:

```
:::Creating Hardcopy of Sheet:::
```

After a few seconds, the worksheet starts printing.

To send a copy of your worksheet to a file, be sure that **HARDCOPY Destination** is set to **File.** Then select **Print Worksheet. Draft** creates a file with the name specified under **HARDCOPY Destination File.**

HARDCOPY Block Print

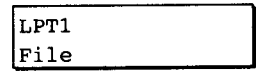
Use **Block Print** when you wish to print a specific area of your worksheet. When you select this option, you define the block to print with **Begin** and **End**. **Draft** displays

```
:::Creating Hardcopy of Sheet:::
```

After a few seconds, the worksheet starts printing.

HARDCOPY Destination

You can send a worksheet either to a printer or to a file. This command selects the hardcopy destination.



Select **Destination**. **Draft** displays a box like the one above.

LPT1

The upper entry is the printer port specified on the **Configure Schematic Design Tools** screen as the **Printer Port.** The configurable choices that might appear are LPT1, LPT2, LPT3, COM1, COM2, COM3, or COM4.

△ *NOTE: Some versions of DOS and BIOS do not support ports COM3 and COM4. If your printer is connected to one of these serial ports and HARDCOPY does not print, try changing to a different serial port or to a parallel port.*

File

Select **File** to specify a filename as the destination for the **HARDCOPY Print Worksheet** command. **Draft** displays the prompt "Destination of Hardcopy?" followed by the default filename HARDCOPY.PRN.

Change the default filename and press <Enter>.

You can specify a complete pathname including a drive specification. Because the output file is a graphics file, it requires more disk space than the source schematic file.

To actually create the file, select **HARDCOPY Print Worksheet**. Files created this way can later be sent to the printer using the DOS COPY command.

For example, if you create a printer file named MY.PRN and your printer is attached to LPT1, you can print with the following DOS command:

```
C:\> copy my.prn lpt1 /b
```

The /B parameter is needed because MY.PRN is a binary file. /B prevents COPY from terminating prematurely when it encounters the first Ctrl-Z character in the file.

△ *NOTE: Because the hardcopy file is a binary file, the DOS PRINT command cannot be used. For more information on COPY, see your DOS manual.*

HARDCOPY File Mode

File Mode determines whether output from **HARDCOPY** commands replace or are appended to the contents of any existing hardcopy file.

```
File Mode
Appended
Replaced
```

Select **File Mode. Draft** displays the menu shown above.

Appended Adds new data to the contents of the destination file. With this command, you can save a series of hardcopies to the same filename.

Replaced Replaces the contents of the destination file with new data. This command overwrites the current contents of the destination file with the new hardcopy.

HARDCOPY Width of Paper

Use **Width of Paper** to choose between narrow and wide paper.

Select **Width of Paper. Draft** displays the menu shown at right.

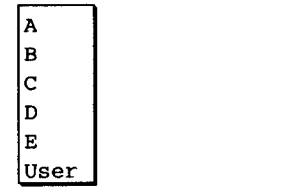
```
Width of Paper
Narrow
Wide
```

Narrow Select **Narrow** if you have narrow paper (8 inches wide).

Wide Select **Wide** if you have wide paper (13 inches wide).

HARDCOPY Scale

Use **Scale** to fit your hardcopy to a specific sheet size. You may choose one of the five standard sheet sizes or specify a scale. If you want a draft print at half scale, enter **0.5**.



A through E

Select **A, B, C, D,** or **E** to scale the worksheet to the corresponding paper size.

User

Select **User** to set the scale to a specific ratio. Draft displays "Enter User Scale" and provides the current scale as a default answer. Enter the desired scale.

For example, to print a draft copy at half size, enter `0.5`.

INQUIRE

INQUIRE displays text associated with stimulus, trace, vector, layout, library parts, and error objects. **Draft** associates text strings, some potentially lengthy, with these objects. **INQUIRE** keeps the schematic neat and uncluttered by making it possible to see text information belonging to an object when needed.

Move the pointer to an object and select **INQUIRE.** The text associated with the object displays on the top line of the screen. Press <Enter> or <Esc> to return to the main menu level.

If the text is too large to fit across the top of your screen (more than 80 characters), scroll the text left and right using the <←> and <→> keys or the mouse.

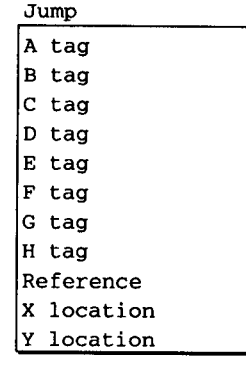
If you repeat **INQUIRE** on a position in the schematic and there is more than one object located there, **Draft** cycles through the texts associated with each object.

When you select **INQUIRE** on a part, **Draft** displays the original part name and the name of the library the part is loaded from.

JUMP

JUMP quickly moves the pointer to specific locations on the worksheet. The specific locations can be tags, grid references, or X,Y coordinates.

Select **JUMP**. **Draft** displays the menu shown at right.



JUMP A, B, C, D, E, F, G, H Tag

When you select one of the **JUMP Tag** commands, the pointer jumps to the specified tag on the worksheet (the tag must have been previously set with the **Tag** command). For information about the **Tag** command, see the *Tag* section in this chapter.

JUMP Reference

The **Reference** command moves the pointer to a specified grid reference on the worksheet border. Grid references are invisible until you set them using the **SET Grid Parameters** command. For information on grid parameters, see the *SET Grid Parameters* section of this chapter.

To jump to a grid reference, follow these steps:

1 Select **JUMP Reference**.

2. **Draft** displays "Jump to Reference." Select the desired Y-axis grid alphabetic reference from the menu. See table 2-1 for the range of letters that can be used as grid references.

3. Select the desired X-axis numeric grid reference from the menu. See table 2-3 for the range of numbers that can be used as grid references.

The pointer jumps to the grid reference location specified and **Draft** returns to the main menu level.

Sheet Size	ANSI Reference		Common References	
	X Grid Range	Y Grid Range	Y Grid Range	X Grid Range
A	N/A	N/A	A..D	1..8
B	N/A	N/A	A..D	1..8
C	A..D	1..4	A..D	1..8
D	A..D	1..8	A..D	1..8
E	A..H	1..8	A..D	1..8

Table 2-3. X and Y grid references. N/A indicates that the value is not applicable because the sheet size does not have grid references per ANSI Y14.1-1980.

JUMP X-Location

The **X-Location** command moves the pointer a specific distance along the X-axis. Each step represents $^{1}/_{10}$ (0.1) inch on the worksheet if **SET Grid Parameters Stay On Grid** is turned on. Otherwise it is $^{1}/_{100}$ (0.01) inch.

To jump to an X-location, follow these steps:

1. Select **JUMP X-Location.**
2. **Draft** displays "Jump X." Enter the number of steps to jump. A number without a plus or minus sign moves the pointer to the actual grid coordinate. A number with a positive sign (+10, +25, +30, and so on) moves the pointer to the right, and a number with a negative sign (–10, –25, –30, and so on) to the left. If you enter +10, for example, the pointer jumps to the right 1 inch from its current position (if you have **SET Grid Parameters Stay On Grid** turned on). If you enter 10, without a plus or minus sign, the pointer moves to the actual X grid coordinate 10.0.

When you press <Enter>, the pointer jumps to the specified location and **Draft** returns to the main menu level.

JUMP Y-Location

The **Y-Location** command moves the pointer a specific distance along the Y-axis. Each step represents $^{1}/_{10}$ (0.1) inch on the worksheet if the **SET Grid Parameters Stay On Grid** command is turned on; otherwise it is $^{1}/_{100}$ (0.01) inch.

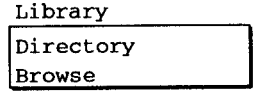
To jump to a Y-location, follow these steps:

1. Select **Y-Location.**
2. **Draft** displays "Jump Y." Enter the number of steps to jump. A number without a plus or minus sign moves the pointer to the actual grid coordinate, a number with a positive sign (+10, +25, +30, and so on) moves the pointer up, and a number with a negative sign (–10, –25, –30, and so on) down. If you enter +10, for example, the pointer jumps up 1 inch from its current position (if you have **SET Grid Parameters Stay On Grid** turned on). If you enter 10, without a plus or minus sign, the pointer moves to the actual Y grid coordinate 10.0.

When you press <Enter>, the pointer jumps to the specified location and **Draft** returns to the main menu level.

LIBRARY

LIBRARY displays part list directories of libraries and displays images of the parts in libraries configured to load with **Draft**.

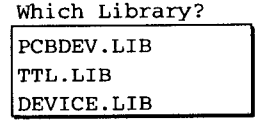


Select **LIBRARY**. **Draft** displays the menu shown above.

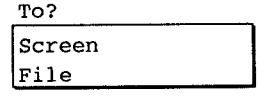
LIBRARY Directory

Use the **LIBRARY Directory** command to select a library and list its contents. The list can be displayed or saved in a file.

Select **LIBRARY Directory**. **Draft** displays a menu similar to the one at right, listing the libraries currently configured in **Draft**. From this menu, select the library for which to view a directory.



Draft displays the menu shown at right.

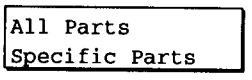


Screen Select **Screen**. **Draft** displays the library directory on the screen. When the screen fills, **Draft** pauses and displays the **More** and **Quit** commands. Select **More** to display another screen of part names. Select **Quit** to return to the main menu level.

File Select **File**. **Draft** displays "File?" on the prompt line. Enter the path and filename. Draft sends the library directory to the file.

LIBRARY Browse

Use the **Browse** command to view the contents of a library, or select a part and view it on the screen.



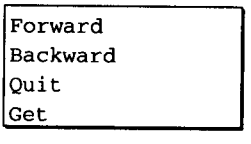
Select **Browse**. The menu shown above displays.

△ *NOTE: Some devices may be too large to fit entirely on the screen. Use the GET command to view these devices.*

All Parts

Select **All Parts** to view all the parts in a library. **Draft** displays a menu showing a list of the libraries currently configured to load with **Draft**. Select the library you want to view. **Draft** displays the menu shown below.

Select **Forward** or **Backward** to browse through the library. Select **Quit** to return to the main menu level. Select **Get** if you wish to place the current part.



Specific Parts

Select **Specific Parts** to view individual parts from the libraries. **Draft** displays "Part?"

Enter the name of the part you want to view. The part displays on the screen. If you do not know the name of the part, press <Enter> to display a list of parts. See *Chapter 20: Edit Library* for details.

MACRO

Macros are recordings of commands that you create and play back to run commands quickly, reducing the number of keystrokes required to perform repetitive and complex tasks. Use the **MACRO** command to capture, delete, initialize (erase), list, write to, and read macros from a file.

Each macro consists of a name and a script and may contain commands, pauses, other macros, and text. Each macro file contains one or more macros. Macros are stored on your hard disk in a text file.

You can assign macros to function keys, selected keyboard keys, keys used with <Ctrl>, <Shift>, and <Alt>, or the middle mouse button on a three-button mouse.

Schematic Design Tools 386+ includes two macro files, MACRO1.MAC and MACRO2.MAC on the product disks. These files include macros for drawing, editing, file management, and setting **Draft**'s environment.

You can create macros two different ways:

- Record typed keystrokes as you select commands in **Draft**.
- Write a macro script in a text editor.

Select the **MACRO** command. The menu at right displays.

```
Macro
Capture
Delete
Initialize
List
Read
Write
```

MACRO Capture

Select **Capture** to create a macro. **Draft** displays "Capture macro?"

Press the key or keys to use to call the macro. The keys you press appears on the prompt line. See table 2-2 for a list of valid keys that can be assigned as macros.

Press <Enter>. **Draft** displays "<macro>," showing **Draft** is in macro capture mode.

In macro capture mode, **Draft** records any sequence of keystrokes, mouse button clicks, and mouse movements. Commands you normally perform in **Draft** can be recorded and executed later as macros.

When you finish recording the macro keystrokes, choose one of the following:

- Press <M> or select **MACRO** to stop the **Capture** command at the main menu.
- Press <Ctrl><End> to stop the **Capture** command anywhere in the application.

Valid macro keys

A macro can be called by pressing selected keyboard keys; keys used with <Ctrl>, <Shift>, and <Alt>; or the middle mouse button on a three-button mouse. The macro name of the middle button is MMB.

Table 2-2 on the next page shows the keys to which macros can be assigned. The **Alt+**, **Ctrl+**, and **Shift+** headings indicate the key you press in combination with a key listed in the left-hand column. The **Key** heading indicates that you press the key alone.

	Alt +	Ctrl+	Shift+	Key
A	√	√		
B	√	√		
C	√			
D	√	√		
E	√	√		
F	√	√		
G	√	√		
H	√			
I	√	√		
J	√	√		
K	√	√		
L	√	√		
M	√			
N	√	√		
O	√	√		
P	√	√		
Q	√	√		
R	√	√		
S	√	√		
T	√	√		
U	√	√		
V	√	√		
W	√	√		
X	√	√		
Y	√	√		
Z	√	√		
Tab			√	
Ins				√
Left		√		

	Alt+	Ctrl+	Shift+	Key
F1	√	√	√	√
F2	√	√	√	√
F3	√	√	√	√
F4	√	√	√	√
F5	√	√	√	√
F6	√	√	√	√
F7	√	√	√	√
F8	√	√	√	√
F9	√	√	√	√
F10	√	√	√	√
1	√			
2	√			
3	√			
4	√			
5	√			
6	√			
7	√			
8	√			
9	√			
0	√			
^		√		
=	√			
-	√			
\		√		
]		√		
Right		√		
Pgup		√		√
Pgdn		√		√

Table 2-4. Valid key combinations for macros.

Grouping macros by type of task and assigning an extending key to each type helps organize your macros. For example, use the <Alt><Function> keys to set the environment and the <Ctrl><Alpha> keys to draw schematics.

Within each group of macros, you may assign logical initials for functions. For example, a macro that places a junction may be named <Alt><J>.

Nesting macros

A macro can call another macro or call itself from within a macro. When you are capturing a macro, type the key name of a previously saved macro. For example: Suppose you want to nest the macro assigned to F2 within a new macro. Type F2 at the appropriate time while you are capturing the new macro.

You may create a macro that calls itself, however the macro is recursive, or looping, and does not end until you press <Ctrl><Break>. To nest a macro inside another macro, insert the macro name, enclosed by curly brackets, inside the text of another macro. For example:

```
{F3}=sry{F2}{}
```

Pause

If you want the macro to pause and wait for a command or text, press <Ctrl><Home> followed by <Enter> and continue entering the remaining commands. When a macro containing such a command is running, it pauses for your input and resumes when you press <Enter> or click the left mouse button.

Debugging macros

After capturing or writing a macro, test it for correctness. If you need to fix any problems, you may either capture it again or edit the macro using a text editor. Here are some hints for debugging:

- Print the file containing the macro and use it as a script for entering the commands manually.
- Use <Ctrl><Break> to stop a macro that doesn't stop on its own.
- If a macro does not run, another macro might still be running. When a macro is expecting <Enter> after <Macrobreak> and you use keyboard commands instead, the macro does not stop. If a macro runs the same commands endlessly, it is stuck in a loop. This sometimes happens when you nest macros and specify that a macro calls itself. It can also occur if you mix keyboard and mouse commands when you use macros assigned to the middle button.

Initial macros

Initial macros run automatically each time you run **Draft.**

You may use an initial macro to set the environment to suit your preferences or project. You specify the initial macro and the macro file containing it on the **Configure Schematic Design Tools** screen. You may specify only one initial macro at a time. For information about configuring **Draft** for macros, see *Macro Options* in *Chapter 1: Configure Schematic Tools.*

△ *NOTE: Initial macros* must *begin with* <*Enter*><*Esc*>.

MACRO Delete

To delete a macro, select **Delete. Draft** displays "Delete macro?" Type the key combination assigned to the macro you wish to delete, then press <Enter>.

MACRO Initialize

This command erases all macros in the macro buffer. To erase all of the macros, select the **Initialize** command. **Draft** displays the prompt "Initalize – Erase All Macros?"

Select **No** to return to the main menu level, or **Yes** to erase all macros.

MACRO List

Select **List** to list all the key combinations assigned to macros.

MACRO Read

Read loads a macro file into an area of memory called the *macro buffer,* where the macros can be accessed by **Draft.** If the macro buffer already contains macros, the macros with unique key combination assignments are added to those in the buffer. Macros in the file with key combination assignments that are identical to key combinations assigned to macros in the buffer overwrite the macros in the buffer.

To load a macro file, select **Read. Draft** displays "Read all macros from?"

Enter the path and filename in which the macros are stored. **Draft** loads the macro file.

MACRO Write

When you capture macros **Draft** keeps them in memory, in the macro buffer. **MACRO Write** saves all macros currently in the macro buffer to a file. To save the macros to a file, select **Write. Draft** displays "Write all macros to?"

Enter a filename. If the file you name already exists, it is overwritten.

Macro files can be loaded automatically when **Draft** runs by entering the macro filename and the name of the initial macro in the **Macro Options** area of the **Configure Schematic Design Tools** screen (see *Chapter 1: Configure Schematic Tools*).

Using macros

Before **Draft** can use macros in a macro file, it must read the file into the macro buffer. The macro—when called—runs from the buffer.

To use a macro, run **Draft** and read the macro file into the macro buffer with the **MACRO Read** command.

Calling a macro

To run a macro, press the key combination for that macro. All instructions in the macro script run.

Macro buffer

The macro buffer defaults to 8192 bytes of memory. If the buffer fills, **Draft** displays a warning message. To increase buffer memory you can either change the buffer size or delete unused macros. See **Macro Options** in *Chapter 1: Configure Schematic Tools* for more information about macro buffer memory.

Macro text files

A macro is a text file, and can be edited or created using a text editor or word processing program such as **Edit File.** The application you use must be able to save the macro file in text-only format.

Macro syntax

The syntax of a macro definition is shown below.

```
{Macro Name} = Macro Script { }
```

- *Macro name* is a valid key, key combination, or the acronym MMB enclosed in curly brackets.
- The equal sign (=) indicates that commands follow.
- *Macro script* is the list of commands the macro runs.
- The empty curly brackets ({ }) mark the macro's end.

The table below lists translations for command keys and macro names.

Key	*Translation*
Alt	\
Ctrl	^
Shift	SHIFT
Ctrl-Home	{MACROBREAK}
Enter	{ENTER}
Up	{U}
Down	{D}
Left ←	{L}
Right →	{R}

Name	*Translation*
Backspace	{RUBOUT}
Escape	{ESC}
Home	{HOME}
End	{END}
Shift-Tab	{BACKTAB}
Tab	{^I}
Insert	{INS}
Delete	{DEL}

Table 2-5. Key translations.

When you create macros using a text editor, you may organize the macros any way you choose, and include any character with these restrictions:

- Use only characters representing **Draft** commands, text, or filenames in macro scripts.
- Avoid line breaks or paragraph breaks in long macros. You may use groups of spaces to force the macro to wrap on the screen.
- Use only MMB or valid key combinations from table 2-2 for macro names.

- Do not use curly braces or equal signs anywhere except where required by macro syntax.

When you finish writing the macro, name the file and save it in text-only format.

Example: macro for cutting wires

To cut a wire in **Draft**, use this macro instead of deleting and redrawing the entire wire. This macro places a junction on the wire, drags the junction to break the wire, and deletes the junction. To cut off the extra wire, put the cursor on the extra segment and select **DELETE Object Delete**.

```
{\B}=pjp{ESC}bdbe{U}{D}pdodj{ESC}{}
```

Example: macro for placing parts on grid

After using the **Cleanup Schematic** processor to identify parts that are off grid, you can use this macro to find and move parts back onto the grid. At the *Find?* prompt, enter the name or part value of the part that is off grid. The macro copies, deletes, and gets the part. The part appears highlighted, ready for you to position it. To place the part, press <Enter>. If a part is already on grid, this macro does not move it off grid.

```
{\A}=f{macrobreak}bsbedod{esc}sgsybg{macrobreak}
p {esc}{}
```

Macro comments

Using a text editor, you may add comments to macros that remind you and others what each macro is for. Macros with comments are also useful for quick reference. Follow these guidelines to add and preserve comments in macro files.

- Place comments before or after the macro, or in both positions. You may also insert paragraphs between macros. The comments in these macros are shown in bold:

```
PLACE Junction {^P}=pjp{}
{^P}=pjp{} PLACE Junction
PLACE Junction {^P}=pjp{} for SuperSquig project
```

- Do not use the left brace ({) in comments.
- Use a text editor to alphabetize the list by comment or by macro name. Save the file in text-only format.
- Make a back-up copy of the file and store it in a safe location such as another directory or on a diskette.
- Consider changing the extension of original macro files. For example:

```
C:> COPY SAMPLE.MAC SAMPLE.MAS
```

- Avoid writing over original macro files with **MACRO Write**. When you select **MACRO Read, Draft** ignores all text outside macros. When you select **MACRO Write, Draft** erases all comments placed in the file.
- Print the macro file to create a quick reference card.

Once you have a collection of macros, keeping track of what each macro does may get complicated, particularly if you share macro files with other OrCAD users. You may want to organize your macro files by establishing conventions for naming the macros and by adding comments to your macro files.

Middle mouse button macros

This section describes a method for controlling multiple **Draft** macros using the middle mouse button.

For each macro you assign to the middle button, you need two macros. The first macro contains the commands for doing a task such as placing a junction. You save this macro in its own file. For example, the macro for placing a junction looks like this:

```
{mmb}=pjp{esc} {}
```

Save the macro to the following file:

```
putjunct.mac
```

The second macro directs the application to read the macro file containing the first macro. You may either include the second macro in the macro file that you specified during configuration, or you may use the **MACRO Read** command to read the macro file.

For example, the macro for assigning the junction macro to the middle mouse button looks like this:

```
{^j}=mrputjunct.mac{enter} {}
```

The file you save this macro to might be this:

```
mymacros.mac
```

If you develop a number of macros for your three-button mouse, you may find it helpful to create a subdirectory to contain macro files. If you create such a subdirectory, named ORCADESP\SDT\MACRO for this example, the second macro changes to this:

```
{^j}=mrc:\
ORCADESP\SDT\MACRO\putjunct.mac{enter}{}
```

Assignment macros

Once you set up the macro files, you assign macros to the middle button using keyboard commands. For example:

```
{\b}=mrmacro_b.mac{enter}{}
{\d}=mrmacro_d.mac{enter}{}
{\g}=mrmacro_g.mac{enter}{}
{\j}=mrmacro_j.mac{enter}{}
{\k}=mrmacro_k.mac{enter}{}
{\s}=mrmacro_s.mac{enter}{}
{\w}=mrmacro_w.mac{enter}{}
{\z}=mrmacro_z.mac{enter}{}
```

Individual macros

```
PLACE Bus
    {mmb}=pb{}
    Saved in macro_b.mac
DELETE Block Begin
    {mmb}=dbb{macrobreak}e{}
    Saved in macro_d.mac
BLOCK Drag Begin
    {mmb}=bdb{macrobreak}e{macrobreak}p{}
    Saved in macro_g.mac
PLACE Junction Place
    {mmb}=pjp{esc}{}
    Saved in macro_j.mac
BLOCK Move Begin
    {mmb}=bmb{macrobreak}e{macrobreak}p{}
    Saved in macro_k.mac
BLOCK Save Begin
    {mmb}=bsb{macrobreak}ebg{}
    Saved in macro_s.mac
PLACE Wire Begin
    {mmb}=pwb{}
    Saved in macro_w.mac
ZOOM Out and Back
    {mmb}=zo{macrobreak}zi{}
    Saved in macro_z.mac
```

Creating efficient macros

When you capture a macro, you may press keys or click the mouse. When you write a macro, you may duplicate keystrokes and mouse clicks. Macro size and execution speed depend on the method you use to select commands. Pressing keys results in fewer macro instructions. In complex macros, using keystrokes:

- Speeds the macro up
- Increases its readability
- Minimizes its size and thereby maximizes the number of macros that the macro buffer may hold
- Frees more memory for the worksheet

The following examples of initial macros show the difference between a macro captured with keystrokes and the same macro captured with mouse clicks.

- Macro captured with keystrokes, 32 bytes:

```
{F1}={ENTER}sgvysdysxy{U}{D}{}
```

- Macro captured with mouse clicks and keystrokes, 364 bytes:

```
{F2}={ENTER}{D}{D}{D}{D}{D}{D}{D}{D}{D}{D}{D}{D}
{D}{D}{ENTER}{D}{D}{D}{D}{D}{D}{D}{D}{D}{D}{R}{D}
{ENTER}{D}{D}{D}{ENTER}{ENTER}{ENTER}{D}{D}{D}{D}
{D}{D}{D}{D}{D}{D}{D}{D}{D}{D}{ENTER}{U}{U}{R}{U}
{U}{U}{D}{D}{ENTER}{ENTER}{ENTER}{D}{D}{D}{D}{D}
{D}{D}{D}{D}{D}{R}{D}{D}{D}{R}{D}{ENTER}{D}{D}{D}
{D}{D}{D}{D}{D}{D}{D}{L}{D}{U}{U}{L}{D}{ENTER}
{ENTER}{D}{U}{}
```

You may also increase the speed at which macros run by selecting **SET Macro Prompts No** either before running the macro or as part of the macro. Figure 2-7 compares the execution times of F1 and F2 with **Macro Prompts** set **On** and **Off**.

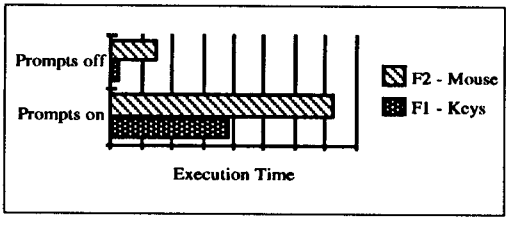


Figure 2-7. Macro execution times for key and mouse macros.

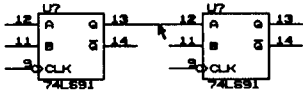
PLACE

PLACE puts wires, buses, junctions, bus entries, labels, text, module ports, power, dashed lines, and hierarchical sheets on your worksheet.

Select **PLACE** from the main command menu. **Draft** displays the menu shown at right.

```
Place
Wire
Bus
Junction
Entry (Bus)
Label
Module Port
Power
Sheet
Text
Dashed Line
Trace Name
Vector
Stimulus
NoConnect
Layout
```

PLACE Wire



Select **PLACE Wire** to place wires in the worksheet. **Draft** displays:

```
Begin  Find  Jump  Zoom
```

To draw a wire, first place the pointer at the point on the worksheet where you want the wire to start.

Select **Begin**. **Draft** displays:

```
Begin  End  New  Find  Jump  Zoom
```

Draw the wire by moving the pointer. Use the following **PLACE Wire** commands to finish drawing the wire.

Begin

Use the **Begin** command to:

- Start drawing a wire segment.
- Finish drawing a wire segment and begin a new one (if the wire you are drawing makes a 90° turn).

You can use **Begin** over and over again to draw a complex wire. As you move the pointer, a dashed guide line representing the wire is drawn.

To continue drawing the wire from a 90° turn, select **Begin** where the turn starts (point A in figure 2-8). You may also move to the end of the wire (point B in figure 2-8) and select either **Begin, End,** or **New** to place a wire segment. When you finish placing a wire segment with **Begin, End,** or **New,** the dashed guide line becomes a solid line. A dashed guide line shows placement of a wire segment has not been completed with the **Begin, End,** or **New** commands.

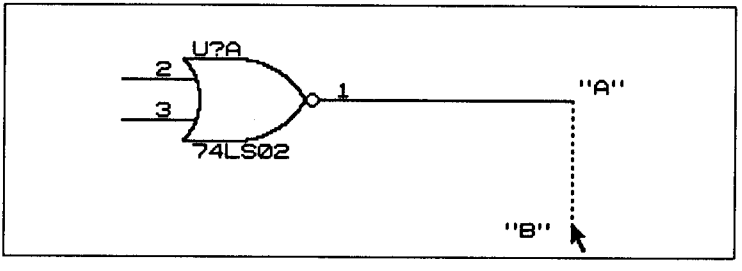


Figure 2-8. Drawing a wire.

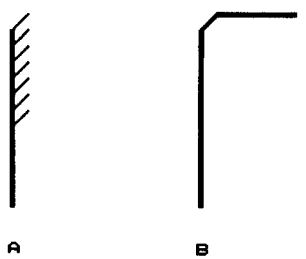
Continue drawing the wire. To end the wire, select either **End** or **New.**

△ *NOTE: See the* Schematic Design Tools 386+ User's Guide *for examples of macros that simplify wire placement.*

End Select **End** when you are done drawing a wire. **Draft** returns to the main menu level.

New Select **New** when you are done drawing a wire and would like to start drawing another wire. **Draft** remains in wire-placing mode, and returns to the "Begin Find Jump Zoom" command line.

PLACE Bus Select **Bus** to place buses on the worksheet. **Draft** displays:




```
Begin  Find  Jump  Zoom
```

To draw a bus, place the pointer at the worksheet location where you want the bus to start. Select **Begin**. **Draft** displays:

```
Begin  End  New  Find  Jump  Zoom
```

Draw the bus by moving the pointer. Select one of the **PLACE Bus** commands to finish drawing the bus.

Begin Use the **Begin** command to:

- Start drawing a bus segment.
- Finish drawing a bus segment and begin a new one (if the bus you are drawing needs to make a 90° turn.

You can use **Begin** repeatedly to draw a complex bus.

To continue drawing the bus from the 90° turn, select **Begin** where the turn starts. You may also move to the end of the bus and select either **Begin**, **End**, or **New** to fill it in.

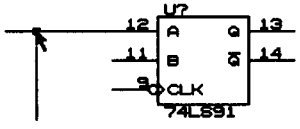
Continue drawing the bus until you come to where you want it to end. To connect the bus to an end point, select either **End** or **New**.

End With the pointer at the end point, select **End**. **Draft** returns to the main menu level.

New With the pointer at the end point, select **New**. **Draft** stays in bus-placing mode and returns to the "Begin Find Jump Zoom" command line. Drawing a bus is identical to drawing a wire (see the **PLACE Wire** command).

△ *NOTE: Buses with module ports attached to them are automatically labeled with the same name as the module port, and do not need an additional label.*

PLACE Junction



On a worksheet, many wires and buses connect or cross each other. Junctions are placed on the worksheet to distinguish a connection from a cross-over. If more than two wires or buses connect to a common node, place a junction there to tell the **Check Electrical Rules** reporter and **Create Netlist** processor that the node is a physical connection.

If you don't place a junction at an intersection of wires or buses, **Check Electrical Rules** and **Create Netlist** interpret the intersection as a cross-over.

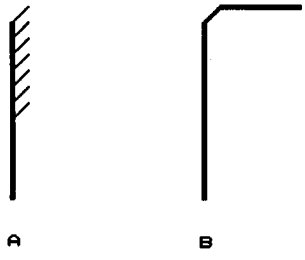
In many designs, you may want to connect a wire at 90° angles to a bus. If you do, you must place a junction at the connect point. Junctions are not required if you use a bus entry (see the **PLACE Entry (Bus)** command).

To place a junction in the worksheet, select **PLACE Junction**. **Draft** displays:

```
Place  Find  Jump  Zoom
```

Position the pointer where you want the junction and select **Place**. **Draft** remains in **"PLACE Junction"** mode until you press the <Esc> key.

PLACE Entry (Bus)



Use the **Entry (Bus)** command to place bus entries on a worksheet. Use bus entries are for aesthetic purposes to connect wires or other buses to a bus. In the picture at the left, A shows a series of wire bus entries and B shows a bus entry used for a bus turn.

When you select **PLACE Entry (Bus), Draft** shows the last bus entry selected, with this command line:

```
Place  /  \  Wire  Bus  Find  Jump  Zoom
```

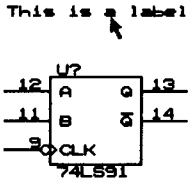
To place a bus entry, select **Place**. Select the / or \ command to change the bus entry angle. **Draft** shows the last bus entry angle when you select **PLACE Entry (Bus)**.

Select **Wire** to place wire entries. Use this command when a wire is to exit or enter a bus from another object.

Select **Bus** to place bus entries. Use this command when a bus makes a turn or is joined to another bus.

△ *NOTE: Junctions are not required to connect a bus entry to a bus. (See **PLACE Junction.**)*

PLACE Label



A label is an identifier placed on a worksheet that connects signals (wires and buses) together without actually drawing the wires connecting them. A label is automatically assigned to a bus when you attach it to a module port, so you do not need to place an additional label. You can place labels horizontally or vertically on a worksheet.

Labels are *not* comments. Labels have meaning for other tools, such as **Create Netlist**. To place a comment on the worksheet, use the **PLACE Text** command.

To place a label, select **Label**. **Draft** displays the prompt: "Label?" Enter the name of the label. **Draft** displays:

```
Place  Orientation  Value  Larger  Smaller  Find
```

Place

Select **Place** to place the label on the worksheet.

When the label is placed, the "Label?" prompt returns. You can place another label or press <Esc> to return to the main menu level.

△ *NOTES: If you always stay on grid, the label is always correctly positioned to connect to a wire or bus. Just move the pointer on the wire or bus to place the label.*

If you enable ***Do not append sheet number to labels*** *in the local configuration of IFORM or HFORM, you need to use labels for local nets on one sheet that are different from local netnames on other sheets. This way, your netlist will not mix local nets on one sheet with local nets of another sheet.*

Orientation

Select **Orientation** to change the orientation of the label to **Horizontal, Vertical, Up, Over,** or **Down**.

Value Select **Value** to enter a value for the label. When you select **Value**, the prompt "Text?" displays followed by the current value. Edit the value by moving the cursor with the <←> and <→> keys, and the <Home> and <End> keys, erasing characters with <Backspace> and <Delete>, and adding new alphanumeric or extended ASCII characters. To type an extended ASCII character, hold down the <Alt> key and type the number of the character using the number keypad. When you finish editing the label's value, press <Enter>.

Larger Select **Larger** to make a label larger. Select **Larger** as many times as you like until the label is large enough.

Smaller Select **Smaller** to make the label smaller. As with **Larger**, you may select **Smaller** over and over.

Placing labels correctly In order for **Check Electrical Rules** and **Create Netlist** to associate internal and bus member labels with wires and buses, you must place labels in "contact" with the bus or wire. The bottom edge of the leftmost character is the "hotpoint" of a label. Some portion of it must be right next to the bus or wire to establish contact.

Figure 2-9 shows correct label positions for both vertical and horizontal wires. Notice the hotpoints for each label (bottom edge of the character "L") are next to the wire.

Figure 2-10 shows incorrect label positions. None of the label hotpoints are next to the wire.

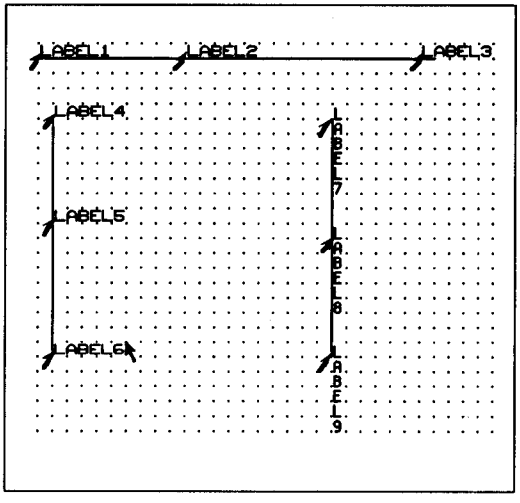


Figure 2-9. Correct label positions.

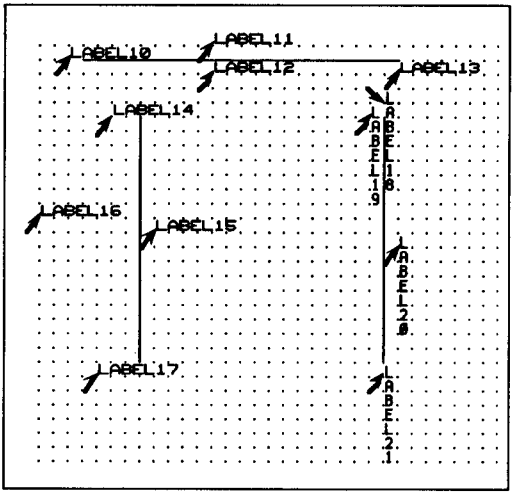
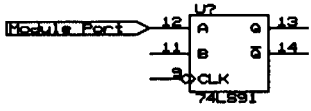


Figure 2-10. Incorrect label positions.

In both figures, the right-pointing arrows indicate label hotpoints.

PLACE Module Port



A module port is used to connect signals on the current sheet to signals on other worksheets. Unspecified module ports may be used to transfer isolated power from one sheet to another. Module ports may be connected to either wires or buses.

Signals remaining internal to the worksheet should use labels, not module ports.

All module ports and labels with the same name are considered to be electrically connected, just as are all labels with the same name. For example, if you have a module port named "ABC" and a wire labeled "ABC" that connects two components somewhere else on the worksheet, the module port and the label are electronically connected even though they aren't physically connected on the worksheet.

To place a module port, select **Module Port** command. The prompt "Module Port Name?" displays. Enter the module port name. **Draft** displays the menu shown at right.

```
Input
Output
Bidirectional
Unspecified
```

Select **Input** if the module port is used as a signal input, **Output** if the module port is used as a signal output, **Bidirectional** if the module port is used as a bidirectional signal, or **Unspecified** if the module port is used to transfer power or "don't care" signals. Figure 2-11 shows the four types of module ports and their default styles.

After selecting one of the commands shown at right, **Draft** draws a preliminary version of the module port with its name. You may move it before placing it. The following command line displays:

```
Input          I>
Output         <I
Bidirectional  <>
Unspecified    II
```

```
Place  Value  Type  Style  Find  Jump  Zoom
```

Select **Place** to place the module port on the worksheet. The prompt "Module Port Name ?" displays, so that you may place another module port. Press <Esc> to return to the main menu level.

Before placing the module port, you can also change its **Value, Type,** or **Style.** A module port's value is whatever you typed after "Module Port Name?" Its type is **Input, Output, Bidirectional,** or **Unspecified.** Its style is how it looks on the screen. If you select **Style,** you can choose from the options shown on the next page.

A module port's style is independent from its type. For example, a **Both pointing** style does not necessarily mean the module port is **Bidirectional.**

```
Module Port Style
Right pointing
Left pointing
Both pointing
Neither pointing
```

△ *NOTE: Normally, an unspecified module port is specified when isolated power is transferred between worksheets. For more information, see* Chapter 11: Create Netlist.

The figure below shows the four types of module ports with their default styles.

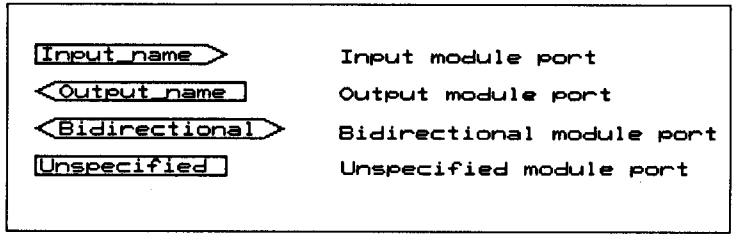
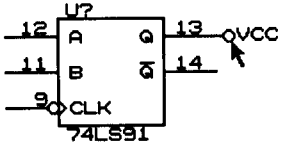


Figure 2-11. Types of module ports.

△ *NOTE: Module ports are not intended to be used as physical connectors, such as DB-9, and so on. Physical connectors are objects that should be created as library parts. For information on working with connectors, see* Chapter 11: Create Netlist.

PLACE Power



Use **PLACE Power** to place power supply objects on the worksheet.

To place a power object on a worksheet, select **Power. Draft** displays:

```
Place  Orientation  Value  Type  Find
```

The power object appears on the screen, ready to be positioned and placed on the worksheet. Before placing the power object on the worksheet, you can change its **Orientation, Value,** or **Type.**

If you select **Orientation,** you can choose **Top, Bottom, Left,** or **Right.**

△ *NOTE: The power pin default is a circle with a value of VCC. When you execute* ***PLACE Power****, the orientation returns to* ***Top****. However, type and value will be whatever was set previously.*

A power object's value is the text (for example, VCC) associated with it. If you select Value, **Draft** displays the prompt "Power Value? xxx" where xxx represents the current value. You can backspace over the current value or append to it. Enter the new value (for example, +5, GND, + 5 VDC, -12 VDC, VSS, VEE, or any other text string). **Draft** returns to the main menu level.

A power object's **Type** is its appearance on the screen. If you select **Type,** you can choose **Circle, Arrow, Bar,** or **Wave.**

Select **Place** to place the power object where you want it on the worksheet. Then, press <Esc> to return to the main menu level.

If you use the **Create Netlist** processor, see *Chapter 11: Create Netlist* for information on handling isolated power such as in battery backup and other applications.

Figure 2-12 shows the four kinds of power objects and their orientations.

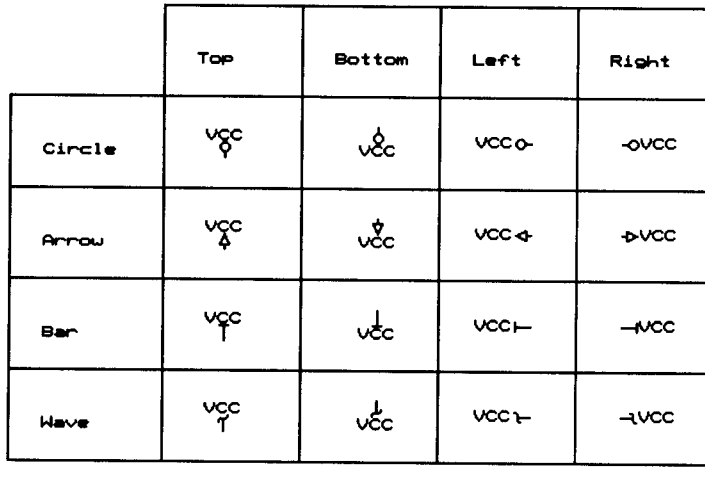
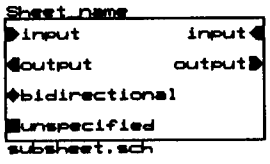


Figure 2-12. Circle, Arrow, Bar, and Wave power objects and their orientations.

PLACE Sheet



In **Draft** you can create hierarchical designs using *sheet symbols*. A sheet symbol, which represents a worksheet in a hierarchy, contains *net names*. The net names connect signals on the active worksheet to the sheet the symbol represents.

A *sheet net* is the way a connection is made between the signal attached to the sheet net on the current sheet and the module ports on the worksheet represented by the sheet symbol.

To place a sheet symbol, select **Sheet. Draft** displays:

```
Begin  Find  Jump  Zoom
```

Select **Begin**, outline the area, then select **End** to finish it. **Draft** then displays:

```
Add-Net  Delete  Edit  Name  Filename  Size
```

Pointer movement is now restricted to the left and right sides of the box of the hierarchical sheet, the proper locations for sheet names and nets.

Add-Net

Use **Add-Net** to add sheet nets so that connections can be made between worksheets. To add a net sheet, place the pointer at the location where you want the name and select **Add-Net. Draft** displays the prompt "Net Name?" Enter the net name. **Draft** displays the menu shown below. Select the appropriate type for your sheet net.

For more information, see *Editing sheets* earlier in this chapter, and the discussion of hierarchies in the *Schematic Design Tools 386+ User's Guide*.

```
Input          I>
Output         <I
Bidirectional  <>
Unspecified    II
```

Delete

To delete sheet nets, place the pointer at the net's location and select **Delete**.

Edit To edit a sheet net, place the pointer at the net's location and select **Edit**. **Draft** then displays a menu allowing you to choose between **Name** and **Type**.

To edit the net name, select **Name**. Edit the net name by positioning the cursor with the <←> and <→> keys or the <Home> and <End> keys, erasing characters with <Backspace> and <Delete>, and adding new characters with the alphanumeric keys. When you finish editing the net name, press <Enter>.

To edit the net type, select **Type**. You can then choose **Input**, **Output**, **Bidirectional**, or **Unspecified**.

Name Use the **Name** command to edit the sheet name. The initial sheet name is a question mark (?) located at the top of the sheet. Typical sheet names are "Memory Array" or "Dynamic RAM Refresh circuitry."

To specify a sheet name, select **Name**. **Draft** displays "Sheet Name?" followed by the current sheet name. Edit the sheet name using the editing techniques discussed above. When you finish editing the sheet name, press <Enter> to place it at the top of the sheet.

Filename Use **Filename** to change the filename of the file representing the hierarchical worksheet. **Draft** automatically produces a filename based on the date and time of day the sheet symbol was placed in the schematic. This ensures no two filenames will be the same.

Edit this filename using the editing techniques discussed above. When you finish, press <Enter>.

Move Use **Move** to move a net connection. Move the pointer to the net connection you want to move, select **Move**, move to the new location, and select **Place** or press <Enter>.

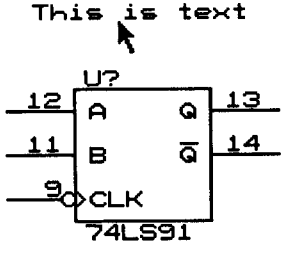
Size

Size increases or decreases the sheet size. When you select **Size, Draft** displays:

```
End  Jump  Zoom
```

Draft automatically positions the pointer on the lower right corner of the sheet. To change sheet size, move the pointer until you reach the desired size. Then select **End.**

PLACE Text



Use **PLACE Text** to place comments on your worksheet. Comments are useful for including revision history, tolerance, and other information in the worksheet.

When you select **Text, Draft** displays the prompt "Text?" Enter the text you want as a comment. You may use alphanumeric or extended ASCII characters. When you finish, press <Enter>. **Draft** displays the command line:

```
Place  Orientation  Value  Larger  Smaller  Find  Jump
```

Place

Select **Place** to place the text on the worksheet.

When the text is placed, the "Text?" prompt returns. You can place more text or press <Esc> to return to the main menu level.

Orientation

Select **Orientation** to specify the direction of the baseline of the label's text.

Horizontal and **Vertical** orient the text to the bottom of the screen.

Up rotates the label 90° counterclockwise, orienting the text to the right edge of the schematic.

Over rotates the label 180° counterclockwise, orienting the text to the top edge of the schematic.

Down rotates the label 270° counterclockwise, orienting the text to the left edge of the schematic.

Value Select **Value** to edit the text. When you select **Value**, the prompt "Text?" displays followed by the current value. Edit it by positioning the cursor with the <←> and <→> keys, and the <Home> and <End> keys, erasing characters with <Backspace> and <Delete>, and adding new characters with the alphanumeric keys or with <Alt> and the number keypad. When you finish editing the value, press <Enter>.

Larger Select **Larger** to make the text larger. You may select **Larger** more than once until the text is as large as you like.

Smaller Select **Smaller** to make the text smaller. As with **Larger**, you may select **Smaller** over and over.

PLACE Dashed Line

Dashed Line places a dashed line on the worksheet. This is useful for setting off sections of your design. You can then label sections with comments constructed with **PLACE Text**.

Placing a dashed line is similar to placing a wire. When you select **Dashed Line, Draft** displays:

```
Begin  Find  Jump  Zoom
```

To draw a dashed line, place the pointer at the point on the worksheet where you want the dashed line to start.

Select **Begin. Draft** displays:

```
Begin  End  New  Find  Jump  Zoom
```

Draw the line by moving the pointer. Use the following **PLACE Dashed Line** commands to finish drawing the line.

Begin

Use the **Begin** command to:

- Start drawing a dashed line.
- Finish drawing a dashed line and begin a new one (if the line you are drawing makes a 90° turn).

You can use **Begin** over and over again to draw a complex line. To end a dashed line, select either **End** or **New**.

End

Select **End** when you are done drawing a dashed line. **Draft** returns to the main menu level.

New

Select **New** when you are done drawing a dashed line and would like to start drawing another dashed line. **Draft** remains in the line-drawing mode, and returns to the "Begin Find Jump Zoom" command line.

PLACE Trace Name



A trace is a special marker placed on a worksheet that identifies a node to be traced in the digital simulation of the design. To keep the design from being cluttered, the trace name is not visible on the schematic. To view the contents of a trace, use the **INQUIRE** command.

To place a trace, select **Trace Name**. **Draft** displays the prompt: "Trace Name?" Enter the name you wish this signal to be displayed as in the simulator. **Draft** displays:

```
Place  Value  Find  Jump  Zoom
```

Place

Select **Place** to place the trace on the worksheet. After placing the trace, the "Trace Name?" prompt returns. You can place another trace or press <Esc> to return to the main menu level.

△ *NOTE: If you always stay on grid, the trace is always correctly positioned to connect to a wire or bus. Just move the pointer on the wire to place the trace.*

Value

Select **Value** to edit value for the trace before you place the trace. When you select **Value**, the prompt "Trace Name?" displays followed by the current value. Edit the value using the editing techniques describe previously When you finish editing the trace name, press <Enter>.

Trace name format

A trace may be placed on either a wire or a bus. The trace name may consist of any printable characters including spaces, except a period. If the display name contains a left or right square bracket, then the name is considered to be a bus name and the trace must be placed on a bus. For buses, the format of the bus may be specified after the right square bracket in the name. There must be a space between the right square bracket and the format character. The format is a single character from the following list:

B	Binary format	D	Decimal format
H	Hexadecimal format	O	Octal format

Examples

```
ADD 5
DATA[0..15] D
```

PLACE Vector

A vector is a special marker placed on the worksheet that identifies a node to be stimulated by a test vector. The test vectors are referred to by column number, therefore, the vector placed on the worksheet has a vector column value. To keep the design from being cluttered, the vector column is not visible on the schematic. To view the vector column number of a particular vector, use the **INQUIRE** command.

To place a vector, select **Vector**. **Draft** displays the prompt: "Vector Column?" Enter the column number of the vector. **Draft** displays:

```
Place  Value  Find  Jump  Zoom
```

Place

Select **Place** to place the vector on the worksheet.

When the vector is placed, the "Vector Column?" prompt returns. You can place another vector or press <Esc> to return to the main menu level.

△ *NOTE: If you always stay on grid, the vector is always correctly positioned to connect to a wire or bus. Just move the pointer on the wire to place the vector.*

Value

Select **Value** to edit the vector column. When you select **Value**, the prompt "Vector Column?" displays followed by the current value. Edit the value by positioning the cursor with the <←> and <→>> keys, and the <Home> and <End> keys, erasing characters with <Backspace> and <Delete>, and adding new characters with the alphanumeric keys. When you finish editing the vector column, press <Enter>.

Vector column format

The vector column indicates which column of the test vector file to use when the simulation is run. The column number is a decimal whole number such as 5.

PLACE Stimulus

A stimulus is a special marker placed on a worksheet that identifies a node to be stimulated by in the digital simulation of the design. The stimulus is an expression describing the pattern of logic states. To keep the design from being cluttered, the stimulus pattern is not visible on the schematic. To view the contents of the stimulus, use the **INQUIRE** command.

To place a stimulus, select **Stimulus**. **Draft** displays the prompt: "Stimulus?" Enter the value of the stimulus. **Draft** displays:

```
Place  Value  Find  Jump  Zoom
```

Place

Select **Place** to place the stimulus on the worksheet.

When the stimulus is placed, the "stimulus?" prompt returns. You can place another stimulus or press <Esc> to return to the main menu level.

△ ***NOTE:*** *If you always stay on grid, the stimulus is always correctly positioned to connect to a wire. Just move the pointer on the wire to place the stimulus.*

Value

Select **Value** to edit the value of the stimulus. When you select **Value**, the prompt "Stimulus?" displays followed by the current value. Edit the value by positioning the cursor with the <←> and <→> keys, and the <Home> and <End> keys, erasing characters with <Backspace> and <Delete>, and adding new characters with the alphanumeric keys. When you finish editing the value, press <Enter>.

Stimulus value format

The stimulus describes the pattern of logic states to be applied to a net in the simulation. The pattern is described using a series of functions. A value must have a minimum of one function. The maximum is limited to the number of functions that can be placed in the text associated with the stimulus object.

There are two types of functions: set and branch. At least one space is required between each function.

Set function

A set function consists of two variables: the time and the value. These two variables are separated by a colon. The time is an unsigned whole number. The value is a single letter from the following list:

- 0 Set to 0
- 1 Set to 1
- U Set to undefined
- Z Set to high impedance
- T Toggle

For example, the function to set the signal to a 1 at time 50 is 50:1.

△ *NOTE: Separate multiple functions with spaces.*

Branch function

The branch function consists of two variables: the time the branch is to occur and the time to branch to. The two variables are separated by :G:. Both of the time values are unsigned whole numbers.

For example, the function to branch from time 450 back to time 400 is 450:G:400.

△ *NOTE: Only one branch function is allowed per stimulus value.*

Examples The following examples show proper stimulus expressions:

```
0:0  100:T  200:G:100
0:U  37:1 59:Z  83:G:37
0:U  50:0  100:Z  150:1  200:Z  250:U  255:0
0:1  20:0  40:Z  60:U  80:G:20
```

For additional information on stimulus, see the *Digital Simulation Tools 386+ Reference Guide.*

PLACE NoConnect



A no-connect is a special symbol that identifies a pin on a device that is intentionally to be left unconnected. This object causes reports that show unconnected pins to ignore pins with no-connect symbols. No-connect pins should not be connected together. No-connects may not be placed on buses, wires, sheet nets, module ports, labels, power objects, or bus entries. You can place no-connects anywhere, but the other tools will not recognize the no-connect.

To place a no-connect, select **NoConnect. Draft** displays:

```
Place  Find  Jump  Zoom
```

Place Select **Place** to place the no-connect on the worksheet.

You can place another no-connect or press <Esc> to return to the main menu level.

△ *NOTE: If you always stay on grid, the no-connect is always correctly positioned to connect to a pin. Just move the pointer on the pin to place the no-connect.*

PLACE Layout

A layout object is a special marker intended to give a layout directive to **PC Board Layout Tools**. This function is not currently implemented in **PC Board Layout Tools**, but it can be used to place messages on a schematic for use within **Schematic Design Tools 386+**.

To prevent design clutter, **Draft** displays the layout object symbol on the screen but not the message. To view the contents of the message, you select the **INQUIRE** command.

To place a layout object on the worksheet, select **Layout**. **Draft** displays the prompt: "Layout Directive?" You can enter up to 126 alphanumeric or extended ASCII characters. **Draft** displays:

```
Place  Value  Find  Jump  Zoom
```

Place Use **Place** to place the layout object on the worksheet.

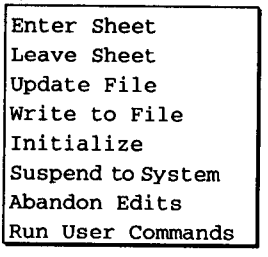
When the layout object is placed, the "Layout Directive?" prompt returns. To place another layout object, enter another message. If you don't want to place another layout object, press <Esc> to return to the main menu level.

Value Select **Value** to edit the message. When you select **Value**, the prompt "Layout Directive?" displays followed by the current message. When you finish editing, press <Enter>.

QUIT

Use **QUIT** to enter and leave hierarchical worksheets, load, update, and write to files, clear the worksheet, suspend to DOS, and abandon edits.

Select **QUIT**. The menu shown at right displays.

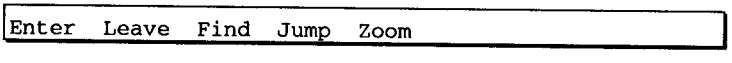


QUIT Enter Sheet

Use the **Enter Sheet** command to view a schematic represented by a sheet symbol, which includes sheet parts and sheetpath parts, in a hierarchical design. To edit the schematic nested inside the one you are working on, select **Enter Sheet**.

If you have made any changes to the current worksheet without saving it, **Draft** displays "Enter – Abandon changes made?"

This message means that unless you save your changes you will lose them. Select **No** to cancel the **Enter Sheet** command. If you select **Yes**, all changes made during this work session are lost, and **Draft** displays:



Place the pointer inside the sheet symbol for the subsheet you wish to enter and select **Enter** sheet. For information about saving your latest design session to a file, see the **QUIT Update File** command.

Draft returns to the **Enter Sheet** menu, so you can enter other subsheets. Press <Esc> to return to the main menu level.

QUIT Leave Sheet

To leave a subsheet, select **Leave Sheet**. When you invoke this command, you move one level up in the schematic hierarchy. If you are at the top of the schematic hierarchy, **Draft** briefly displays the error message "ERROR : Already at the Root Level."

QUIT Update File

Update File writes your latest worksheet design session to a file.

To update a file, select **Update File**. If the current worksheet had been previously loaded from a file, the file is updated. If the current worksheet is unnamed, **Draft** responds "Write to File?" Enter the desired filename.

To update a file other than the current file, use the **QUIT Write to File** command.

Press <Esc> to return to the main menu level.

QUIT Write to File

Write to File saves the current worksheet to any file you specify. When you choose **Write to File**, **Draft** displays "Write to File?"

Enter the desired filename. **Draft** saves the worksheet to the file specified and then returns the **QUIT** menu.

Press <Esc> to return to the main menu level.

QUIT Initialize

Initialize either loads a worksheet file or erases everything from it, thus clearing it. To perform these tasks, select **QUIT Initialize**.

If there are parts on the worksheet and you have made changes since the file was loaded or saved, **Draft** displays "Initialize – Are you sure?" Select **No** to cancel the **Initialize** command and return to the main menu level. Select **Yes** to abandon any changes.

Draft displays "Load File?" Select a file in one of two ways.

You may enter the name of the file to load. If the file exists, the worksheet loads and displays. If the file does not exist, **Draft** displays a blank worksheet and the message "<<<new worksheet>>>."

You may press <Enter> to display a list of schematic files, including "New Schematic," in the current design directory.

Press <Esc> to return to the main menu level.

QUIT Suspend to System

Suspend to System temporarily leaves **Draft** and the worksheet, saves the worksheet in memory, and returns to the operating system. Once you have suspended **Draft**, you may run operating system commands, including using other programs, so long as there is enough system memory.

▲ *CAUTION: Always save your worksheet to a disk file before using* ***Suspend to System.***

To suspend to the operating system, select **Suspend to System**. **Draft** suspends operation, loads the system command interpreter, and adds an additional ">" to the system command prompt. This is a reminder that **Draft** is suspended and in the "background."

To return to **Draft**, type **EXIT** at the system prompt. **Draft** then returns to the "foreground" and the worksheet you were working on displays.

QUIT Abandon Edits

Select **Abandon Edits** to exit **Draft** and return to the **Schematic Design Tools** screen. If parts have been placed on the worksheet since the last update, **Draft** displays "Abandon – Are you sure?" Select **No** to cancel the command. Select **Yes** to quit and return to the **Schematic Design Tools** screen.

QUIT Run User Commands

Use this command to quickly exit **Draft** and run an externally-defined operating system command.

Select **Run User Commands**. **Draft** suspends to the operating system and issues the command DRAFTUSR. You can create DRAFTUSR.BAT, a batch file containing system commands, or you can create DRAFTUSR.EXE or DRAFTUSR.COM containing compiled commands of your choice. DRAFTUSR may be in the current design directory or elsewhere in the system, in which case, the path the operating system searches must be set to find DRAFTUSR. For more information on writing batch files and setting the search path, see your operating system's configuration documentation.

After DRAFTUSR runs to completion, **Draft** prompts "Press any key to continue." Press any key to return to **Draft**.

△ *NOTE: It is possible to use macros and the* ***Block Text Export*** *command to create DRAFTUSR.BAT. For example, you can set up a macro to move to a clear area of a schematic, place one or more lines of text, and do a* ***Block Text Export*** *of this text to the current directory. The macro could then select* ***QUIT Run User*** *to perform the tasks specified in the exported text.*

REPEAT

Use **REPEAT** to duplicate the last entered object, label, or text string and place it on the worksheet.

The location of the duplicate object, label, or text is determined by the Repeat parameters specified with **SET Repeat Parameters**. You can also use **SET Repeat Parameters** to automatically increment or decrement the numeric suffix of a duplicated module port, label, or text string. For details on the **SET Repeat Parameters** command, see the *SET* command on the next page.

For an example of how to use the **REPEAT** command, see chapter 6 in the *Schematic Design Tools 386+ User's Guide.*

SET

Use **SET** to control the following **Draft** options:

- Panning the screen automatically
- Creating backup files
- Dragging buses when rubberbanding
- Ringing the error bell
- Having the left mouse button execute <Enter> when released
- Macro prompting
- Drawing non-orthogonal wires
- Showing pin numbers
- Turning off the standard title block
- Displaying pointer coordinates, grid dots, and grid references
- Release the pointer from the stay-on-grid constraint
- Setting repeat parameters
- Changing the worksheet size, A through E
- Making certain items visible or invisible in zoom scale 2
- Setting the cursor style to a pointer or cross.

To change the status of an option, select **SET. Draft** displays the menu shown at right.

If you prefer options other than the defaults, you may change them automatically every time **Draft** runs using an initial macro. See the **MACRO** command in this chapter and *Chapter 1: Configure Schematic Tools* for information about initial macros.

Set

Auto Pan	YES
Backup File	YES
Drag Buses	NO
Error Bell	YES
Left Button	NO
Macro Prompts	YES
Orthogonal	YES
Show Pins	YES
Title Block	YES
Worksheet Size	A
X,Y Display	NO
Grid Parameters	
Repeat Parameters	
Visible Lettering	
Cursor Style	

SET Auto Pan

Auto Pan controls movement past the screen boundary. While **Auto Pan** is turned on, when the pointer crosses a screen boundary, the screen pans in that direction.

When you select **Auto Pan**, you then choose between **Yes** and **No**. **Yes** turns on auto panning; **No** turns it off.

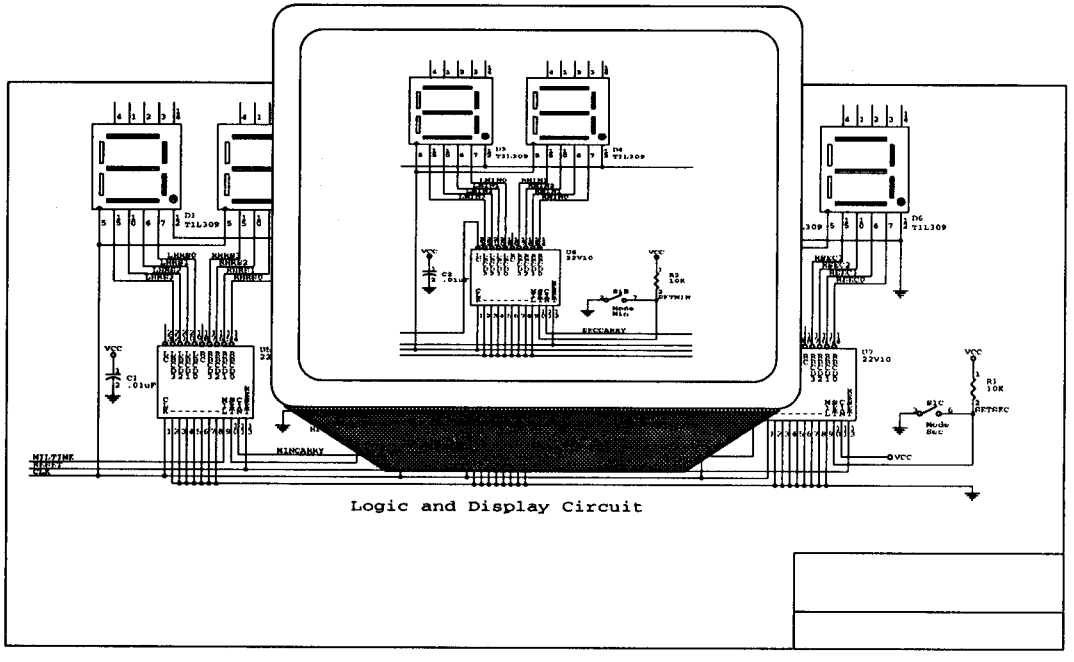


Figure 2-13. Panning changes the area of the schematic that displays.

SET Backup File

Backup File controls whether or not **Draft** creates a backup file of your worksheet when you write or update files using the **QUIT** command. The backup file contains the previous version of your edited worksheet.

When you select **Backup File**, you then choose between **Yes** and **No**. **Yes** turns on the creation of backup files; **No** turns it off.

△ *NOTE: Turning off **Backup File** can be dangerous. If your file should accidentally be damaged or erased, you will be unable to recover it.*

SET Drag Buses

Use **Drag Buses** to stretch buses, rubberband-like, when you use the **BLOCK Drag** command. Because there are more points to locate when rubberbanding, system performance slows down when you use **BLOCK Drag** with **Drag Buses** turned on.

When you select **Drag Buses**, you then choose between **Yes** and **No**. **Yes** turns on rubberbanding buses; **No** turns it off.

SET Error Bell

Error Bell turns the error bell (your computer's speaker) on and off. When you turn this option on and you are in **Draft**, error messages and errors sound the speaker.

When you select **Error Bell**, you then choose between **Yes** and **No**. **Yes** turns on the error bell; **No** turns it off.

SET Left Button

When **Left Button** is on, releasing the left mouse button executes the <Enter> key for command line commands only. Pressing the left mouse button continues to select the command highlighted in pop-up menus.

For example, suppose you select the **PLACE Wire** command and the "Begin Find Jump Zoom" command line displays at the top of the screen. To select **Begin** with the mouse when **SET Left Button** is turned off, you click the left button once to display the pop-up menu and once again to select **Begin**. When **SET Left Button** is turned on, you instead press the left button and hold it down while you move the highlight to **Begin**, then release the left button. This selects **Begin** and saves one button click.

When you select **Left Button**, you then choose between **Yes** and **No**. If you select **Yes**, releasing the left button on your mouse executes <Enter>. If you select **No**, releasing the left button on your mouse does not execute <Enter>.

SET Macro Prompts

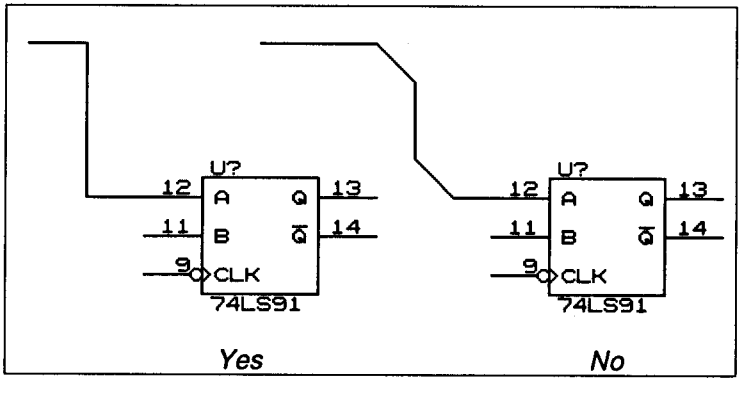
When **Macro Prompts** are turned on, the commands making up your macros display on the screen when the macro runs. Turn **Macro Prompts** on when debugging macros or to watch the commands being performed when you run the macro.

After selecting **Macro Prompts**, you choose between **Yes** and **No**. **Yes** turns on macro prompts; **No** turns them off.

△ *NOTE: Setting **Macro Prompts** to **No** turns screen redraws off when a macro runs. This speeds up macro execution. As a result, **Auto Pan** is set to **No** during macro execution and macros using the **ZOOM** command won't work.*

SET Orthogonal

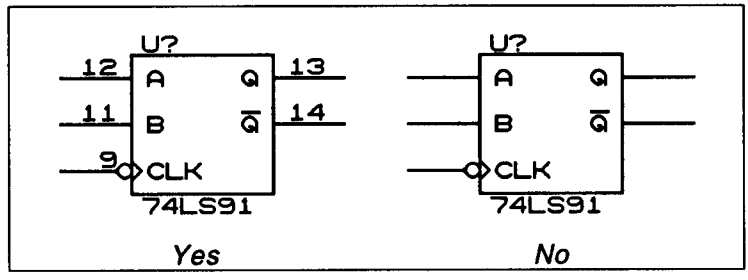
When **Orthogonal** is on, wires and buses are drawn orthogonally (perpendicular to each other). When turned off, wires and buses can be drawn at any angle.



As the above figure shows, when you select **Orthogonal**, you then choose between **Yes** and **No**. **Yes** restricts wires and buses to connections at 90° angles; **No** places no restrictions on the angles and so you can draw wires and buses at any angle.

SET Show Pins

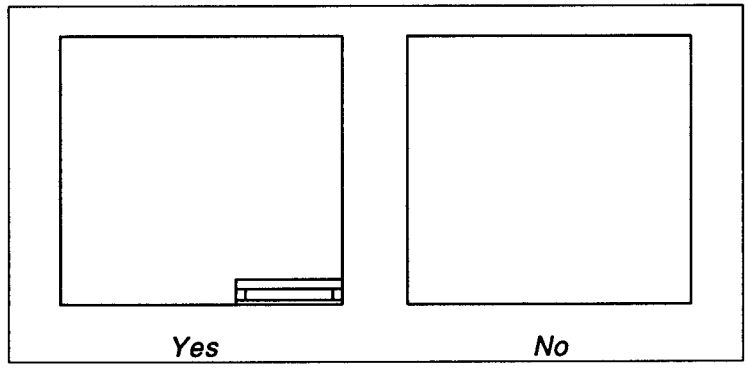
When **Show Pins** is turned on, pin numbers for library parts are shown on the screen and in worksheet hardcopies. When disabled, pin numbers are not shown on the screen or in hardcopies.



After you select **Show Pins**, you then choose between **Yes** and **No**. As the figure above shows, **Yes** turns on the display of pin numbers; **No** turns it off.

SET Title Block

When **Title Block** is turned on, **Draft** puts the standard title block on the worksheet. With the option turned off, you may create a custom title block using the **PLACE Wire/Bus** and **PLACE Text** commands.



When you select **Title Block**, you then choose between **Yes** and **No**. **Yes** turns on the display of the title block; **No** turns it off.

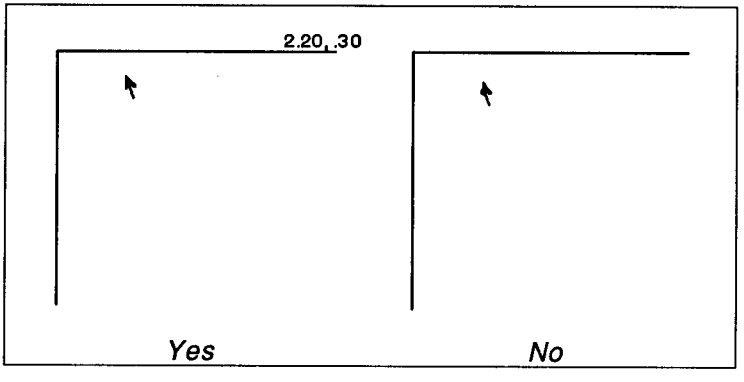
SET Worksheet Size

Use **Worksheet Size** to select the worksheet size, A through E.

After you select **Worksheet Size**, choose a letter from **A** through **E**. The exact dimensions of each worksheet size are defined in the **Template Table** on the **Configure Schematic Design Tools** screen. For more information, see *Chapter 1: Configure Schematic Tools.*

SET X,Y Display

When **X,Y Display** is turned on, the upper right part of the prompt line shows the pointer coordinates. The work-sheet origin (0,0) is the upper left corner. Coordinates do not appear on the screen until the pointer is moved.



When you select **X,Y Display**, you then choose between **Yes** and **No**. **Yes** displays pointer coordinates; **No** doesn't.

▲ *CAUTION: When grid coordinates are not displayed, it is easy to make small errors when placing parts, wires, and buses. These errors may cause problems when using the* ***Check Electrical Rules*** *reporter and* ***Create Netlist*** *processor. This is because wires and buses may look as if they are connected when they are not.* ***Check Electrical Rules*** *and* ***Create Netlist*** *interprets these incomplete connections as opens. Do not place parts, wires, or buses in the worksheet with* ***X,Y Display*** *disabled or with* ***Stay on Grid*** *turned off (described on the next page).*

SET Grid Parameters

Select **Grid Parameters**. **Draft** displays the menu shown at right. Use **Grid Parameters** to display grid references, keep the pointer on grid, and display grid dots.

Set Grid Parameters

Grid References	NO
Stay on Grid	YES
Visible Grid Dots	NO

Grid References

When **Grid References** are turned on, **Draft** displays an alphanumeric border on two of the four worksheet sides. The top border shows grid reference numbers, and the left border shows reference letters. The borders are scaled to the size of the worksheet. Grid references can be used as destination for the **JUMP** command.

Stay on Grid

When **Stay on Grid** is turned on, the pointer location is confined to the predefined grid. When disabled, the pointer may be moved off-grid to any position on the worksheet at a resolution ten times that of the grid.

After you select **Stay on Grid**, you select either **Yes** or **No**. **Yes** restricts the pointer to the grid; **No** does not.

▲ *CAUTION: Placing parts, wires, and buses with **Stay on Grid** turned off may cause problems, because wires and buses may look as if they are connected when they are not. Unconnected wires and buses are reported as errors by the **Check Electrical Rules** reporter and **Create Netlist** processor. To avoid trouble, do not place parts, wires, or buses in the worksheet with **Stay on Grid** turned off.*

Visible Grid Dots

When **Visible Grid Dots** are turned on, grid dots display on the worksheet. When grid dots are visible and **SET Stay on Grid** is turned on, it is easier to place parts at specific points on the worksheet.

The distance between grid dots depends on the current zoom scale. The table at right lists the grid dot spacing at different zoom scales.

Zoom scale	Grid dot spacing
1	1/10 of an X or Y unit
2	2/10 of an X or Y unit
5	1/2 of an X or Y unit
10	1 X or Y unit
20	2 X or Y units

After you select **Visible Grid Dots**, select either **Yes** or **No**. **Yes** displays grid dots; **No** does not.

SET Repeat Parameters

Repeat Parameters are used to determine how the **REPEAT** command works.

Select **Repeat Parameters.** **Draft** displays the menu shown above.

Set Repeat Parameters

X Repeat Step	0
Y Repeat Step	+ 1
Label Delta	+ 1
Auto Increment	NO

X Repeat Step

X Repeat Step determines the number of unit steps in the X direction the object being repeated is offset from the previous object. The X direction goes horizontally across the worksheet, with positive to the right and negative to the left of the current pointer position. A unit step in the X direction is defined as $^{1}/_{10}$ of an X unit when **SET Stay on Grid** is turned on and $^{1}/_{100}$ of an X unit when **SET Stay on Grid** is turned off. You can view the change in X units as the pointer moves on the display by turning on the **SET X,Y Display** option.

When you select **X Repeat Step,** the prompt "X Repeat Step?" displays. Enter any integer.

Y Repeat Step

Y Repeat Step determines the number of unit steps in the Y direction the repeated object is offset from the previous object. The Y direction goes vertically on the worksheet, with positive above and negative below the current pointer position. A unit step in the Y direction is defined as $^{1}/_{10}$ of a Y unit when **SET Stay on Grid** is turned on and $^{1}/_{100}$ of a Y unit when **SET Stay on Grid** is turned off. You can view the change in Y units as the pointer moves on the display by turning on the **SET X,Y Display** option.

When you select **Y Repeat Step,** the prompt "Y Repeat Step?" displays. Enter any integer.

Label Delta

Label Delta determines how much the numeric suffix information on labels, module ports, and text changes, and in what direction, when they are repeated.

When you select **Label Delta**, the prompt "Label Repeat Delta?" displays. Enter a whole number. If you enter a positive number, the numeric suffixes on labels, module ports, and text are incremented by that number when they are placed with the **PLACE** command or the **REPEAT** command. If you enter a negative number, label suffixes are decremented by that number when placed with the **PLACE** command or the **REPEAT** command.

Auto Increment

When **Auto Increment** is turned on, the numeric suffix (if it exists) of labels, module ports, and text is automatically incremented or decremented when the objects are placed on the worksheet with the **PLACE** command. After a label, module port, or text string is placed, its numeric suffix is changed by the amount specified by the **Label Delta** command.

When you select **Auto Increment**, you then choose between **Yes** and **No**. **Yes** turns on automatic incrementing or decrementing of labels, module ports, and text; **No** turns it off.

SET Visible Lettering

You can choose to have some items on the worksheet display in zoom scale 2. When you select **Visible Lettering**, the menu shown at right displays.

To make an item visible, select it and choose **Yes**. To make it invisible, select it and choose **No**.

This option only affects how the lettering of the object appears on the screen in zoom scale 2.

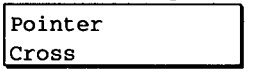
Visible lettering for Scale 2

Part Field	YES
Pin Number	NO
Pin Name	NO
Label	NO
Text	YES
Module Port	NO
Power Value	YES
Sheet Name	YES
Sheet Net	NO
Title Block	YES

SET Cursor Style

Cursor Style determines whether **Draft** displays a crosshair cursor along with the arrow pointer.

Select **Cursor Style. Draft** displays the menu at right.

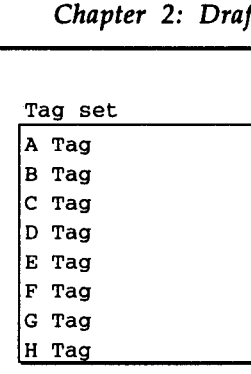


Pointer Select **Pointer** for an arrow pointing up and to the left: ↖.

Cross Select **Cross** to add a full-screen crosshair to the arrow. The crosshairs are especially useful for alignment when you place objects or edit a schematic.

TAG

The **TAG** command identifies and remembers locations on the worksheet. You can specify eight locations (A through H) using the pointer. Tagged locations can be used as destinations for the **JUMP** command. Tags are invisible when set on the worksheet and are not saved with the worksheet.



To set a tag, place the pointer at a location you want to remember. Then select the **TAG** command. **Draft** displays the menu shown above.

When the **TAG** menu displays, select the tag to set from the menu. Once selected, **Draft** remembers the tag location. Once the tag is set, **Draft** returns to the main menu level.

ZOOM

ZOOM zooms in or out from the worksheet, changing the amount of detail you see on the screen. You can select from the five zoom levels described below.

Scale 1 The most detailed zoom scale. All lettering is visible.

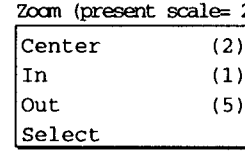
Scale 2 The second most detailed zoom scale, representing 1/2 of scale 1.

Scale 5 The third most detailed zoom scale, representing 1/5 of scale 1.

Scale 10 The fourth most detailed zoom scale, representing 1/10 of scale 1.

Scale 20 The least detailed zoom scale, representing 1/20 of scale 1.

To change the zoom scale, select **ZOOM. Draft** displays the menu shown at right.



ZOOM Center Selecting **ZOOM Center** centers the displayed portion of the sheet around the pointer. This command is useful for centering an object on the screen so you can easily edit it.

For example, if the display of an object is partially off the screen, center it by placing the pointer near the object and selecting **ZOOM Center.**

The number in parentheses shows the current zoom scale.

ZOOM In **ZOOM In** zooms in on the worksheet for a more detailed view. The number in parentheses shows what the zoom scale will be the next time you select **ZOOM In.**

ZOOM Out **ZOOM Out** zooms out to display a larger worksheet area. The number in parentheses shows what the zoom scale will be the next time you select **ZOOM Out.**

ZOOM Select Use **ZOOM Select** to select any one of five zoom scales **(1, 2, 5, 10,** or **20)** from a menu.



Guidelines for creating designs

This chapter provides information to help you create netlists from your schematics. To obtain predictable results, you should follow several simple rules. You can then use **Create Netlist** and **Create Hierarchical Netlist** to create netlists in over thirty formats. If you plan to use your schematics with any of OrCAD's other tools, such as **PC Board Layout Tools**, following these rules greatly simplifies the task of creating a usable netlist.

If these rules are not followed, the database that contains the connectivity of all of the components may contain incorrect connections.

Label names

Draft is quite liberal concerning label names: it accepts any of the standard ASCII printable characters in naming labels and buses. Some netlist formats are more restrictive. You should be aware of any naming restrictions imposed by your target netlist format. For instance, **PC Board Layout Tools** does not allow spaces, left parentheses or braces, or right parentheses or braces. See the file NETFORMS.DOC in **View Reference** for the syntax rules.

Wire labels

Use labels to connect signals from one worksheet area to another without using wires or buses.

For example, assume you have a signal labeled ABC in your worksheet and you would like to connect another object on the worksheet to the same signal. Instead of drawing a wire from ABC on one side of the worksheet to the other object, you can identify each signal with a label, as shown in figure 3-1.

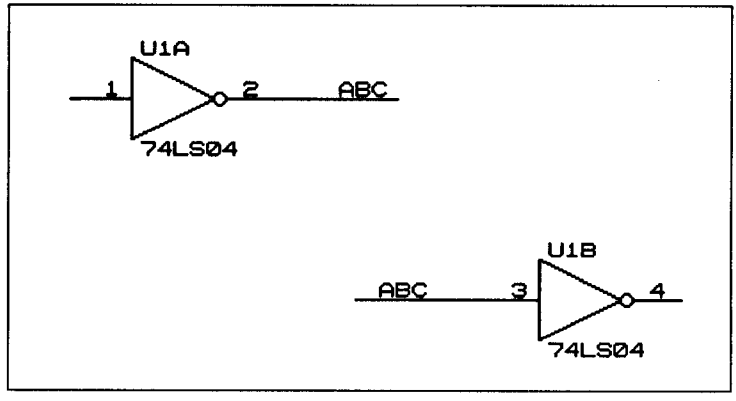


Figure 3-1. Using a label.

△ *NOTE: If you enable the* ***Do not append sheet number to labels*** *option in the local configuration of IFORM or HFORM, the labels for local nets on one sheet should be different from those on other sheets. This way, your netlist will not mix local nets on one sheet with local nets of another sheet.*

Bus labels

Labels are also used for naming buses and bus members. For **Create Netlist** and **Create Hierarchical Netlist** to properly associate a bus with its individual members, both the bus and the wires (bus members) branching from the bus *must* be labeled following a specific format.

Bus labels must be in the form:

```
BUSNAME[x..y]
```

The *prefix* of the label, indicated above by BUSNAME, represents the name of the bus. The *suffix* of the label, indicated above by [x..y], specifies a range of decimal integers representing the number of wires branching from the bus. x represents the first wire number in the bus; y represents the number of the last wire branching from the bus. There will be $(y - x + 1)$ wires in the bus. The prefix and suffix must not have spaces between them.

Examples are:

```
ADDR[0..31]     (This bus has 32 members.)
DATA[16..31]    (This bus has 16 members.)
CONTROL[1..4]   (This bus has  4 members.)
A[100..190]     (This bus has 91 members.)
```

Figure 3-2 shows a bus label properly positioned.

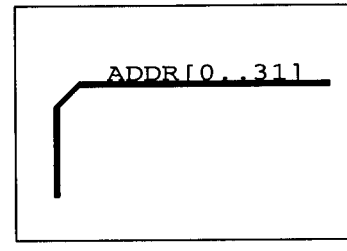


Figure 3-2. Correct position for a bus label.

A bus label may be placed anywhere on the bus, as long as the label's hotpoint is touching the bus. The hotpoint is the lower-left point of the first character; see the **PLACE Label** command in *Chapter 2: Draft* for a description of label hotpoints. A bus may have more than one label placed on it.

Wires branching from a bus must be labeled in a form corresponding to the bus. For example, suppose a bus is labeled as follows:

```
BUSNAME[0..9]
```

Then the wires branching from the bus must have labels of the form shown below:

```
BUSNAMEx
```

where *x* is a decimal integer in the range of 0–9. BUSNAME and *x* must not have space between them.

Figure 3-3 shows an example of a properly labeled bus and its members.

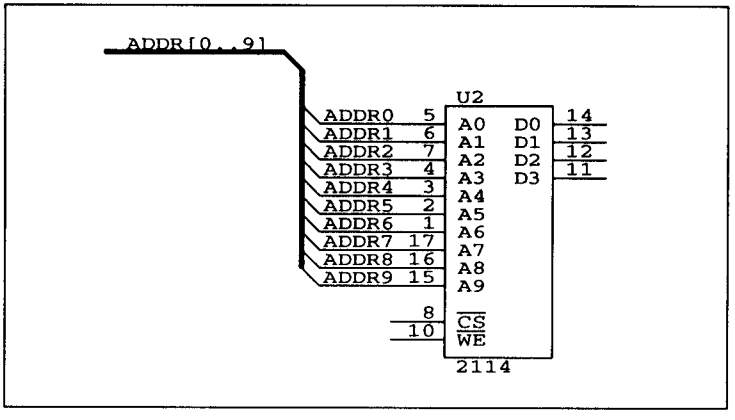


Figure 3-3. Label format required by ***Create Netlist*** *and* ***Create Hierarchical Netlist*** *for buses and bus members.*

Multiple labels on a bus

A bus may have more than one label placed on it. In actual applications, bus labels may be placed anywhere on the bus and still be associated with their respective bus signal labels. As a rule, you may place any number of labels on a bus or wire. However, only the busname in the form:

```
BUSNAME[x..y]
```

will be used for the netname. The other labels are strictly for your convenience. They are not used in netlists.

Combining labels

Often times it is necessary to refer to bus members by names other than that of the bus. For instance, one member of a bus might be MEM10, but you still want to refer to it as C\S\ (this notation represents CS with a bar over it, meaning the complement of CS). This is an example of combining labels. Figure 3-4 shows label MEM[0..11] as a bus containing 12 members. U1 is connected to the bus via labels MEM0 through MEM11.

Notice on the left side of the figure that label MEM10 has label C\S\ placed next to it, and that label MEM11 has label W\E\ next to it. On the other side of the figure, C\S\ and W\E\ are labels that have been placed on pins 8 and 10 of U2 and U3.

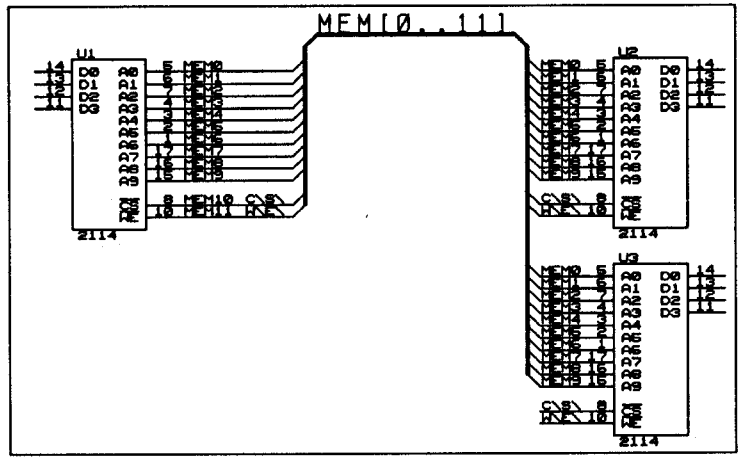


Figure 3-4. Combining labels.

This example shows how to connect signals MEM10 and MEM11 to U2, by labeling them as C\S\ and W\E\, respectively. In the case of U3, C\S\ and W\E\ signals are connected to the 2114 device without being physically connected to the bus.

Intersheet connections

For **Create Netlist** and **Create Hierarchical Netlist** to list interworksheet connections properly, they must be specified on the worksheets following certain conventions.

Lateral intersheet connections, as in flat file structures, are established by placing complementary module ports with the same names on different sheets.

Vertical intersheet connections, as in hierarchical designs, are established by placing module ports having the same names as nets placed in a sheet symbol one level up in the hierarchy.

When buses connect to module ports or nets, the bus-range format explained in the previous section must be used in both module port names and net names. The ranges specified in module port and net names must match the ranges of the buses to which they connect.

The *prefix* of a module port name or net name need not be identical to the prefix of the label on the bus to which it connects. However, making them the same is a good practice, unless there is a reason for doing otherwise.

Figure 3-5 shows three examples of buses properly connected to module ports. Notice the bus label suffix, [0..15], is identical to the module port suffix.

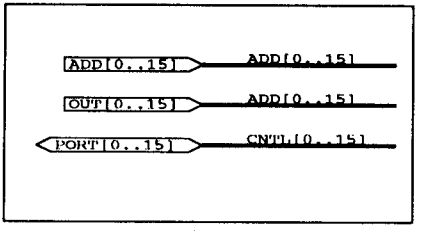


Figure 3-5. Connecting buses to module ports.

Figure 3-6 shows a more detailed example of connecting bus signals to module ports.

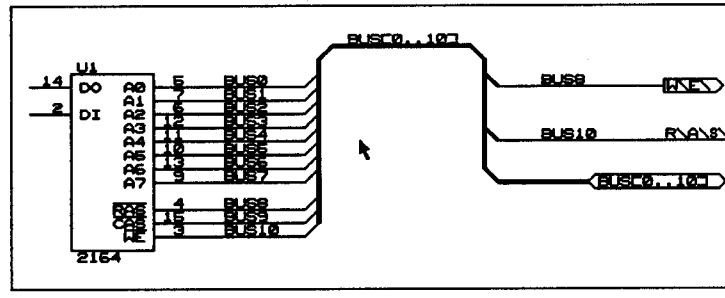


Figure 3-6. Connecting buses to module ports.

The main bus in figure 3-6 is labeled BUS[0..10]. Electrical connection between the wires branching from this bus and component U1 is established by labeling the wires BUS0 through BUS10.

The signal labeled BUS8 is conducted off the worksheet through the module port named W\E\. The wire label for this signal is based on the format of the bus label. But, as shown, the name of the module port on this wire does not have to match the wire label. However, to establish connection with a module port on some other worksheet, the other module port must also be named W\E\.

The bus BUS[0..10] also leaves the worksheet. It connects to a module port having the same name. While the prefix, BUS, *could* legally be different, the suffix specifying the number of signals must be identical.

The signal labeled R\A\S\ can be connected to a wire elsewhere in the worksheet simply by giving the other wire the same label. Because the R\A\S\ signal is transferred through the bus, BUS[0..10], it must also have a label based on the format of the bus label; in this case, BUS10.

Module ports connected to individual, non-bus wires may be named in any format. They are not required to have a suffix. Figure 3-7 shows typical examples of non-bus signals connected to module ports.

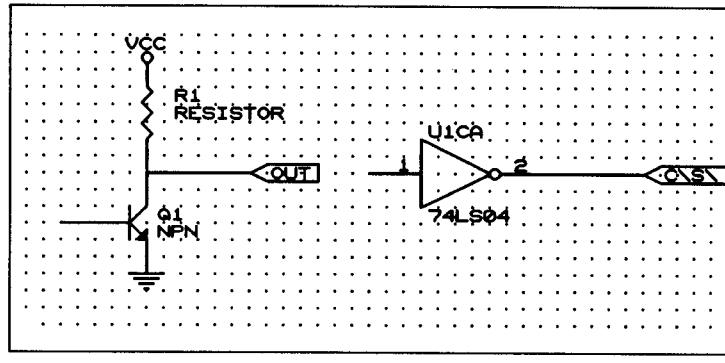


Figure 3-7 Module ports connected to non-bus signals.

△ *NOTE: It is not necessary to place the label BUS[0..10] on the bus since the module port BUS[0..10] is on the bus. At every module port, a label with the same name is assigned when the connectivity database is constructed.*

Splitting buses

You can split buses in your worksheet. Figure 3-8 shows an example.

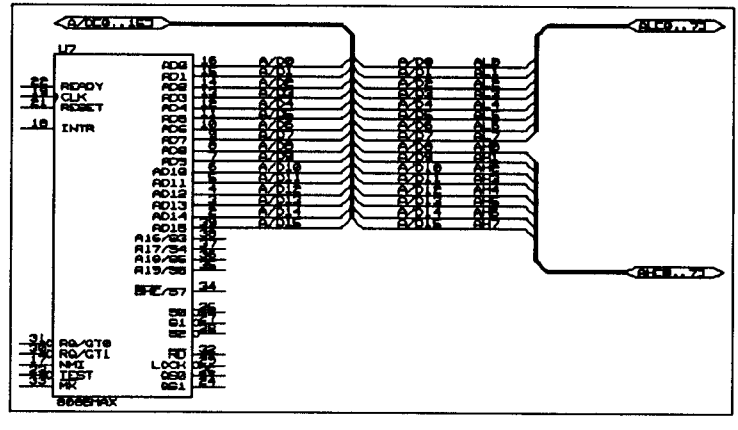


Figure 3-8 Splitting a bus.

As shown in figure 3-8, you can split the same signal off a bus multiple times by attaching more than one wire with the same label.

Bus mapping

If you are processing subsets of bus signals with a common logic block, it is economical to create a subsheet that contains the logic and map bus signals to the subsheet's input signals. To map one set of signals to another, use the technique shown in Figure 3-9.

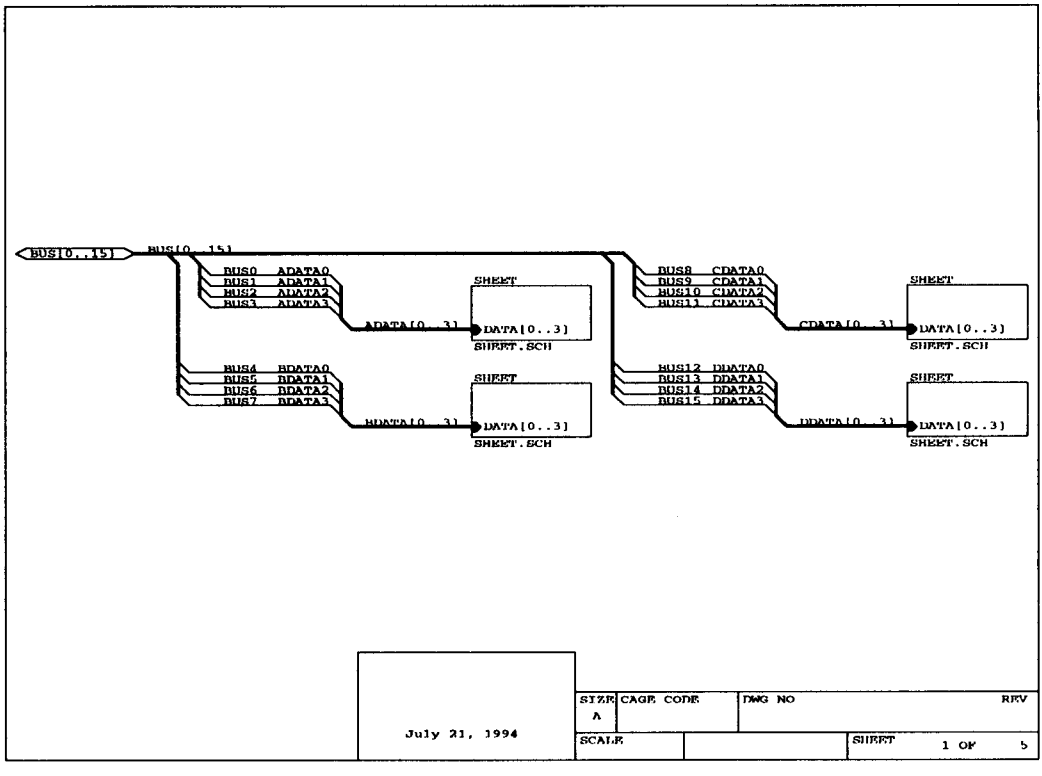


Figure 3-9. Example of bus mapping method.

The bus BUS[0..15] is split into four groups of four signals each. Each individual signal has two labels—BUS4 and BDATA0, for example—so that SHEET.SCH can refer to the signals internally as DATA[0..3]. This mapping gives the netlister unique names for the signals: ADATA[0..3], BDATA[0..3], and so on.

Named properties

The EDIF 2 0 0 netlist format includes named properties as a means of passing information from the designer to the EDIF reader. Named properties are passed in the eight part fields on each part. Passing a named property is a two-step process:

1. Entering the information in the part field.
2. Telling the EDIF netlist format file to convert the part field to a named property.

Step 1 is done in **Draft**, where you enter the information in the following form:

propertyName=value

The section *Editing parts*, under the description of the **EDIT** command in *Chapter 2: Draft*, explains how to enter information in part fields.

Step 2 is done on the **Configure Netlist Format** screen, with FEDIF.EXE or HEDIF.EXE as the selected format. *Appendix B: Netlist formats* describes the **Convert PartFields to Named Properties** configuration option in the description of the EDIF netlist format. The section *Configure IFORM* in *Chapter 11: Create Netlist* describes IFORM's local configuration screen and HFORM's local configuration is described in *Chapter 12: Create Hierarchical Netlist*.

Handling and isolating power

Power connections are handled in a number of ways. Most parts in the libraries supplied by OrCAD have defined power and ground pins. These pins are hidden from the drawing, but nonetheless are part of the symbol definition.

To make connections from the outside world to the hidden power pins in the library part, **Draft** uses a *power object* (placed with the **Place Power** command).

For example, assume you have a CMOS device placed in the worksheet. In the CMOS library source file, this device is defined to have a VDD and VSS power pin. To connect another signal from the outside world to the same VDD potential as in the CMOS device, just connect the signal to a power object named VDD.

Power objects are *global* in scope. A global object is one whose signal (power in this case) connects to all other global signals of the same name. Connectivity between global objects of the same name holds for all worksheet file structures.

The programs that create the netlist connect all power objects and signals of the same name. This power handling ability makes it easy to isolate different power sources.

These programs also treat certain parts in part libraries as power objects if they are defined a special way. The four types of grounds in the DEVICE.LIB library (Earth, Field, Power, and Signal grounds) are good examples.

To be treated as a power object, a device is defined as having zero parts per package, only one pin and no reference designator. The pin is defined to be of type PWR. Figure 3-10 shows the source file definition of the GND POWER symbol found in the DEVICE.LIB part library. Significant entries appear in bold. When a part has these characteristics, **Draft** treats the part from the library as a power object.

```
'GND POWER'
{X Size =}2 {Y Size =}1  Parts per Package =}0 T1 PWR 'GND'
      {00000000011111111112}
      {.....................}
      {01234567890123456789}
{  0.0}####################
{  0.1}....................
{  0.2}....................
{  0.3}...###############...
{  0.4}....................
{  0.5}....................
{  0.6}......#########.....
{  0.7}....................
{  0.8}....................
{  0.9}.........###........
{  1.0}....................

VECTOR
LINE     +0.0 +0.0 +2.0 +0.0
LINE     +0.3 +0.3 +1.7 +0.3
LINE     +1.4 +0.6 +0.6 +0.6
LINE     +0.9 +0.9 +1.1 +0.9
END
```

Figure 3-10. Source file for the GND POWER symbol.

In the example in figure 3-10, notice the pin name is GND. If this power ground symbol is placed on a worksheet, it represents it as being connected to any other object with a power pin named GND.

There are several ways to create different power supplies in a design. One way is to simply place a power object on the worksheet with the **Place Power** command, select the **Value** command and change the value to be whatever you want to distinguish it from other power objects.

Another approach is to create a custom power object and give its pin a unique name using the OrCAD part editor, **Edit Library.**

Or, edit the definition of a power object in a library source file and give its pin a unique name. Then, update the binary form of the library by running the **Compile Library** tool.

For example, in the part library source file, you could edit the signal ground definition, changing the name of its pin from GND to SGND. Or, you could edit the power ground definition, and change its pin name from GND to PGND. Each type of ground will then be connected to any other object with a power pin defined as SGND or PGND.

To find power pin numbers on library components, use the **LIBRARY Browse** command in **Draft** or **Edit Library**.

Connecting power objects with different names

In OrCAD libraries, many of the devices are defined with the positive supply voltage pin named VCC. Others are defined with the positive supply voltage pin named VDD. To operate pins of both types from the same power supply, you must connect those pins to one another.

Similarly, many of the libraries have return power pins defined as GND or as VSS. The same requirement applies for connecting both types to the same potential.

To connect power supply pins together, or connect a power supply pin to any other supply voltage, you place a separate power object for each different supply in the worksheet. Name one power object with the same name as one of the supply voltages, VDD for example. Name the other power object with the same as the remaining supply voltage, VCC for example. Finally, connect the two power objects together with a wire. The following figure shows how this is accomplished.

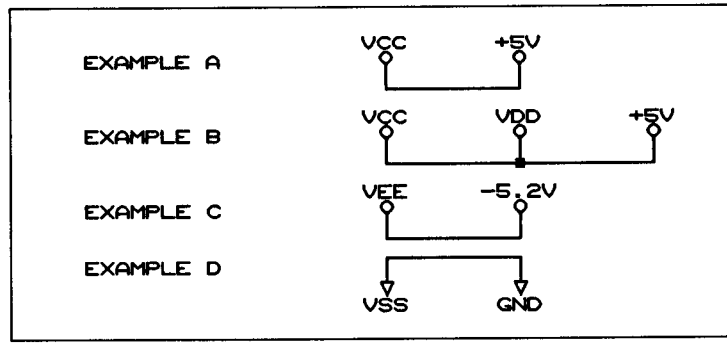


Figure 3-11. Power supply connections.

Example A shows a power object named VCC connected to a +5 volt power supply. In the connectivity database, every object with a VCC pin is connected to a +5 volt power supply. This assumes your design contains a power supply with a power object named +5V attached.

Example B shows two power objects, VCC and VDD, connected to a +5 volt power supply through a power object named +5V.

Example C shows a power object, VEE, connected to a –5.2 volt power supply through a power object named –5.2V.

Example D shows a power object, VSS, connected to a power object named GND. This electrically connects the two types of grounds in the netlist.

Isolating power and ground

One way to isolate power in the worksheet is to edit parts in the source library file, giving their power pins new names, then update the library with **Compile Library**. This is time consuming and makes it difficult to keep track of which parts are to be used on which schematic for any particular supply. There is, however, another way to isolate power.

To isolate power without editing the part libraries, connect a module port to a power object. When the connectivity database is built, the name of the module port supersedes the library name of the power object. Only the module port name is used in conducting the power signal from one worksheet to another.

If a power object is to transfer isolated power from one worksheet to another, either in a linked file or hierarchical structure, it must be connected to a module port of type **Unspecified**. The **Check Electrical Rules** step in **Create Netlist** does not accept other types of module ports.

Figure 3-12 shows three examples of power objects connected to module ports.

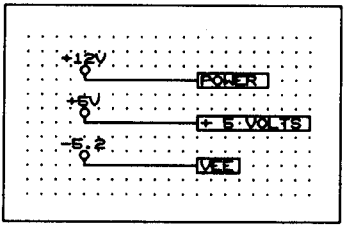


Figure 3-12. Isolating power on a worksheet by connecting power objects to module ports.

Handling power in a hierarchy

Power in a hierarchy is handled in much the same way as it is in a linked file design. Power objects connect to all other objects with the same name. If a module port is connected to a power object, the module port supersedes the power object for conducting the signal off the worksheet.

When passing power from a worksheet through a module port up to a sheet symbol, you must place a sheet net in the sheet symbol to conduct the power signal.

Example of isolating power: battery backup

In battery backup applications, main power can be supplied throughout the design with power objects. Backup power can be isolated from the main source by using a module port. Figures 3-13 through 3-15 show this approach to a battery backup application.

This design is a three-sheet hierarchy. The root sheet, shown in figure 3-13, contains the CPU and control circuitry of the design. Two sheet symbols are also placed in the root worksheet. One sheet symbol represents the power supply; the other represents the memory backed up by battery.

Notice a VDD power object is placed in the root worksheet and connects to a +5V power object. Since the 80C51 and the 82C82 power pins are labeled as VDD in their library source files, the +5V and VDD power objects connect +5 volts from the power supply (shown in figure 3-14) to the VDD pins of both devices.

Figure 3-15 shows the battery backed CMOS memory. The memory control signals are conducted from the CPU root sheet through module ports AD[0..7], WE, and A[0..7]. In the POWER SUPPLY worksheet, the power signal to the CMOS MEMORY worksheet is isolated from the +5V supply through a module port named BACKUP.

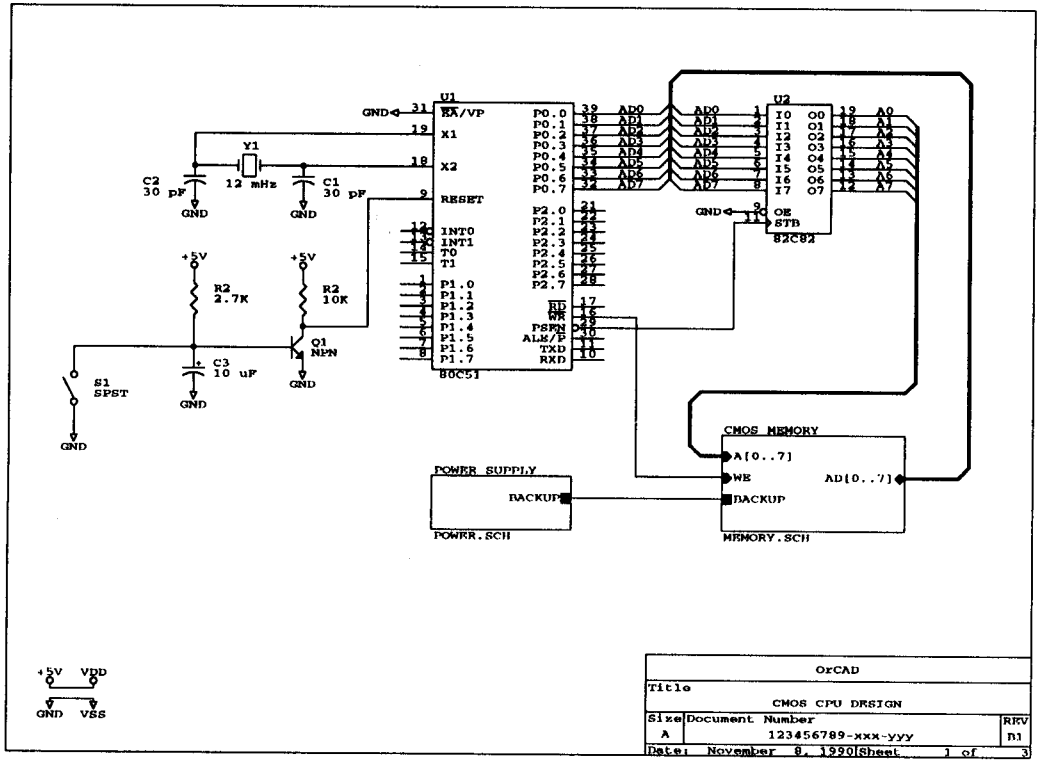


Figure 3-13. Root CPU sheet.

In the CMOS memory sheet (figure 3-15), another module port named BACKUP connects to a power object named VDD, isolating VDD on this sheet from VDD on all of the other sheets.

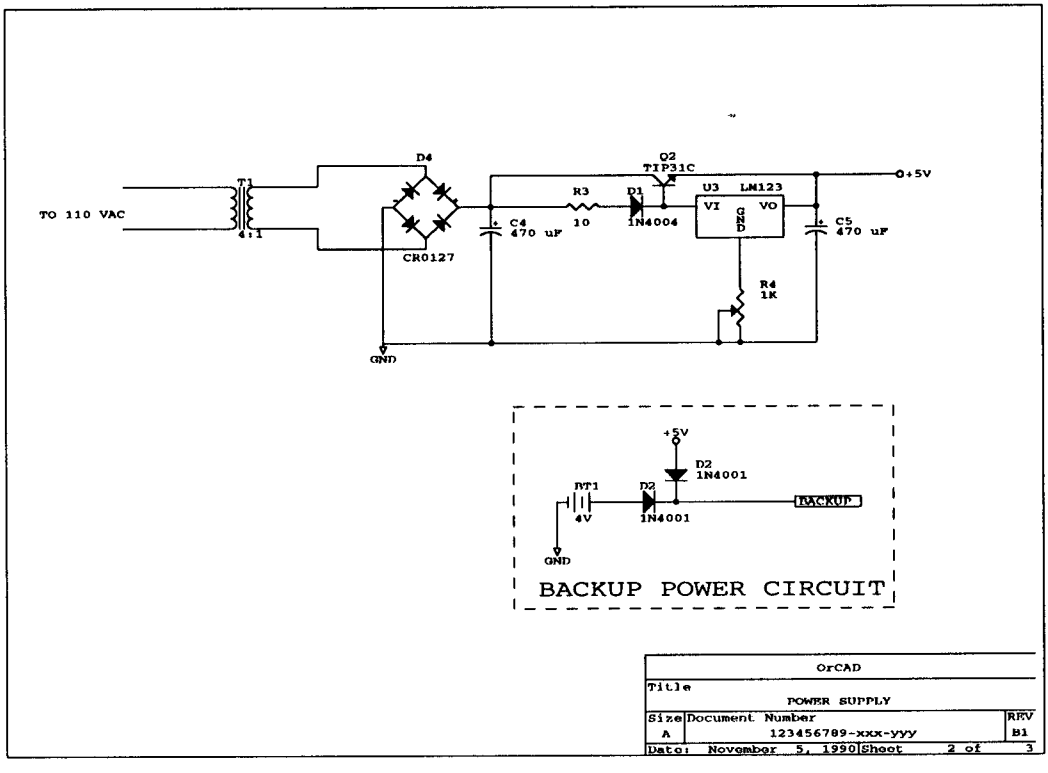


Figure 3-14. Power supply sheet.

GND and VSS power objects are also placed in the CMOS memory worksheet. This connects the VSS power return pins from the memory devices to the power ground object.

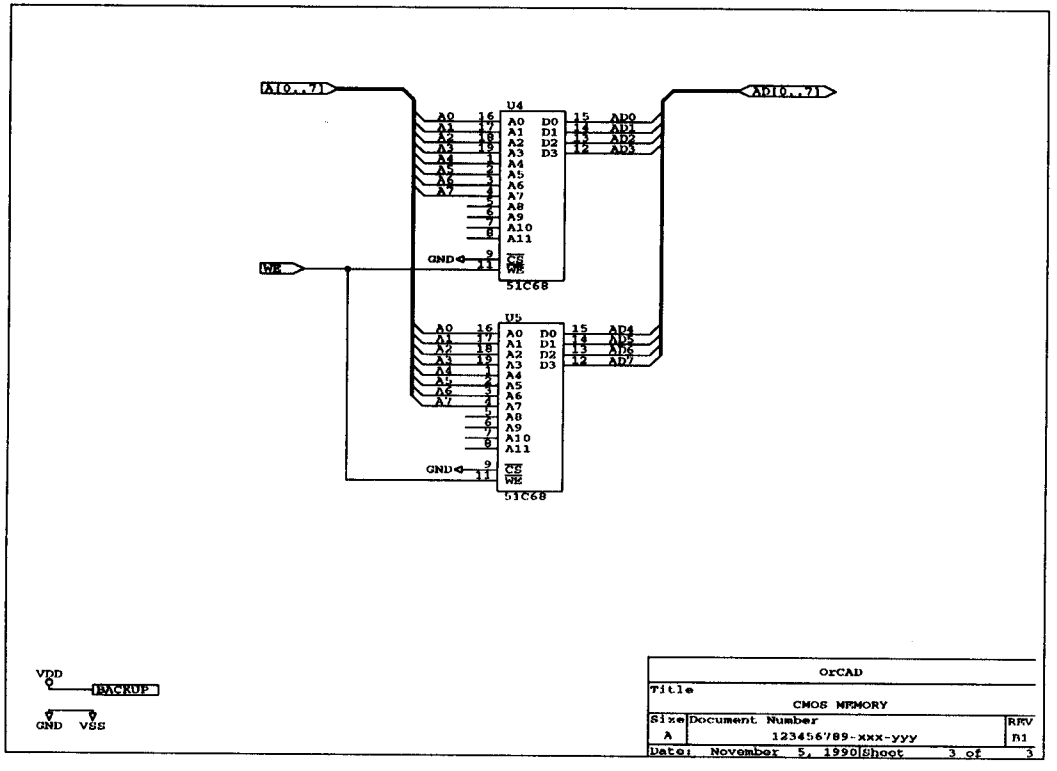


Figure 3-15. CMOS memory sheet.

To summarize, isolate power in a design by conducting it through module ports connected to power objects selectively named to match the names of power pins in library source files.

Although this example design is a hierarchy, it could have been created as a flat file structure. In applications where you isolate power, place all of the circuitry to be isolated in a separate worksheet. This keeps isolated power specific to one worksheet.

Handling physical connectors

Module ports are not intended to be used as physical connectors in a design. Use module ports to connect signals from one worksheet to another. Physical connectors are library parts, since connectors require a reference designator and part value.

To make schematics easier to read, don't separate individual pins in a connector and place them all over the worksheet. This makes finding connector pins difficult, especially in multiple sheet designs. Instead, place a connector on one worksheet and use module ports or labels to connect its pins to other signals in the design. Figure 3-16 shows an example.

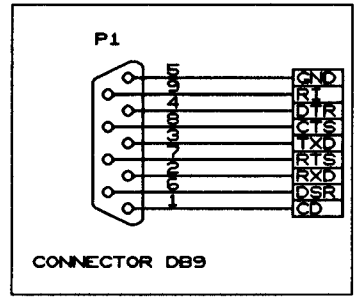


Figure 3-16. Handling connectors.

Make large connectors with alphanumeric pin names using a block symbol with zero parts per package or a grid array.

Figure 3-17 shows part of a library source file definition for an IBM PC 62-pin edge connector. Because of its large size, only part is shown. Since there are no pin *numbers* defined in the source file, **Create Netlist** will use pin *names* instead to identify the pins when it builds the connectivity database. In this definition, B1–B31 and A1–A31 are the pins on the 62-pin edge connector.

```
'CONNECTOR IBM'
REFERENCE 'J'
10    32    0
L1    PAS    'B1'
L2    PAS    'B2'
L3    PAS    'B3'
L4    PAS    'B4'
      .
      .
L30   PAS    'B30'
L31   PAS    'B31'
R1    PAS    'A1'
R2    PAS    'A2'
R3    PAS    'A3'
R4    PAS    'A4'
      .
      .
R30   PAS    'A30'
R31   PAS    'A31'
```

Figure 3-17. Source file for an IBM 62-pin edge connector.



Edit File

The **Edit File** button runs a text editor. When you receive the ESP design environment from OrCAD, it is configured to run a text editor called M2EDIT. However, you can configure the design environment to run the text editor of your choice.

For instructions on how to configure the design environment to run your text editor, see the *ESP Design Environment User's Guide.* To use the M2EDIT editor, see the *Stony Brook M2EDIT Text Editor User's Guide.*

Execution

With the **Schematic Design Tools** screen displayed, select **Edit File.** Select **Execute** from the menu that displays. The **Edit File** screen displays. This screen contains a list box that shows the files you can edit. Select a file from the list box or enter a filename in the **File to Edit** entry box, and then select **OK.**

The file displays on the screen for the configured text editor. When you are done editing the file, quit the text editor, and then select **Cancel** to dismiss the **Edit File** screen.



View Reference

Using **View Reference**, you can view several reference files provided by OrCAD. These files generally have an extension of .DOC and contain information such as:

- List of drivers supported by the design environment
- List of drivers that can be made using GENDRIVE
- Information about library parts

When you receive the ESP design environment from OrCAD, it is configured to run a text editor called M2EDIT. However, you can configure the design environment to run the text editor of your choice.

For instructions on how to configure the design environment to run a different text editor, see the *ESP Design Environment User's Guide.* To use the M2EDIT editor, see the *Stony Brook M2EDIT Text Editor User's Guide.*

Execution

With the **Schematic Design Tools** screen displayed, select **View Reference.** Select **Execute** from the menu that displays. The **Edit File** screen displays. This screen contains a list box that shows the reference files you can view. Select the file to view, and select **OK.**

The reference information displays on the screen for the configured text editor. When you are done viewing the information, quit the text editor, and then select **Cancel** to dismis the **Edit File** screen.

PART III: PROCESSORS

Schematic Design Tools 386+ includes seven processors that read, modify, and then rewrite the design database. Processors generally do not create human-readable reports, but rather create or modify database information.

Part III describes processors and provides instructions for their use.

Chapter 6: *Annotate Schematic* describes how **Annotate Schematic** scans a design and automatically updates the reference designators of all parts in the worksheet.

Chapter 7: *Back Annotate* describes how **Back Annotate** scans a design and updates part reference designators according to the instructions you provide in a "Was/Is" file.

Chapter 8: *Cleanup Schematic* describes how **Cleanup Schematic** scans a design and checks for wires, buses, junctions, labels, module ports, and other objects that are placed on top of each other.

Chapter 9: *Check Design Integrity* describes how to use **Check Design Integrity** to prepare a design before creating a netlist.

Chapter 10: *Creating a netlist* gives an overview of incremental netlisting.

Chapter 11: *Create Netlist* describes how to create a linked and flattened netlist.

Chapter 12: *Create Hierarchical Netlist* describes how to create a hierarchically formatted netlist.

Chapter 13: *Select Field View* describes how to use **Select Field View** to change visible attributes of specified fields on a worksheet.

Chapter 14: *Update Field Contents* describes how to use **Update Field Contents** to load user-defined information into the fields of parts on a worksheet.

Chapter 15: *Decompile Schematic* describes how to use **Decompile Schematic** to create an ASCII text representation of a worksheet.

Chapter 16: *Compile Schematic* describes how to use **Compile Schematic** to convert an ASCII text representation to a worksheet.



Annotate Schematic

The **Annotate Schematic** processor scans a design and automatically updates the reference designators of all parts in the worksheet. This includes updating the corresponding pin numbers (associated with a particular instance of a part) with multiple devices in the package.

Annotate Schematic updates reference designators in the order the parts were placed in the worksheet. **Annotate Schematic** also reports all unused devices. When the worksheet is annotated, all parts may be assigned a new reference designator, including any manually edited parts. To selectively change reference designators and leave others unmodified, use the **Back Annotate** tool or **Draft**.

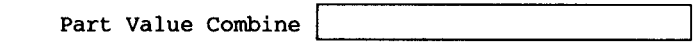
Execution

With the **Schematic Design Tools** screen displayed, select **Annotate Schematic**. Select **Execute** from the menu that displays.

While **Annotate Schematic** runs, messages display on the screen. When the annotation is complete, the **Schematic Design Tools** screen displays.

Key Fields

Annotate Schematic has one key field. It is shown on the **Key Fields** area on the **Configure Schematic Design Tools** screen as follows:



This key field determines how **Annotate Schematic** will group parts in devices that have multiple parts per package. If the parts have empty key fields and matching part values, **Annotate Schematic** assigns parts to the same package. If the parts have data in their key fields, **Annotate Schematic** assigns parts to the same package only if their key fields match. For this reason, if you use the **Annotate Schematic** key field, you should include the part value as part of the key field.

For more about key fields, see *Chapter 1: Configure Schematic Tools.*

Before annotation and after annotation

Figure 6-1 illustrates a worksheet that is not annotated. Figure 6-2 shows the same worksheet, after annotation.

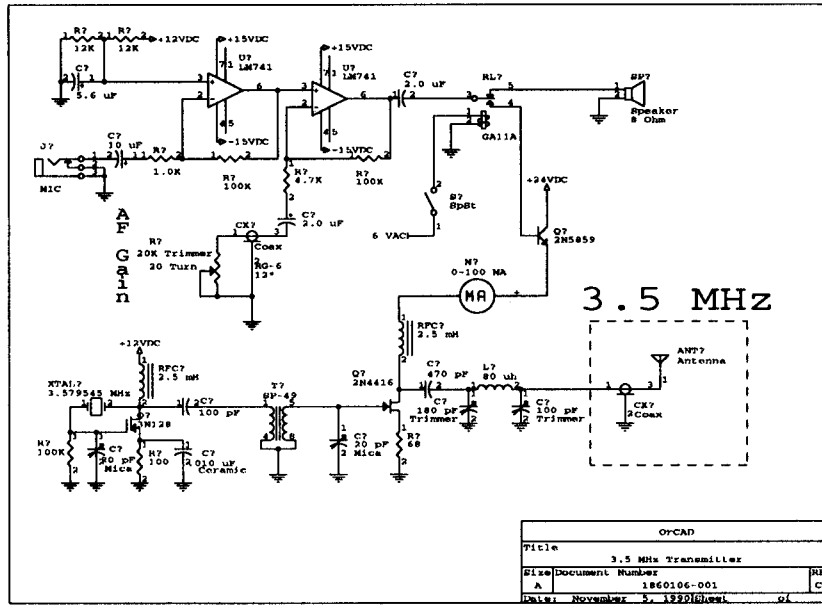


Figure 6-1. Worksheet before annotation.

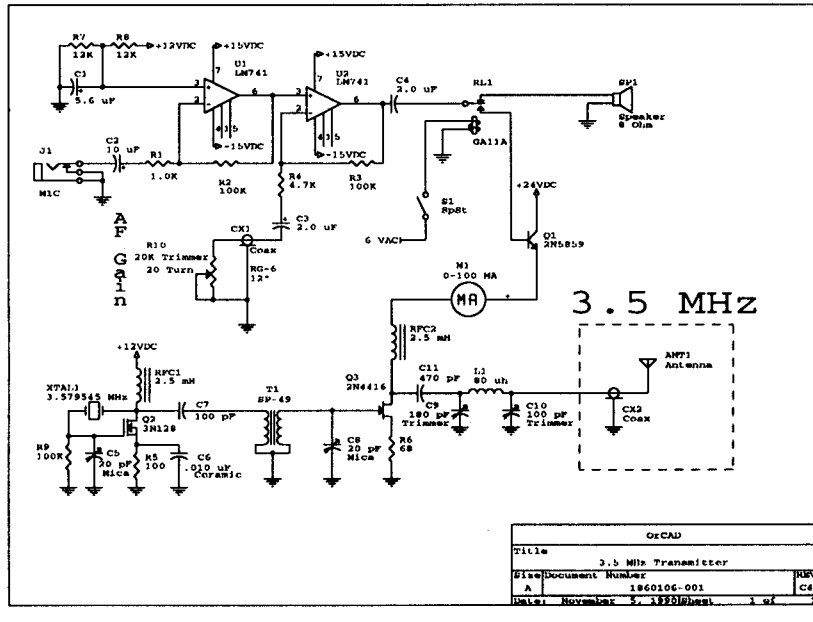


Figure 6-2. Annotated worksheet.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Annotate Schematic**. Select **Local Configuration** from the menu that displays.

Select **Configure ANNOTATE**. A configuration screen appears (figure 6-3).

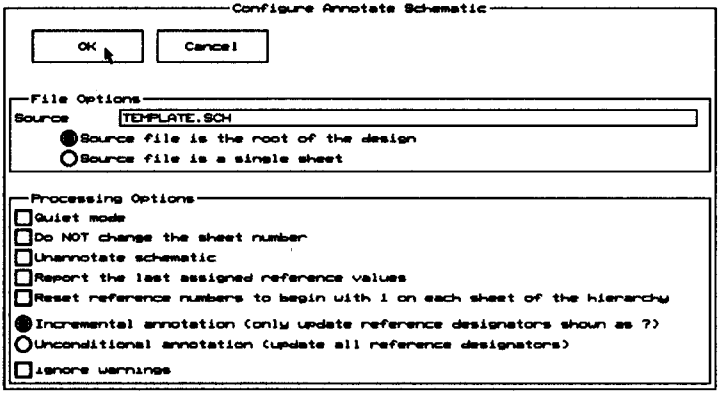


*Figure 6-3. **Annotate Schematic's** local configuration screen.*

File Options

File Options defines the source file and its type.

Source

The **Source** is the root of the design or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following options:

- `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a !Link, it is a flat design.

- `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Processing Options

You may select from the following options:

- ☐ `Quiet mode`

  Turns quiet mode on.

- ☐ `Do NOT change the sheet number`

  Causes the sheet number set in the title block to remain unchanged. If this option is not selected, **Annotate Schematic** renumbers all of the schematics in the design.

- ☐ `Unannotate schematic`

  Resets all reference designators in the design. When parts were first placed, the reference designator had a question mark (?) as the numeric component. This option resets all numbers to "?". If this option is selected, the next two options are unavailable.

- ☐ `Report the last assigned reference values`

  Tells **Annotate Schematic** to report the last reference designator assigned to a design, after it is done annotating. The report is placed in a file with the same name as the source, ending with the extension .END. This option is unavailable if **Unannotate schematic** is selected.

- ☐ `Reset reference numbers to begin with 1 on each sheet of the hierarchy`

  Restarts all reference designators used at 1 for each sheet of the design instead of only on the first sheet. Use this option to annotate a complex hierarchy. If you are using OrCAD's **Digital Simulation Tools** and your design is a complex hierarchy, this option is recommended. This option is unavailable if **Unannotate schematic** is selected.

Select one of the following options:

- ○ `Incremental annotation (only update reference designators shown as ?)`

  Updates only the reference designators that have a question mark (?) as the numeric component. When new parts are placed on a sheet, the reference is not assigned a numeric value, but rather is given the unassigned designation (?). If you do not wish to renumber all reference designators, including all previously set references, use this option.

- ○ `Unconditional annotation (update all reference designators)`

  Updates all reference designators in the order in which they are placed in the worksheet. All references are updated, even those that may have been assigned previously.

If desired, select this option:

- ☐ `Ignore warnings`

  Prevents **Annotate Schematic** from displaying a message when it encounters warnings.

△ *NOTE: If you assign reference designators manually, you can use **Check Design Integrity** to check for duplicates and mismatched types (such as parts with different values assigned to a single package).*



Back Annotate

Execution

Back Annotate scans a design or a single sheet and updates part reference designators. To run **Back Annotate**, you must provide a text file that lists old and new reference designators. This file is called a "Was/Is" file.

Was/Is file format

A Was/Is file is a text file containing the old and new reference designators. You create it using a text editor.

A Was/Is entry begins with the old reference designator that you want to modify, followed by any number of space, tab, or new line characters, which is in turn followed by the new reference designator value. Make a Was/Is entry for each reference designator you want to change. A new line is not required after each entry.

The following is an example of a typical Was/Is file:

```
R1    R5
R2    R12
R3    R6
C5    C1
C12   C2
U5C   U1A
U3B   U3A
```

In the above example, the occurrence of R1 in the design is changed to R5, R2 becomes R12, and so on.

Running Back Annotate

With the **Schematic Design Tools** screen displayed, select **Back Annotate**. Select **Execute** from the menu that displays.

When the reference designators are changed, the **Schematic Design Tools** screen displays.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Back Annotate.** Select **Local Configuration** from the menu that displays.

Select **Configure BACKANNO.** A configuration screen displays (figure 7-1).

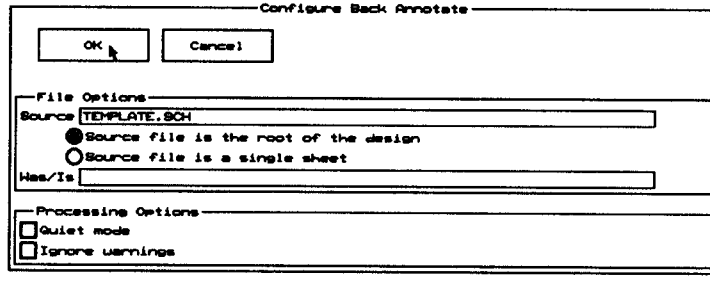


*Figure 7-1. **Back Annotate's** local configuration screen.*

File Options

File Options defines the source file and its type, and the Was/Is file.

Source

The **Source** is the root of the design or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following options:

- O `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a !Link, it is a flat design.

- O `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Was/Is

Was/Is specifies the name of the text file containing the old and new reference designator pairs. It may have any valid path and name. The format of this file is discussed at the beginning of this chapter.

Processing Options

You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Ignore warnings`

  Prevents **Back Annotate** from displaying a message when it encounters warnings.



Cleanup Schematic

Cleanup Schematic scans a file and checks for wires, buses, junctions, labels, module ports, and other objects that are placed on top of each other. It can scan an entire design or a single sheet.

Cleanup Schematic removes duplicate or overlapping wires, buses, and junctions, and displays warning messages advising you of other duplicate objects. It does not check for objects overlapping part leads, wires overlapping buses, or wire-width bus entries overlapping bus-width bus entries.

Use **Cleanup Schematic** whenever you want to check for and correct drawing errors in the worksheet. Check all worksheets with **Cleanup Schematic** to reduce errors and warnings that may occur when you use other utilities.

Execution

With the **Schematic Design Tools** screen displayed, select **Cleanup Schematic**. Select **Execute** from the menu that displays.

When **Cleanup Schematic** finishes, the **Schematic Design Tools** screen displays.

△ *NOTE: Cleanup Schematic also runs as part of Check Design Integrity.*

When **Cleanup Schematic** processes a very large worksheet, the tool may display the message: "CLEANUP will need to be repeated for this file." This means there was not enough memory to complete the cleanup process in one pass. If this occurs, run **Cleanup Schematic** again.

Cleanup Schematic renames the original schematic files (the files as they existed before **Cleanup Schematic** was run) with a .BAK extension. It saves the changed schematic files using the filenames of the original files.

If your disk becomes full during the cleanup operation, you will always have your original design intact. The original design is preserved because **Cleanup Schematic** does not rename the original file until after it has been successfully processed. After the process is complete, the original file is renamed to a .BAK extension and the processed version of the file is given the original name.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Cleanup Schematic**. Select **Local Configuration** from the menu that displays.

Select **Configure CLEANUP**. **Cleanup Schematic**'s local configuration screen displays (figure 8-1).

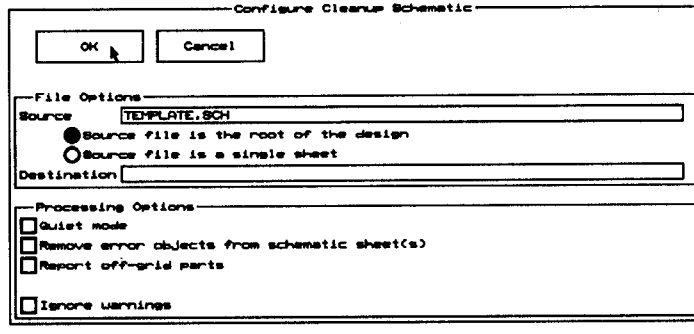


Figure 8-1. Cleanup Schematic's local configuration screen.

File Options

File Options defines the source file and its type, and the destination file.

Source

The **Source** is the root of the design or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following options:

- `Source file is the root of a hierarchy`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a |Link, it is a flat design.

- `Treat source file as a one-sheet file`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Destination

The **Destination** is any valid pathname where the report output of **Cleanup Schematic** is placed. This entry is optional.

Processing Options You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Remove error objects from schematic sheet(s)`

  Causes **Cleanup Schematic** to remove error markers from each worksheet in the design.

- ❑ `Report off-grid parts`

  Causes **Cleanup Schematic** to check the entire worksheet for parts that are placed off grid and report them.

- ❑ `Ignore warnings`

  Prevents **Cleanup Schematic** from displaying a message when it encounters warnings.



Check Design Integrity

Check Design Integrity prepares a design for processing by **Create Netlist** or **Create Hierarchical Netlist**, or for transfer to **PC Board Layout Tools 386+** using the **To Layout** tool. **Check Design Integrity** runs the following processes:

- **Cleanup Schematic**, which checks for drawing errors, such as overlapping wires, fixes the ones it can fix, and reports any others.
- **Cross Reference Parts**, which checks the reference designators and reports duplicates and type mismatches.
- **Check Electrical Rules**, which checks for electrical rules violations.

Execution

With the **Schematic Design Tools** screen displayed, select **Check Design Integrity**. Select **Execute** from the menu that displays.

When the process is complete, the **Schematic Design Tools** screen displays.

Local Configuration

Because CLEANUP, CROSSREF, and ERC are each configured individually, **Check Design Integrity** has three configuration screens. To configure **Check Design Integrity**, select **Check Design Integrity**, and then select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose the **Check Design Integrity** process to configure. You can also use it turn processes on or off. When you run **Check Design Integrity**, only the processes that are turned on run. For most cases, you must have CLEANUP, CROSSREF, and ERC all turned on.

```
Configure CLEANUP
Configure CROSSREF
Configure ERC
CLEANUP   on
CROSSREF  on
ERC       on
```

To turn a process on or off, choose the desired process from the menu. For example, to turn ERC off, select **ERC on** from the menu. This prompt displays:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **off** to turn the process off.

Configure CLEANUP, CROSSREF, and ERC

See *Chapter 8: Cleanup Schematic*, *Chapter 26: Cross Reference Parts*, and *Chapter 25: Check Electrical Rules* for information on configuring these tools.

△ *NOTE: You may need to run **Check Design Integrity** several times as you correct the conditions causing warnings and errors. Run **Check Design Integrity** until it issues no warnings or errors, or until the warnings are acceptable.*



Creating a netlist

To exchange information with tools created by third-party vendors with their own proprietary input formats, **Schematic Design Tools 386+** can create netlists in a variety of industry-accepted formats.

Incremental design

OrCAD's design database is incremental. The incremental approach speeds up the design process, especially during revision, verification, and maintenance cycles for hierarchical designs.

The incremental netlist process consists of three steps:

- Compile
- Link (when necessary)
- Format (when necessary)

Incremental netlisting resembles the compile and link process used to create executable computer programs from a source language (such as C or Pascal). As in the process of creating a computer program, the schematic (source) files are compiled into an intermediate form, and are then linked to produce a file that contains all of the connectivity information from all of the schematic files. If you are using a third-party application that requires a netlist as input, the connectivity information is then formatted into a netlist in one of over 30 formats.

Compile: INET

First, the schematic files are compiled by INET into an intermediate form called the *incremental connectivity database.*

The first time it runs, INET compiles all the sheets in a design. From then on, it compiles only sheets that have changed since the last time it ran. When INET runs, it compares the time stamp for a sheet against the one for the incremental connectivity database belonging to that sheet. If the sheet's time stamp is more recent than the incremental connectivity database, INET recompiles the sheet.

Link: ILINK

Then, the increments are linked together by ILINK to make a file that contains all the connectivity information from all of the schematic files. The result is the *linked connectivity database.*

Format: IFORM or HFORM

Finally, the connectivity database is translated and formatted for use by tools that cannot read the design database directly. There are two such tools: IFORM makes flat format netlists, and HFORM makes hierarchical format netlists (EDIF 2 0 0 and SPICE).

△ *NOTE: You can create your own netlist formats written in the C programming language. See* Appendix D: Creating a custom netlist format *for details.*

Creating a netlist

Figure 10-1 on the next page shows the process flow for creating a netlist. All netlists originate with the compile process, INET. After compilation, the incremental connectivity database can be linked to produce flat or hierarchical netlists for use by other tools.

OrCAD's **Digital Simulation Tools** uses the incremental connectivity database directly for input. It does not use the formatting process. OrCAD's **PC Board Layout Tools** uses a netlist in the flat EDIF format.

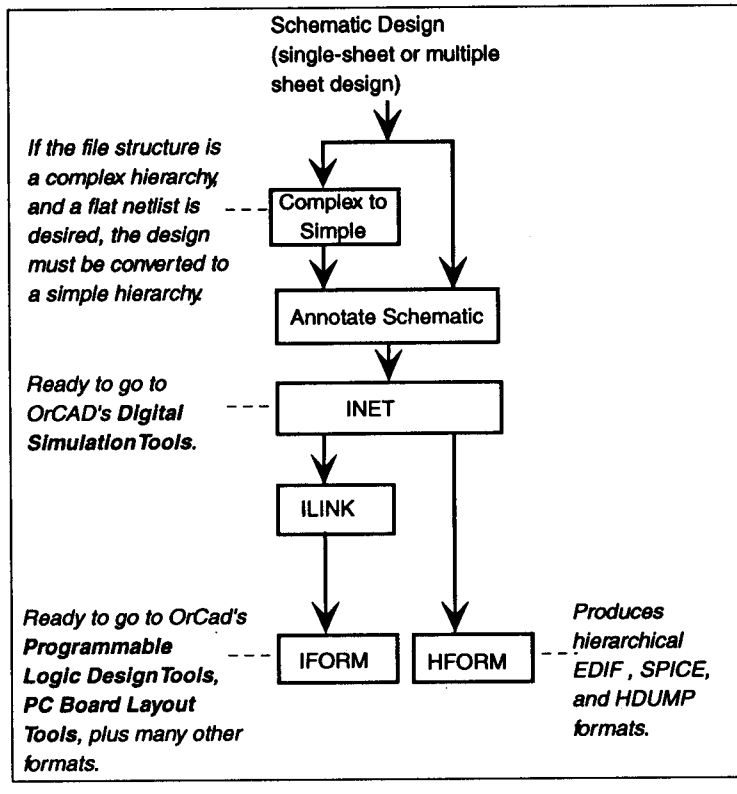


Figure 10-1. The process flow for creating a netlist.

△ *NOTE: If the file structure is a complex hierarchy, it must be converted into a simple hierarchy using* ***Complex to Simple*** *before it can be flattened and annotated with unique references.* ***Complex to Simple*** *is a function provided on the* ***Design Management Tools*** *screen. This tool creates a new design with all sheets referenced only once. In this new design, a flattened netlist is then created. The* ESP Design Environment User's Guide *explains this process.*

The compiler: INET

The first step in creating either flat (one-sheet, or many sheets) or hierarchical (complex or simple) netlists is to create the incremental connectivity database.

The incremental connectivity database

INET creates the incremental connectivity database. It consists of an .INF file for each sheet in the design and one .INX file for the entire design, as shown in figure 10-2.

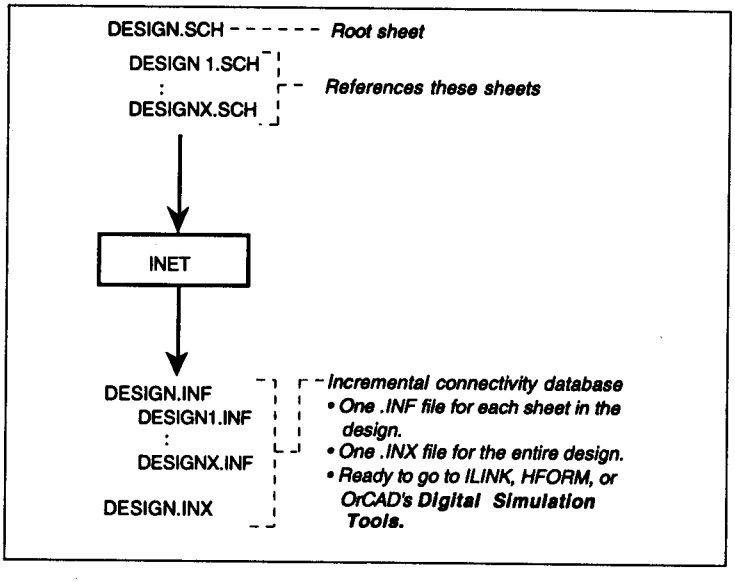


Figure 10-2. INET produces the incremental connectivity database.

The .INF file

Compiling the root sheet of the design creates a compact database of the connectivity information for each sheet referenced by the design, which is stored in an .INF file for each sheet. This representation includes all the devices on the sheet, the connectivity between parts, the pipe commands on the sheet, and information entered in the title block.

You don't need to keep track of which sheets you've changed and therefore have to be recompiled. INET does this for you. When INET runs, it compares the time stamp (that is, the date and time) for each sheet against the time stamp of any pre-existing .INF file for that sheet. If the sheet is more recent than its .INF file, then INET recompiles that sheet.

▲ *CAUTION: The proper functioning of INET depends on the clock in your computer being set properly. If your clock must be reinitialized each time your computer is started, please take the time to set it properly.*

The .INX file

While processing the file structure, INET constructs an .INX file from the sheet name you give it. Each sheet in the file structure is listed in the .INX file, and any sheets that were recompiled are marked with an asterisk. INET uses the .INX file to speed processing and organize the database. You can use **Edit File** to view the .INX file and see the files in the design, *but do not modify the .INX file in any way.*

Creating a multiple-sheet design using the | LINK command

How does INET know what files belong in a file structure?

For a hierarchical file structure, the answer is simple. When a sheet is added to the hierarchy, a name is created and stored with the sheet. Thus, the process of compiling a hierarchy becomes an iterative process of compiling a sheet, then compiling all of the sheets referenced on the sheet just compiled.

Designs that are flat, however, do not have sheet symbols to refer to the other sheets in the design. Rather, a different mechanism is used: the | LINK command. On the root sheet of the design, a series of text lines must be placed in the following format:

```
| LINK
| SHEET2
| SHEET3
```

Notice that the vertical symbols are vertically aligned, the word "LINK" is uppercase, and that the sheet names listed are the filenames. If the filename includes an extension, the extension must appear in the command.

This tells INET that the root sheet, *SHEET2*, and *SHEET3* are all part of the design. If new sheets are added to the design, simply add new text entries to the root sheet in the same vertical column as the other | LINK text. The order of the sheet names from top to bottom is the order in which all of the processors, reporters, and transfers process the design.

Caveats

As previously mentioned, INET is incremental. It compiles one sheet at a time. This can lead to apparent anomalies.

For instance, say you have a multisheet file structure, and INET reports that two sheets have errors. If you fix the errors in one of those sheets, then rebuild the netlist, no errors show up. There should still be errors in the other sheet. Why didn't INET report them?

Because the second sheet was not altered, it was not recompiled. If you check the .INX file, you will see that the sheet you fixed was recompiled (it has an asterisk next to it), but the other sheet was not. For this reason, when you are making a final netlist, if you are uncertain whether you have corrected all of the errors in the design, select **Rebuild file stack** and **Unconditionally process all sheets in design** on the **Configure Incremental Netlist** screen to force INET to compile *all* files.

The linker: ILINK

ILINK creates the intermediate netlist structure. This consists of the .INS, .RES, and .PIP files. These files are used by IFORM.

Figure 10-3 shows this process.

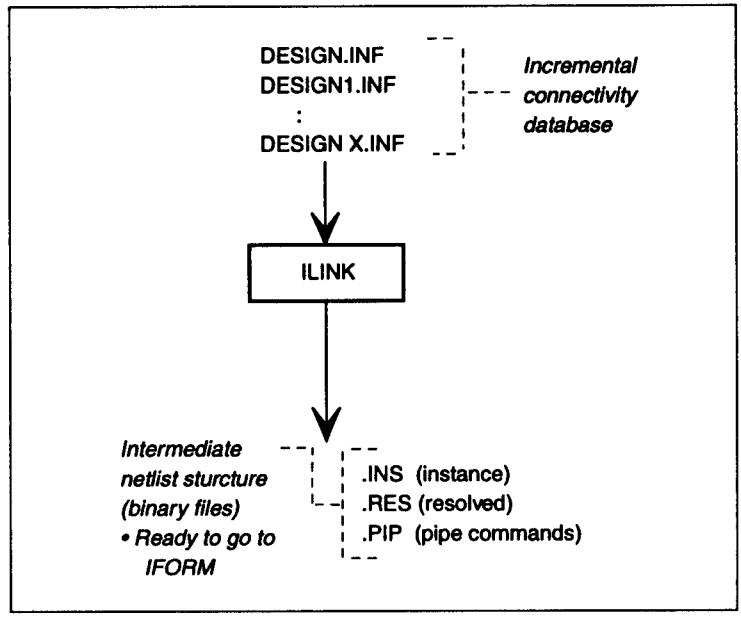


Figure 10-3. ILINK produces the intermediate netlist structure.

△ *NOTE: ILINK is not used for hierarchical netlist formats. See* The hierarchical formatter: HFORM *in this chapter.*

It is not always necessary to use ILINK. If tool sets are designed to access the incremental connectivity database directly, formatting is not needed. Some tool sets (such as **Digital Simulation Tools**) are designed to access the database in its unlinked form.

Intermediate netlist structure

The intermediate netlist structure consists of three files: the .INS and .RES files are binary, the .PIP file is text. These are the files used by IFORM.

The .INS file

One of the files produced by ILINK is the instance (.INS) file. This file contains information on all the parts in all the .INF files.

The .RES file

Another file produced by ILINK is the resolved (.RES) file. This file contains information about the connectivity of parts in all the .INF files in a design.

The .PIP file

The .PIP file is only produced when pipe commands are present on a root sheet. Several netlist formats allow pipe commands to be present on the schematic and written to the netlist. One such pipe command is the !SPICE command used by the SPICE netlist formats.

The flat formatter: IFORM

If you are transferring design information to a third-party EDA tool, you will probably need to format ILINK's output to make it readable by your destination tool.

The last step formats the .INS, .RES, and .PIP files into any of over thirty different netlist formats. These formats are discussed in *Appendix B: Netlist Formats. Chapter 10: Create Netlist* explains how to create a netlist in one of these formats. Figure 10-4 shows the IFORM process.

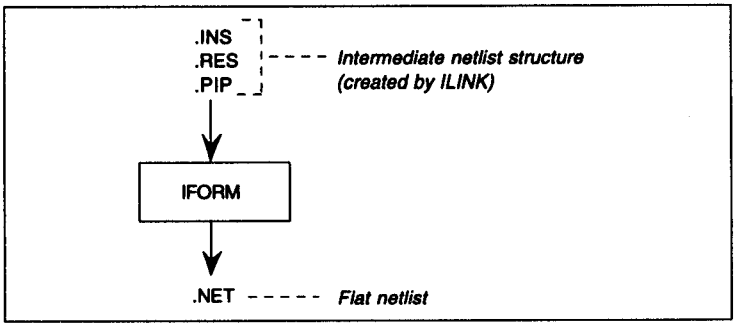


Figure 10-4. IFORM produces a flattened netlist.

Unnamed nets

Each time you run IFORM, it creates netnames for any nets not specifically named on the schematic. These system-generated netnames meet all the requirements for the selected netlist format.

With some netlist formats, you can tell IFORM to retain the netnames it generates for unnamed nets. This makes it easier to bring design modifications (forward ECOs) into an existing layout. See the descriptions of the flat EDIF, PADS 2000, and PADS-PCB formats in *Appendix B: Netlist formats* for information on the **Keep system generated netnames** option.

The hierarchical formatter: HFORM

Creating a hierarchical netlist is similar to creating a flat netlist except that ILINK is not used. HFORM reads the .INF files directly. These formats are discussed in *Appendix B: Netlist formats. Chapter 11: Create Hierarchical Netlist* explains how to create a netlist in one of these formats. Figure 10-5 shows the HFORM process.

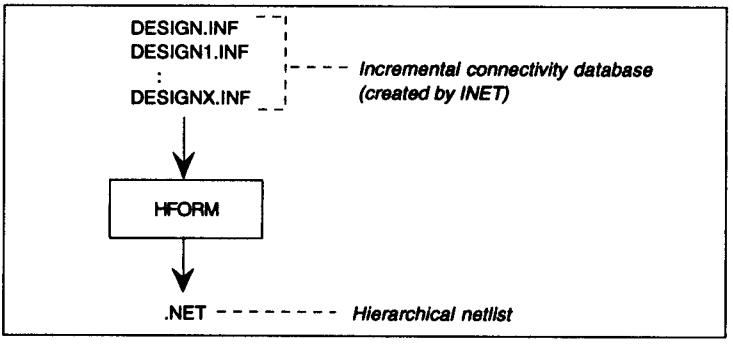


Figure 10-5. HFORM produces a hierarchical netlist.



Create Netlist

There are two basic types of netlist formats:

- A linked format where all ports and signals have been resolved across the entire design (the design is totally flat at this point). **Create Netlist**, discussed in this chapter, creates this type of netlist.
- A hierarchical format where all sheets and subsheets remain intact and are used to reference subnets. **Create Hierarchical Netlist**, discussed in *Chapter 11: Create Hierarchical Netlist*, creates this type of netlist.

These netlists are intended primarily for use in inter-facing to tools outside the ESP design environment. If you are using OrCAD's **Digital Simulation Tools** or **PC Board Layout Tools**, the connectivity database is managed using the transfer buttons to those tools. A netlist is not produced when transferring to **PC Board Layout Tools** or **Digital Simulation Tools**.

Linked format

The linked form reproduces the sheets giving all parts unique references and, hence, all nodes unique names. This view is the "actual" view of the design since each part is unique. However, a linked view of a design removes any evidence of the structure of the design as well as any evidence of design reuse. The design may be either a simple hierarchy or a flat file structure.

Creating linked and flattened netlists

Create Netlist creates linked and flattened netlists in three steps. They are:

1. Process the design with INET to produce the incremental connectivity data-base for the design. INET updates the incremental connectivity database efficiently by updating the database only for those sheets that have changed.
2. Next, link the incremental connectivity database into a single database that can be used by the formatting process. This is done by ILINK which produces either intermediate netlist structure files that require formatting or the linked connectivity database that is ready to go to OrCAD's **PC Board Layout Tools.**
3. Finally, format the intermediate netlist structure with IFORM to produce the final flattened netlist in one of over thirty netlist formats (Wirelist, PCAD, etc.).

Example If the design is called *root* and the format is contained in *my_format,* netlist creation is as follows:

Processor	*Input file(s)*	*Output file(s)*
1. INET	*root.sch*	*root.inf*
2. ILINK	*root.inf*	*root.res* *root.ins* *root.pip*
3. IFORM	*my_format* *root.res* *root.ins* *root.pip* (if needed)	The final netlist

IFORM is actually an interpreter that uses a format specification file (*my_format* in item 3 above) to produce the final netlist. You can write your own netlist format specification file if you like (see *Appendix D: Creating custom netlist formats*).

Execution

Create Netlist reads a design and creates a flat netlist.

△ *NOTE: If the source design is a complex hierarchy and you need a flat netlist, you must first change the design into a simple hierarchy using* ***Complex to Simple*** *on the* ***Design Management Tools*** *screen. To create a hierarchical netlist of a complex hierarchy, see* Chapter 12: Create Hierarchical Netlist.

Running Create Netlist

With the **Schematic Design Tools** screen displayed, select **Create Netlist**. Select **Execute** from the menu that displays.

When the netlisting process is complete, the **Schematic Design Tools** screen displays again.

Local Configuration of Create Netlist

Since INET, ILINK, and IFORM are each configured individually, **Create Netlist** has three configuration screens. To configure **Create Netlist**, select **Create Netlist**. Select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose the **Create Netlist** process to configure. You can also use it turn processes on or off. When you run **Create Netlist**, only the processes that are turned on run. For most cases, you must have INET, ILINK, and IFORM all turned on.

```
Configure INET
Configure ILINK
Configure IFORM
INET  on
ILINK on
IFORM on
```

To turn a process on or off, choose the desired process from the menu. For example, to turn IFORM off, select **IFORM on** from the menu. ESP prompts:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **off** to turn the process off.

➤ *Helpful hint . . .*

If you need to change only the filename for **Create Netlist**, use **Design Management Tools** to select the file as the current root sheet. This saves you from having to set the filename in three separate local configuration screens.

△ *NOTE: If you are creating a netlist for **Digital Simulation Tools** or **PC Board Layout Tools**, use the transfer button **To Digital Simulation** or **To Layout**, because they are already properly set up to perform the appropriate netlist processes when you transfer to the respective tool.*

Configure INET

With the **Schematic Design Tools** screen displayed, select **Create Netlist**. Select **Local Configuration** from the menu that displays, and then select **Configure INET**. INET's local configuration screen displays (figure 11-1).

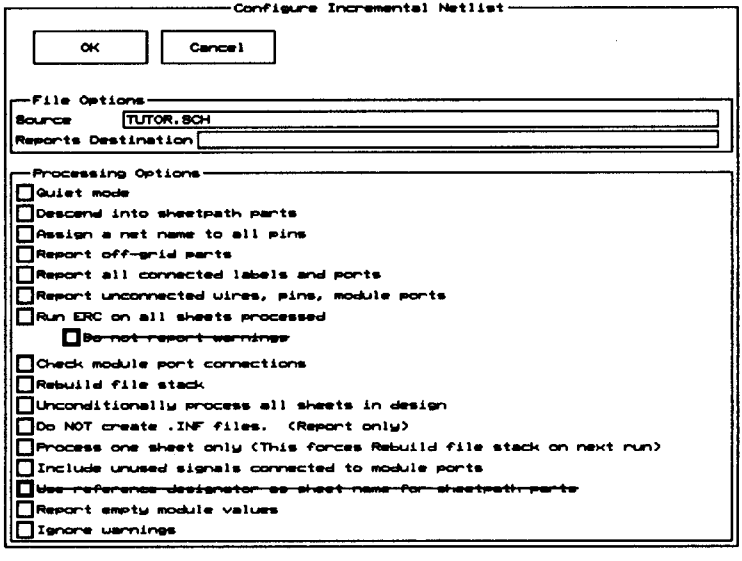


Figure 11-1. INET's local configuration screen.

File Options

File Options defines the source file from which the incremental database files are created. It also defines the name of a file to contain data created by INET.

Source

The **Source** is the root of the design or the filename of a single sheet. It may have any valid pathname. One .INF file is created for each sheet referenced by the root sheet.

△ *NOTE: You can also set the filename using Design Management Tools to specify a current root sheet.*

Reports Destination

The **Reports Destination** is the name of a file where the report is to be placed. This specification is optional. If a **Reports Destination** is not specified, the report is sent to the screen and the file #ESP_OUT.TXT.

Processing Options

You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Descend into sheetpath parts`

  Tells INET to descend into parts defined as sheetpath parts. That is, it treats a sheetpath part as a sheet. This option, while designed for complex hierarchies, may be used in simple hierarchies. It is recommended only for use with FPGA, ASIC, and other designs that require a complex hierarchy.

- ❑ `Assign a net name to all pins`

  Tells INET to assign a net name to all pins, including unconnected ones.

- ❑ `Report off-grid parts`

  Tells INET to check the worksheet for parts, sheets, labels, module ports, and power objects that are off-grid. This information appears in the **Reports Destination** file.

- ❑ `Report all connected labels and parts`

  Tells INET to report all connected labels and module ports. This information appears in the **Reports Destination** file.

- ❑ `Report unconnected wires, pins, module ports`

  Tells INET to include a list of all unconnected wires, pins, and module ports. This information appears in the **Reports Destination** file.

  Also tells INET to use the maximum possible number of points on pins to determine whether the pins are electrically connected.

  This option slows INET's processing.

➤ *Helpful hint . . .*

This option describes overlapping wires and pins, in the .INF file only, as if they were connected. It is important to note that INET *does not* modify the schematic to correct the overlap.

- ❑ `Run ERC on all sheets processed`

  Runs an electrical rules check on all sheets that INET processes. This is the same process provided by the **Check Electrical Rules** tool. The information appears in the **Reports Destination** file.

  - ❑ `Do not report warnings`

    This option is available if you select the **Run ERC on all sheets processed** option. It tells INET to not test some of the conditions for which it normally issues warnings. The conditions are:

    - ❖ Two power objects connected
    - ❖ Single node nets
    - ❖ Input signals without a driving source

    These conditions are always checked—unless you select the **Do not report warnings** option—and cannot be changed with the **Check Electrical Rules Matrix.**

    Use this option with caution. You may end up with a netlist containing conditions that are not acceptable.

- `Check module port connections`

  Tells INET to check that each sheet net has the same name as the underlying module port after tests and processing are completed.

△ ***NOTE:*** *This option does not apply to names that contain overbar characters.*

- `Rebuild file stack`

  Tells INET to rebuild the file stack that is used to determine the sheets that are in the incremental connectivity database. This file stack speeds the processing of the database when incremental updates occur. It may be viewed, but should not be modified. The file stack is given a .INX extension.

- `Unconditionally process all sheets in design`

  Tells INET to ignore the incremental aspect of normal connectivity database processing. All sheets in the design are recompiled, even if the current database is up to date.

- `Do NOT create .INF files (report only)`

  When processing the design, you may wish to leave the incremental connectivity database unchanged and only review the reports. This option causes all checks to be run and reports to be created, but does not update the incremental connectivity database.

- `Process one sheet only (This forces Rebuild file stack on the next run)`

  Tells INET to produce an incremental connectivity database for a single sheet in a design. This option is useful for troubleshooting netlist problems. The next time INET runs, it will also rebuild the file stack used to determine the sheets that are in an incremental connectivity database.

  Once you select this option, you disrupt the incremental process of INET. If you select this option and then later wish to create a netlist of the entire design, you must select the **Unconditionally process all sheets in design** option the next time you run INET.

- `Include unused signals connected to module ports`

  Tells INET to keep module ports that do not match a signal name on another schematic and to maintain those module ports all through the netlist process. When this option is not selected, INET discards module ports which do not connect to a signal.

  This option is useful for passing buses through a hierarchy, because some signals may not be used at every level of the hierarchy.

- `Use reference designator as sheet name for sheetpath parts`

  Tells INET to use the reference designator alone, rather than a combination of the reference designator and the part value, as the sheet name in the .INF file. This option can only be used in conjunction with the **Descend into sheetpath parts** option.

  This option is useful for creating a simulation or an FPGA netlist.

- `Report empty module values`

  Tells INET to generate a warning message if the **Module Value** field (as configured in the key fields) is empty.

- `Ignore warnings`

  Prevents INET from displaying a message when it encounters warnings.

  If your schematic contains several sheet path parts with the same part value and you don't have this option selected, INET cannot create the .INF files.

Configure ILINK

With the **Schematic Design Tools** screen displayed, select **Create Netlist**. Select **Local Configuration** from the menu that displays. Select **Configure ILINK**. A configuration screen displays (figure 11-2).

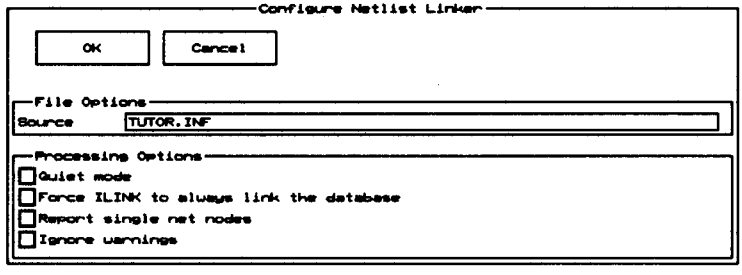


Figure 11-2. ILINK's local configuration screen.

File Options **File Options** defines the source file.

Source The **Source** is the incremental connectivity database root file. There *must* be an extension on the filename. The source file is the output from the INET process discussed previously and should have the recommended extension of .INF. It may have any valid pathname.

△ ***NOTE:*** *You can also set the filename using* ***Design Management Tools*** *to specify a current root sheet.*

Processing Options Select any combination of the following:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Force ILINK to always link the database`

  Tells ILINK to force a link of the database to occur, even if the database is up to date.

- ❑ `Report single net nodes`

  `Tells ILINK to` include in the netlist all nets that have only a single pin. To assure that these nets are named, enable **Assign a net name to all pins** in INET.

- ❑ `Ignore warnings`

  Prevents ILINK from displaying a message when it encounters warnings.

Configure IFORM

With the **Schematic Design Tools** screen displayed, select **Create Netlist**. Select **Local Configuration** from the menu that displays. Select **Configure IFORM**. A configuration screen displays (figure 11-3).

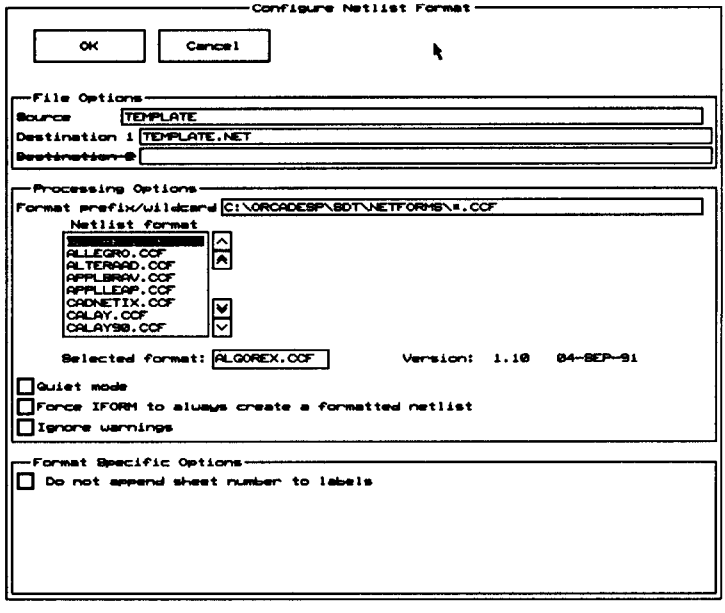


Figure 11-3. IFORM's local configuration screen.

File Options

File Options defines the source and destination files.

Source

The **Source** is the intermediate netlist structure from the ILINK process discussed previously and should *not* have any extensions. It may have any valid DOS pathname.

△ *NOTE: You can also set the filename using Design Management Tools to specify a current root sheet.*

Destination 1 and Destination 2

These are the destination files for the formatted netlist files created by IFORM. All of the netlist formats use the first destination file for the netlist. A few of the netlist formats also create a component or part file. This file, if created, is placed in the second destination. If a filename is not provided where one is needed, its data is sent to the screen.

Processing Options

The **Format prefix/wildcard** entry box is the path from which formatting files are displayed in the **Netlist format** list box. Additionally, it provides a filter to selectively display files from the given directory. The filter defaults to F*.EXE, which displays only valid flat netlist format files.

Select a netlist format file by clicking its name in the list box or by entering its name in the **Selected Format** entry box. If the netlist format file name is typed into the **Selected Format** entry box, the path in the **Format prefix/wildcard** entry box must be valid.

If IFORM cannot find the filename you enter in the **Format prefix/wildcard** entry box, it leaves the **Selected format:** entry box empty.

The **Netlist format** list box lists as many as eighty files. If you enter a **Format prefix/wildcard** that specifies more than eighty files, the files after the eightieth file do not display.

Select any of the following:

- `Quiet mode`

  Turns quiet mode on.

- `Force IFORM to always create a formatted netlist`

  Tells IFORM to force a format to be performed, even if the files created by INET or ILINK are not up to date.

- `Ignore warnings`

  Prevents IFORM from displaying a message when it encounters warnings.

Format Specific Options

Schematic Design Tools 386+ provides format files to create netlists in over thirty formats. When a format is selected from the **Netlist Format** list box, its formatting options are displayed in this area. *Appendix B: Netlist formats* explains each of the formats and any available options.

You may write your own netlist formats and have them appear in the **Netlist Format** list box along with the ones supplied by OrCAD. *Appendix D: Creating a custom netlist format* explains this process.



Create Hierarchical Netlist

There are two basic types of netlist formats:

- A hierarchical format where all sheets and subsheets remain intact and are used to reference subnets. **Create Hierarchical Netlist** creates this type of netlist and is discussed in this chapter.
- A linked format where all ports and signals have been resolved across the entire design (the design is totally flat at this point). **Create Netlist** creates this type of netlist and is described in *Chapter 11: Create Netlist.*

Hierarchical format

The hierarchy presents the design as it was first partitioned, replete with all subsheets (children) and non-unique references and nodes. This form is generally the way "top-down" designs are created and maintained. However, the design does not contain unique references nor node (signal) names (unless the "path" to the part or node is used). Hierarchies represent the designer's view rather than the "stuff the board" view. **Create Hierarchical Netlist** is used to produce hierarchically formatted netlists.

Create Hierarchical Netlist consists of two processes:

1. Process the design with INET to produce the incremental connectivity database for the design. INET updates the incremental connectivity database efficiently by updating the database only for those sheets that have changed.
2. Produce the final hierarchical netlist in the desired format (EDIF or SPICE) using HFORM.

Example

For example, if the design is called *root* and the format is contained in *my_format* the following processes are run:

1. INET *root.sch*
2. HFORM *root my_format*

Execution

Create Hierarchical Netlist reads a hierarchical file structure and creates a hierarchical netlist. First, it scans the file structure and creates an incremental connectivity database for the design. Then it formats the incremental connectivity database into a hierarchical (simple or complex) netlist. See *Chapter 9: Creating a netlist* for more information on the netlisting process.

Running Create Hierarchical Netlist

With the **Schematic Design Tools** screen displayed, select **Create Hierarchical Netlist**. Select **Execute** from the menu that displays.

When the netlisting process is complete, the **Schematic Design Tools** screen displays again.

Local Configuration of Create Hierarchical Netlist

Since INET and HFORM must each be configured individually, **Create Hierarchical Netlist** has two configuration screens. To configure **Create Hierarchical Netlist**, select **Create Hierarchical Netlist**. Select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose which **Create Hierarchical Netlist** process to configure. You can also use it to turn each process on or off. When you run **Create Hierarchical Netlist**, only the processes that are turned on run. For most cases, you must have both INET and HFORM turned on.

```
Select Configuration
Configure INET
Configure HFORM
INET     on
HFORM    on
```

To turn a process on or off, choose the desired process from the menu. For example, to turn HFORM off, select **HFORM on** from the menu. **Schematic Design Tools 386+** prompts:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **off** to turn the process off.

△ *NOTE: If you are creating an incremental connectivity database to use with **Digital Simulation Tools**, turn HFORM off. **To Digital Simulation** is set up in this manner so that when you transfer to **Digital Simulation Tools**, the appropriate netlist processes are performed.*

Configure INET

For information about configuring INET, see *Configure INET* in *Chapter 11: Create Netlist*.

Configure HFORM

HFORM functions exactly as IFORM, except it produces a hierarchical netlist. For information about HFORM, see *Configure IFORM* in *Chapter 11: Create Netlist*, replacing references to IFORM with HFORM.



Select Field View

Select Field View scans a design or a single sheet and changes the visible attribute of the specified field.

Execution

Select **Select Field View** from the **Schematic Design Tools** screen. Select **Execute** from the menu that displays.

When the specified field attributes are changed, the **Schematic Design Tools** screen displays.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Select Field View**. Select **Local Configuration** from the menu that displays.

Select **Configure FLDATTRB**. A configuration screen displays (figure 13-1).

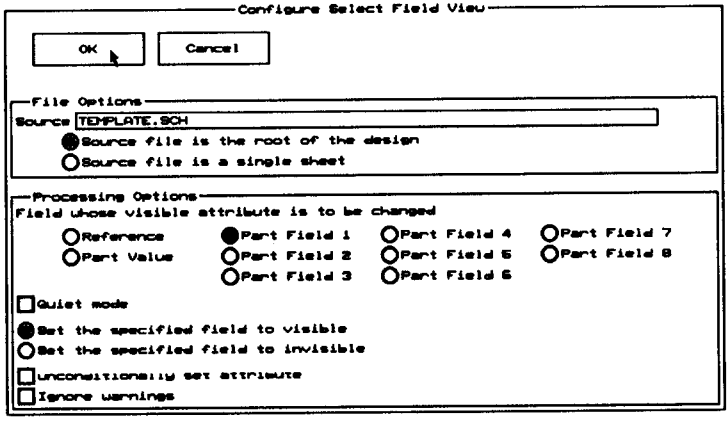


Figure 13-1. Select Field View's local configuration screen.

File Options **File Options** defines the source file and its type.

Source The **Source** is the file on which **Select Field View** operates. It may have any valid pathname.

After entering the source, select one of the following options:

O `Source file is the root of the design`

Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a |Link, it is a flat design.

O `Source file is a single sheet`

Specifies that the source file is a single worksheet and you want to process the single sheet only.

Processing Options Select one of the following options to define which visible attribute is to be changed:

O `Reference`	O `Part Field 4`
O `Part Value`	O `Part Field 5`
O `Part Field 1`	O `Part Field 6`
O `Part Field 2`	O `Part Field 7`
O `Part Field 3`	O `Part Field 8`

If desired, select the following option:

☐ `Quiet mode`

Turns quiet mode on.

Select an option to set the field's visibility attribute:

O `Set specified field to visible`

O `Set specified field to invisible`

Select either or both of the following:

- ❑ `Unconditionally set attribute`

  All parts attributes are affected when this option is selected, regardless of the current contents of the field. When this option is not selected, only those parts with information in the field are affected.

- ❑ `Ignore warnings`

  Prevents **Set Field View** from displaying a message when it encounters warnings.



Update Field Contents

Update Field Contents searches for and replaces text in part fields on schematics. Figure 14-1 shows the steps involved in using **Update Field Contents**. As shown in this figure, you must do three things to prepare:

- Specify which part field or part fields to match.
- Specify the field to update.
- Create an update file.

Once you complete these steps, you can run **Update Field Contents**. The next section describes the process in detail.

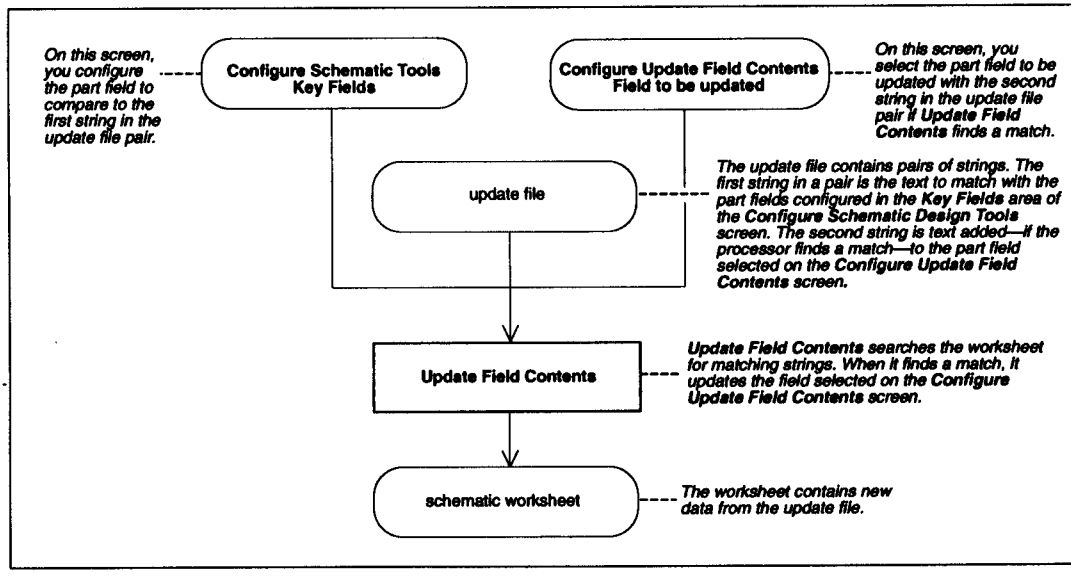


*Figure 14-1. The **Update Field Contents** process.*

Before you run Update Field Contents

Before you run **Update Field Contents**, be sure you:

- Specify which part fields to match by configuring **Key Fields**.
- Specify the field to update by configuring **Update Field Contents**.
- Specify text to search for and text to replace by creating an update file.

This section describes these steps in detail.

Configuring Key Fields

You must specify which part fields are compared with the match field in the update file. To do this, enter text in the **Update Field Contents** entry boxes in the **Key Fields** area of the **Configure Schematic Design Tools** screen. For information about what goes in these entry boxes, see *Chapter 1: Configure Schematic Design Tools.*

Configuring Update Field Contents

You must specify which part field is updated when **Update Field Contents** finds a string that matches the match field in the update file. To do this, select a field in the **Processing Options** area of the **Configure Update Field Contents** screen.

The field selected may or may not be one of the fields specified in the **Key Fields** entry boxes on the **Configure Schematic Design Tools** screen.

Creating an update file

Update Field Contents requires an update file. You create this text file using **Edit File.**

The update file is composed of a list of pairs of strings:

- The first string in each pair contains text **Update Field Contents** compares to part fields. These strings are match strings and can contain text for one or more part fields. You control what part fields are compared by entering the part fields in the **Update Field Contents** entry boxes in the **Key Fields** area of the **Configure Schematic Design Tools** screen.

- The second string in each pair contains text you want placed in a specific field if the match string matches a part field. These strings are called update strings.

Strings can contain up to 100 characters and spaces.

Strings are delimited with single or double quotation marks. The strings can be separated with any number of space, tab, or return characters.

To improve readability, it is a good idea to put each pair of match and update strings on a separate line, using tab characters between them to align the columns.

Figure 14-2 shows a typical update file with match strings in the left column and their corresponding update strings in the right column. For example, **74ls00** is a match string; its corresponding update string is **14DIP300.**

```
'74ls00'        '14DIP300'
'74ls138'       '16DIP300'
'74ls163'       '16DIP300'
'8259a'         '28DIP600'
```

Figure 14-2. Typical update file.

You can include single quotes in a string if you delimit the string with double quotes. For example:

```
"74ls00"        "test of Jack's edited part"
```

Points to remember about update files

When you are creating or editing update files, remember these points:

- Update files are limited in size only by the memory available to **Update Field Contents**. If an update file is too large, **Update Field Contents** does not run and suggests that you split your update file into smaller update files and then rerun **Update Field Contents** with each of the new update files.
- Case is not significant in match strings or update strings because **Update Field Contents** converts both to uppercase. You can, however, select the **Do not convert update string to uppercase** option to force case sensitivity on the match process. You can also select the **Do not convert key field match string to uppercase** option if you want to place lowercase text in a field.
- **Update Field Contents** checks update files for syntax and duplicate match strings. If it finds any errors, it reports them so you can identify and correct them.

Execution

Update Field Contents constructs a string from the key field designators for a field you select on the **Configure Update Field Contents** screen. Then, if that string equals a match string in the configured update file, it replaces the specified field with an update string.

Running Update Field Contents

With the **Schematic Design Tools** screen displayed, select **Update Field Contents**. Select **Execute** from the menu that displays.

When **Update Field Contents** is finished processing the design, the **Schematic Design Tools** screen displays.

During the updating process

When you run **Update Field Contents**, it gets **Key Fields** information from **Configure Schematic Design Tools** to determine which fields to match and **Field to be updated** information from **Configure Update Field Contents** to determine which field to update in the case of a match.

Then the tool scans the schematic. For each part, it converts the contents of the fields specified in the **Key Fields** entry boxes to a single string and compares that string to each update string in the update file.

If the string created from the schematic exactly matches one of the update strings, **Update Field Contents** copies the corresponding update string into the field specified on the **Configure Update Field Contents** screen and proceeds to the next part.

If the string created from the schematic does not match, **Update Field Contents** proceeds to the next part.

After the updating process

When complete, **Update Field Contents** creates these two or three files, depending on the configuration options selected:

- The updated schematic, *.SCH
- A back-up copy of the original schematic, *.BAK
- An optional update report

You can use **Draft** to inspect the updated schematic and **Edit File** to view the update report. If the results are not what you expected, rename *.BAK to *.SCH and try again.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Update Field Contents**. Select **Local Configuration** from the menu that displays.

Select **Configure FLDSTUFF**. A configuration screen displays (figure 14-3).

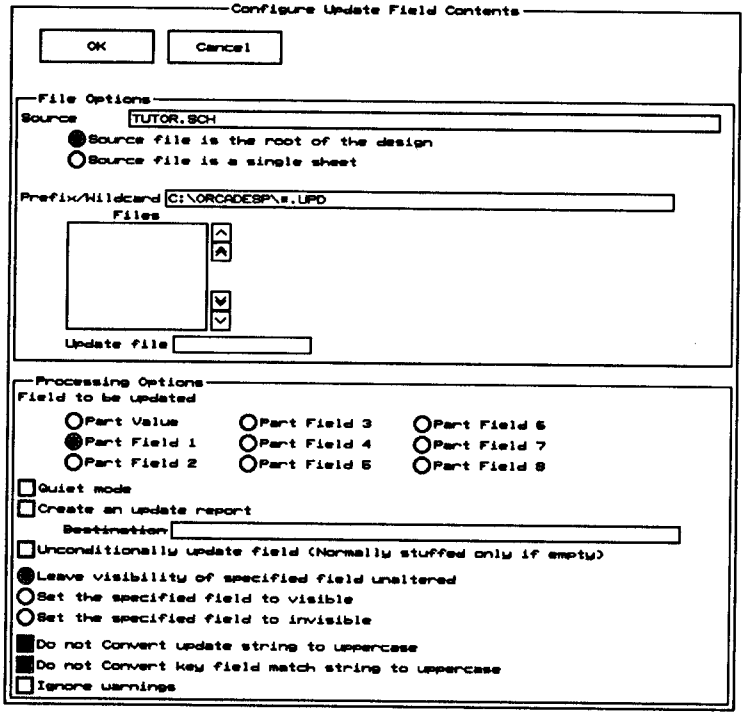


Figure 14-3. Local configuration screen for ***Update Field Contents.***

File Options

File Options defines the design in which to update part fields, and its type. It also defines the update file.

Source

This entry box contains the name of the design in which to update part fields.

Choose one of the following options to define the **Source's** file type:

O `Source file is the root of the design`

Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is a hierarchy. If it contains a !LINK command followed by a list of files, it is a flat design.

O `Source file is a single sheet`

Specifies that the source file is a single worksheet and you want to process the single sheet only.

Update file

The **Update file** is the name of a text file that you create. It is described at the beginning of this chapter.

Processing Options

Select one of the following options to define which field is to be updated:

O `Part Value`	O `Part Field 3`	O `Part Field 6`
O `Part Field 1`	O `Part Field 4`	O `Part Field 7`
O `Part Field 2`	O `Part Field 5`	O `Part Field 8`

△ ***NOTE: Part Field 8 is also called the Module Value.***

Select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Create an update report`

  Creates a report listing all the parts on each sheet and whether a field is updated for each part. If you select this option, specify a filename for the report in the **Destination** entry box.

- ❑ `Unconditionally update field (Normally stuffed only if empty)`

  Unconditionally changes the specified field. By default a field is only updated if it is empty. That is, fields with values already in them are not updated.

Select one of the following to control visibility of the updated field:

- O `Leave visibility of specified field unaltered`

  Causes **Update Field Contents** to leave the visibility of updated fields unchanged. However, if changed part fields were empty to begin with, they remain invisible.

- O `Set specified field to visible`

  Causes **Update Field Contents** to make all updated fields visible.

- O `Set specified field to invisible`

  Causes **Update Field Contents** to make all updated fields invisible.

Select any combination of these options:

- ❑ `Do not Convert update string to uppercase`

  Maintains the case of characters in the update string. The update file itself remains unchanged. Without this option, **Update Field Contents** converts the update string to all uppercase letters before placing the text in a field.

- ❑ `Do not Convert key field match string to uppercase`

  Causes **Update Field Contents** to be case sensitive when matching update file match strings with part fields on the schematic.

- ❑ `Ignore warnings`

  Prevents **Update Field Contents** from displaying a message when it encounters warnings.



Decompile Schematic

Decompile Schematic creates an ASCII text representation of a design. You typically use **Decompile Schematic** to create text files from worksheets, use a text editor to modify the text files, and then use **Compile Schematic** to create new worksheets.

Execution

With the **Schematic Design Tools** screen displayed, select **Decompile Schematic**. Select **Execute** from the menu that displays.

When the schematics are decompiled, the **Schematic Design Tools** screen displays.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Decompile Schematic**. Select **Local Configuration** from the menu that displays.

Select **Configure AEX32**. A configuration screen displays (figure 15-1).

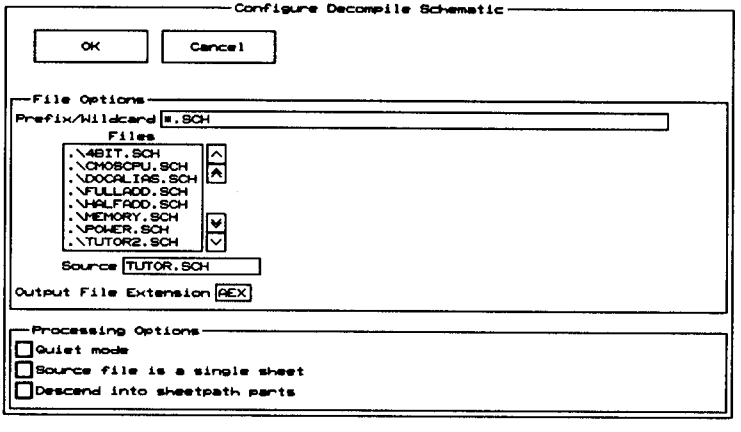


*Figure 15-1. **Decompile Schematic**'s local configuration screen.*

File Options

File Options defines the source file or files, and the output file extension.

Prefix/Wildcard

The **Prefix/Wildcard** defines the drive and directory containing the schematic (.SCH) files to use as source. It also serves as a "filter" of the files listed in the **Files** list box. It may have any valid pathname, including the standard DOS asterisk (*) and question mark (?) wildcards.

For example:

`Prefix/Wildcard` | `*.SCH`

Lists all files in the current design directory with a .SCH extension. This is the default.

`Prefix/Wildcard` | `..\MYDESN\*.SCH`

Lists all files in the MYDESN design directory with a .SCH extension.

`Prefix/Wildcard` | `???*.*`

Lists all files with at least three characters in the filename, and with any extension.

Source

The **Source** is any valid filename. Pick a filename from the **Files** list box, or enter a filename here.

Output File Extension

The **Output File Extension** is the extension that will be used for any ASCII text files created. Each output file has the same filename as the source file, with the specified extension.

Processing Options If desired, select any of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Source file is a single sheet`

  Use this option to prevent **Decompile Schematic** from processing any worksheets referred to in the source file.

  Without this option, **Decompile Schematic** creates a separate text file for each worksheet referred to in the linked or hierarchical design. Each text file has the same filename as the corresponding .SCH file, and has the specified **Output File Extension**.

- ❑ `Descend into sheetpath parts`

  Tells **Decompile Schematic** to descend into parts defined as sheetpath parts. That is, it treats a sheetpath part as a sheet. This option, while designed for complex hierarchies, may be used in simple hierarchies. It is recommended only for use with FPGA, ASIC, and other designs that require a complex hierarchy.

  Without this option, **Decompile Schematic** does not create a text file for sheetpath parts, regardless of the setting of the **Source file is a single sheet** option.

Output file format

Decompile Schematic creates an ASCII text representation of the source schematic and (optionally) any worksheets referred to in the source schematic. The output defines title block contents and any text, lines, parts, junctions, wires, and other objects on the source schematic.

Figure 15-2 shows an example of an ASCII text file created by **Decompile Schematic**.

```
{OrCAD SDT 32bit Schematic File}

HEADER(4.50, 1.80)
FRAME(3, 3, 'A', 0,
    '     June 30, 1994',
    '123456789-xxx-yyy',
    'B1',
    'POWER SUPPLY',
    'OrCAD',
    '',
    '',
    '',
    ''
)
TEXT(0.30, 1.90, 1, 'TO 110 VAC')
DLINE(4.70, 3.70, 8.20, 3.70)
DLINE(8.20, 3.70, 8.20, 5.60)
DLINE(8.20, 5.60, 4.70, 5.60)
DLINE(4.70, 5.60, 4.70, 3.70)
TEXT(4.90, 5.50, 2, 'BACKUP POWER CIRCUIT')
NOCONN(2.00, 1.60)
NOCONN(2.00, 2.00)
PART(2.10, 1.60, '0000000000000002B273DFEA', 'TRANSFORMER', NORMAL,
    REF(VISIBLE, 0.00, 0.00, 'T1'),
    VALUE(VISIBLE, 0.00, 0.00, '4:1'),
    DEVICE(1, 0)
)
PART(3.60, 1.50, '0000000000000002B273DFEF', 'BRIDGE', NORMAL,
    REF(VISIBLE, -0.47, -0.03, 'D1'),
    VALUE(VISIBLE, -0.59, 0.13, 'CR0127'),
    DEVICE(1, 0)
)
PART(4.40, 1.90, '0000000000000002B273DFF4', 'CAPACITOR POL', NORMAL,
    REF(VISIBLE, 0.00, 0.00, 'C4'),
    VALUE(VISIBLE, 0.00, 0.00, '470 uF'),
    DEVICE(1, 0)
)
```

Figure 15-2. Example Decompile Schematic output file (continues).

```
PART(5.20, 1.70, '0000000000000002B273DFF9', 'RESISTOR', UP,
    REF(VISIBLE, 0.00, 0.00, 'R3'),
    VALUE(VISIBLE, 0.00, 0.00, '10'),
    DEVICE(1, 0)
)
PART(5.70, 1.70, '0000000000000002B273DFFE', 'DIODE', NORMAL,
    REF(VISIBLE, 0.00, 0.00, 'D2'),
    VALUE(VISIBLE, 0.00, 0.00, '1N4004'),
    DEVICE(1, 0)
)
PART(6.70, 2.70, '0000000000000002B273E012', 'RESISTOR VAR', UP,
    REF(VISIBLE, 0.00, 0.00, 'R4'),
    VALUE(VISIBLE, 0.00, 0.00, '1K'),
    DEVICE(1, 0)
)
PART(7.60, 1.90, '0000000000000002B273E00D', 'CAPACITOR POL', NORMAL,
    REF(VISIBLE, 0.00, 0.00, 'C5'),
    VALUE(VISIBLE, 0.00, 0.00, '470 uF'),
    DEVICE(1, 0)
)
PART(5.80, 4.50, '0000000000000002B273E017', 'DIODE', NORMAL,
    REF(VISIBLE, 0.00, 0.00, 'D3'),
    VALUE(VISIBLE, 0.00, 0.00, '1N4001'),
    DEVICE(1, 0)
)
PART(6.10, 4.20, '0000000000000002B273E01C', 'DIODE', DOWN,
    REF(VISIBLE, 0.00, 0.00, 'D4'),
    VALUE(VISIBLE, 0.00, 0.00, '1N4001'),
    DEVICE(1, 0)
)
PART(5.20, 4.50, '0000000000000002B273E021', 'BATTERY', OVER,
    REF(VISIBLE, 0.00, 0.00, 'BT1'),
    VALUE(VISIBLE, 0.00, 0.00, '4V'),
    DEVICE(1, 0)
)
PART(6.40, 1.70, '0000000000000002B273E008', 'LM7805', NORMAL,
    REF(VISIBLE, 0.00, 0.07, 'U5'),
    VALUE(VISIBLE, 0.34, -0.03, 'LM123'),
    DEVICE(1, 0)
)
PART(6.00, 1.50, '0000000000000002B273E003', 'NPN', UP,
    REF(VISIBLE, 0.00, -0.03, 'Q2'),
    VALUE(VISIBLE, -0.15, -0.02, 'TIP31C'),
    DEVICE(1, 0)
)
```

*Figure 15-2. Example **Decompile Schematic** output file (continues).*

```
MODPORT(7.30, 4.60, 'BACKUP', 0.60, PLAIN, 'UNSPECIFIED')
POWER(6.20, 4.10, '+5V', 'CIRCLE', TOP)
POWER(3.50, 3.40, 'GND', 'ARROW', BOTTOM)
POWER(5.10, 5.00, 'GND', 'ARROW', BOTTOM)
POWER(8.20, 1.50, '+5V', 'CIRCLE', RIGHT)
JUNCTION(4.50, 1.80)
JUNCTION(6.10, 1.80)
JUNCTION(7.70, 1.80)
JUNCTION(3.50, 3.30)
JUNCTION(4.50, 3.30)
JUNCTION(6.60, 3.30)
JUNCTION(6.80, 3.30)
JUNCTION(6.20, 4.60)
WIRE(2.90, 1.40, 3.90, 1.40)
WIRE(4.50, 1.50, 5.90, 1.50)
WIRE(6.30, 1.50, 8.20, 1.50)
WIRE(1.10, 1.60, 2.00, 1.60)
WIRE(2.40, 1.60, 2.90, 1.60)
WIRE(4.30, 1.80, 5.10, 1.80)
WIRE(1.10, 2.00, 2.00, 2.00)
WIRE(2.40, 2.00, 2.90, 2.00)
WIRE(2.90, 2.20, 3.90, 2.20)
WIRE(3.50, 3.30, 7.70, 3.30)
WIRE(5.50, 4.60, 5.70, 4.60)
WIRE(6.20, 4.60, 7.30, 4.60)
WIRE(2.90, 1.40, 2.90, 1.60)
WIRE(2.90, 2.00, 2.90, 2.20)
WIRE(3.50, 1.80, 3.50, 3.40)
WIRE(4.50, 1.50, 4.50, 1.80)
WIRE(4.50, 2.20, 4.50, 3.30)
WIRE(5.10, 4.60, 5.10, 5.00)
WIRE(6.60, 3.00, 6.60, 3.30)
WIRE(6.80, 2.50, 6.80, 2.60)
WIRE(7.70, 1.50, 7.70, 1.80)
WIRE(7.70, 2.20, 7.70, 3.30)
END()
```

*Figure 15-2. Example **Decompile Schematic** output file (continued).*



Compile Schematic

Compile Schematic creates schematic (.SCH) files from ASCII text files. The text files must have the format created by **Decompile Schematic**, as described in the preceding chapter.

You typically use **Decompile Schematic** to create text files from worksheets, use a text editor to modify the text files, and then use **Compile Schematic** to create new worksheets.

Execution

With the **Schematic Design Tools** screen displayed, select **Compile Schematic**. Select **Execute** from the menu that displays.

When the schematics are compiled, the **Schematic Design Tools** screen displays.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Compile Schematic**. Select **Local Configuration** from the menu that displays.

Select **Configure AIM32**. A configuration screen displays (figure 16-1).

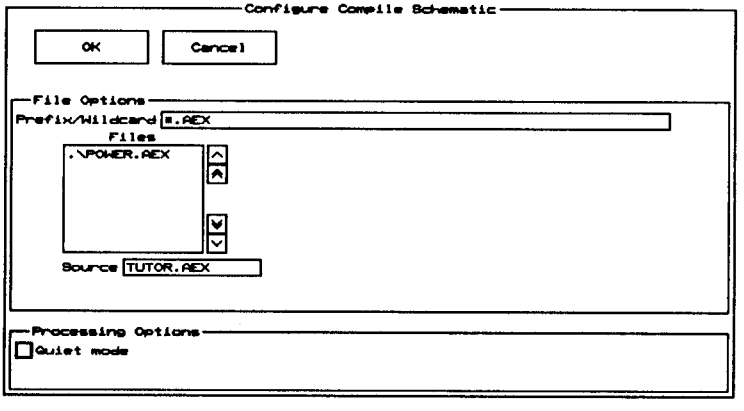


*Figure 16-1. **Compile Schematic's** local configuration screen.*

File Options **File Options** defines the source file or files.

Prefix/Wildcard The **Prefix/Wildcard** defines the drive and directory containing the text files to use as source. It also serves as a "filter" of the files listed in the **Files** list box. It may have any valid pathname, including the standard DOS asterisk (*) and question mark (?) wildcards.

For example:

`Prefix/Wildcard` `*.AEX`

Lists all files in the current design directory with a .AEX extension. This is the default.

`Prefix/Wildcard` `..\MYDESN\*.AEX`

Lists all files in the MYDESN design directory with a .AEX extension.

`Prefix/Wildcard` `SHEET?.*`

Lists all files with a filename of "SHEET" followed by a single character, regardless of the extension.

Source The **Source** is any valid DOS file specification, including the standard DOS asterisk (*) and question mark (?) wildcards. Pick a filename from the **Files** list box, or enter a file specification here.

For example:

`Source` `*.AEX`

Tells **Compile Schematic** to process all files in the current design directory with a .AEX extension.

Processing Options If desired, select the following option:

- ☐ `Quiet mode`

  Turns quiet mode on.

PART IV: LIBRARIANS

Included with **Schematic Design Tools 386+** are extensive part libraries containing more than 20,000 of the most commonly used devices in the electronics industry. These OrCAD-supplied libraries include many parts used in many types of schematic designs, so for most design work, OrCAD libraries contain all the parts you need. Sometimes, though, you may want to create your own custom parts that meet special requirements in a particular design.

Part IV describes library parts, library part files, and the librarian tools used to create and maintain your own custom parts.

Chapter 17: *About libraries* introduces the two forms of a library file: library source files and compiled library files. This chapter also describes two ways to edit a library part and lists the basic components of a part.

Chapter 18: *List Library* describes how to list a library's contents with **List Library**.

Chapter 19: *Archive Parts in Schematic* describes how to use **Archive Parts in Schematic** to make a library source file or a library string file containing only those parts used in a particular schematic worksheet.

Chapter 20: *Edit Library* describes the **Edit Library** graphical part editor and its commands.

Chapter 21: *Decompile Library* describes how **Decompile Library** converts a compiled library file into a library source file.

Chapter 22: *Creating a library source file with a text editor* describes, in detail, how to create a library source file with a text editor.

Chapter 23: *Symbol Description Language* explains how to define a part in a custom library using OrCAD's Symbol Description Language (SDL).

Chapter 24: *Compile Library* describes how **Compile Library** converts a library source file into a compiled library file.



About libraries

This chapter introduces library files and library parts. It describes the two forms of a library file, two ways to create library files, and the components of a library part.

Library files

Libraries are used to group and organize the more than 20,000 parts supplied with **Schematic Design Tools 386+.**

Parts in a particular library have common characteristics, such as parts produced by one manufacturer, parts with a common application, or parts in a family. Some examples are: TTL.LIB, INTEL.LIB, ANALOG.LIB, PCBDEV.LIB, or PLDGATES.LIB.

Some libraries contain hundreds of parts; some contain just a few. **Schematic Design Tools 386+** includes many libraries.

The libraries have been created over time. Old parts are not deleted because you may want to look at a design containing an old part. Also, some pre-release parts (from various manufacturers) are included.

Library files come in two forms: library source files and compiled library files.

Library source file

A *library source file* is much like the source code for a computer program. It is a text file containing instructions in the OrCAD's Symbol Description Language, which is described in *Chapter 23: Symbol Description Language.* These instructions specify how to represent a part graphically.

You can create and modify a library source file with the **Edit File** editor. You then run the tool called **Compile Library** on the library source file to produce a compiled library file.

Decompile Library and **Archive Parts in Schematic** tools can also produce library source files. For details on **Compile Library, Decompile Library,** and **Archive Parts in Schematic,** see chapters 24, 21, and 19, respectively.

Compiled library file

A *compiled library file* is the compiled version of a library source file. This type of file is produced by **Compile Library** or the **Edit Library** graphical part editor (described in chapter 20). This is the type of library file the schematic editor, processors, reporters and transfer tools can read. A compiled library file takes much less disk space than its corresponding library source file and is faster to load and access.

By convention, the names of compiled library files have .LIB as the file extension. For example, MOTO.LIB is the name of the OrCAD-supplied library of Motorola components. Also, by convention, library source files have a .SRC file extension. Note, however, the .LIB and .SRC extensions are conventions, not requirements.

When you configure **Schematic Design Tools 386+,** you specify which of the compiled libraries are loaded when you run a schematic editor, processor, reporter or transfer tool. For details, see *Chapter 1: Configure Schematic Tools.*

△ *NOTE: Libraries can be loaded into main system memory (RAM) or your disk. See* Chapter 1: Configure Schematic Tools *for details.*

You may have parts with the same name in different libraries. If you do and these libraries are selected during configuration, when **Draft** looks for a part, it searches the libraries in the order that they are listed in the **Configured Libraries** list box on the **Configure Schematic Design Tools** screen. The first part **Draft** finds with that name is the part **Draft** gets.

List parts in a library

You can view the names of the parts in a library using the **List Library** tool. For example, one of the OrCAD libraries is called TTL.LIB. To find out what parts are in this library, follow these steps:

1. Display **List Library**'s local configuration screen.
2. Select a library (in this case, TTL.LIB) from the list shown in the **Files** list box. Click on its name or type its name in the **Source** entry box.
3. Click **OK**.
4. Double-click on **List Library**.

List Library displays a list of the part names in TTL.LIB in the monitor box at the bottom of the screen. You can also tell **List Library** to send its output to a file instead of the screen. For details on **List Library**, see chapter 18.

Creating library files

There are two ways to create your own custom libraries or modify existing libraries.

▲ *CAUTION: Avoid changing parts and saving them in libraries with the same names as OrCAD libraries. Instead, create your own custom libraries with unique names. This way you can avoid the risk of losing your parts when OrCAD updates the libraries in the future.*

Edit Library

One way is to use **Edit Library**, a graphical part editor. With **Edit Library**, you use commands similar to **Draft**'s to construct or modify a part graphically and add it to a new or existing library. The part being edited appears on **Edit Library**'s screen exactly as it will appear when you place it on a schematic worksheet with **Draft**. **Edit Library** automatically converts the graphical screen symbols to the compiled format the schematic editor, processors, reporters, and transfer tools use.

You can use OrCAD libraries as the starting point for your own custom libraries. Make a copy of the OrCAD library file and then modify the parts with **Edit Library**. Chapter 20 describes **Edit Library**.

Text editor

Another way to create a custom library is using a text editor. You create a library source file, then run **Compile Library** on it to produce a compiled library file. Library source files are described in chapter 22. **Compile Library** is described in chapter 24.

Figure 17-1 shows the development process using **Compile Library** to create a custom part library.

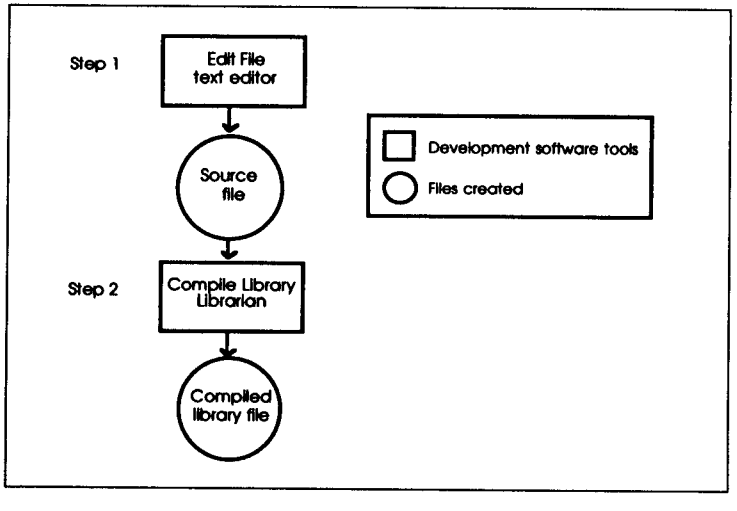


*Figure 17-1. How to use **Compile Library** to develop a library.*

You can also use OrCAD libraries as the starting point for your custom library. Select a library file, convert it to source form using **Decompile Library**, and then edit it using **Edit File**. When you are done editing, compile the file using **Compile Library**. Figure 17-2 shows this process.

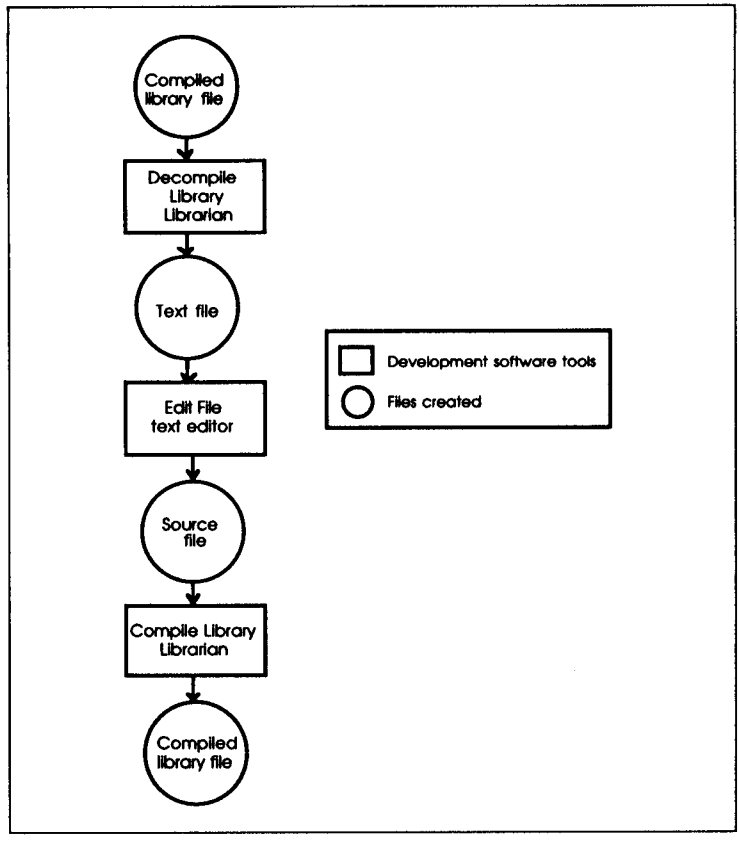


*Figure 17-2. How to use **Decompile Library** and **Compile Library** together to develop a library.*

Components of a library part

Whether you use **Edit Library** or a text editor along with **Compile Library** and **Decompile Library**, you build library parts from the same basic components.

A library part is made up of the following components:

- Body
- Pins
- One or more names
- An optional sheetpath designator
- An optional reference designator
- Optional text

These components are described on the next pages.

Body

Every part has either a block, a graphic, or an IEEE body.

Block

A block part is one whose body is a square or rectangle. For example, a memory chip is a block part. Figure 17-3 shows examples of parts with block bodies. A block part can be up to 25.5 inches by 20.0 inches in size.

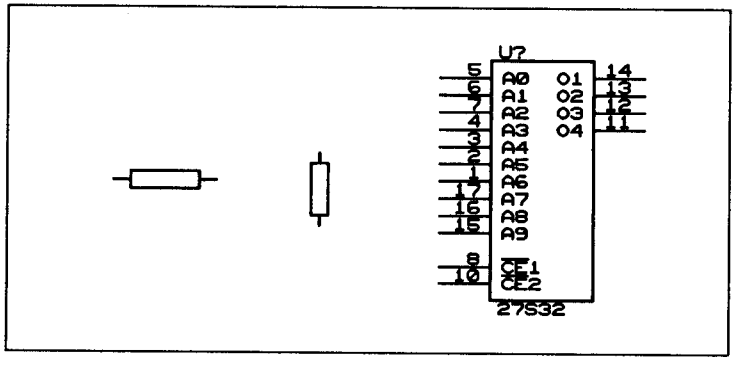


Figure 17-3. Parts with block bodies.

Graphic A graphic part is one whose body contains graphical information such as circles, arcs, and filled areas. It may also contain lines and text. A graphic part can be a simple square or rectangle, or it can be something more complex, such as an OR gate. A graphic part has no visible pin names.

Figure 17-4 shows examples of parts with graphic bodies.

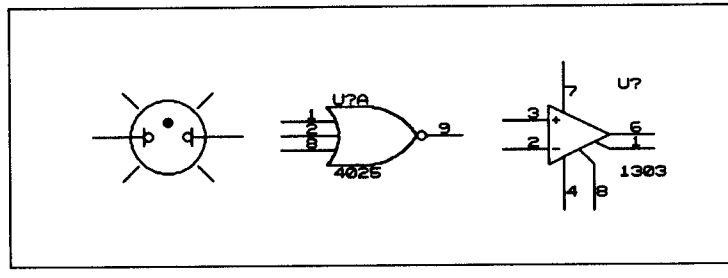


Figure 17-4. Parts with graphic bodies.

IEEE An IEEE part is one that complies with the ANSI/IEEE standard.[1] IEEE parts differ from graphic parts in that they may not contain arcs or fills. An IEEE part can be up to 25.5 inches by 20.0 inches in size. IEEE parts cannot be rotated on the schematic.

[1] See ANSI/IEEE Std 91-1984: *IEEE Standard Graphic Symbols for Logic Functions*, © 1984, or *Graphic Symbols for Electrical and Electronics Diagrams*, © 1975; both published by The Institute of Electrical and Electronics Engineers, Inc.

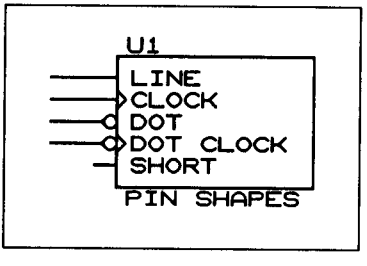
Pins Each pin has a type, a shape, a name, and possibly a number.

Pin type Pin types are:

- ❖ Input
- ❖ Output
- ❖ Bidirectional
- ❖ Power
- ❖ Passive
- ❖ 3 state
- ❖ Open collector
- ❖ Open emitter

A pin can have only one type. Pin type is not apparent from the representation of the part on the screen. Power pins are invisible.

Pin shape The pin shape determines how the pin appears. A pin can have either a normal or a short lead. A normal lead can also have a clock symbol and an inversion bubble (called a dot); a short lead cannot.



The figure above shows a part with each type of pin shape. A clock symbol is shown as a ">" on the lead. An inversion bubble, or dot, is shown as a circle on the lead.

Pin number If the part has pin numbers they appear outside the part beside the pins. Figures 17-3 and 17-4 show parts with pin numbers.

△ *NOTE: For parts with alphanumeric pin numbers, use Grid Array parts.*

Pin name The pin name is associated with the pin function. For example, A0 represents an address line, and CLR represents the clear function. In block parts, pin names appear inside the part beside the pins. In IEEE parts, pin names appear outside the body of the part near the pin when the part is viewed using **Edit Library**; however, the IEEE part's pin names do not display in **Draft**, nor will they print or plot. Graphic parts do not display pin names. They are, however, stored with the part.

Names A part has one or more names. Parts with identical symbols are represented in a library as one part with multiple names. For example, 2114, 2146, and 2149 identify the same symbol and represent a 1K x 4 static RAM.

Sheetpath designator A sheetpath designator is a filename referencing a schematic file that is used as a part. As such, a sheetpath part should be saved in the library directory so it can be used in any schematic.

The sheetpath designator is optional. Sheetpath designators provide a higher level of abstraction for the circuit under construction and are useful for frequently used circuits of macro functions, such as gate arrays or FPGAs.

Reference designator The reference designator specifies the default reference used when the part is first placed on a worksheet with **Draft**.

The prefix of the reference designator represents the class of the part. For example, U is normally used for ICs, Q is for transistors, C is for capacitors, and R is for resistors. The reference designator is optional and will default to "U" if one is not specified.



List Library

List Library lists the names of the parts in a library. The list can be shown on the screen or sent to a text file you view using **Edit File**.

Execution

With the **Schematic Design Tools** screen displayed, select **List Library**. Select **Execute** from the menu that displays.

- If a **Destination** is not specified on **List Library**'s local configuration screen, the parts are listed in the monitor box at the bottom of the screen. If the report is configured to report the total number of devices in the library, the parts are not listed (see *Local Configuration* in this chapter).
- If a destination is specified on **List Library**'s configuration screen, use **Edit File** to view the file created by **List Library**.

When **List Library** is complete, the **Schematic Design Tools** screen appears again.

Figure 18-1 shows a sample of one kind of list produced by **List Library**. Follow the prefix column down to the number on the left that corresponds to the library part you want. Two asterisks (**) mean the part is in the library you listed. Two periods (..) mean the part is not in the library you listed.

TTL.LIB

	74LS	74S	74ALS	74AS	74HCT	74HC	74ACT	74AC	74AHCT	74FCT	74F	74C	74
00	**	**	**	**	**	**	**	**	**	..	**	**	**
01	**	**	**	..	..	**	..	..	**	..	..	..	**
02	**	**	**	**	**	**	**	**	**	..	**	**	**
03	**	**	**	..	**	**	..	..	**	..	..	..	**
04	**	**	**	**	**	**	**	**	**	..	**	**	**
05	**	**	**	..	**	**	..	..	**	..	..	..	**
						and so on							
75188	..	..	..	..	..	..	..	..	..	..	..	..	..
75189	..	..	..	..	..	..	..	..	..	..	..	..	..

Figure 18-1. An example of List Library output for TTL.LIB.

For example, follow the column under "74AS" down to the "04" row. The double asterisks means the part 74AS04 is in the TTL.LIB library.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **List Library**. Select **Local Configuration** from the menu that displays.

Select **Configure LIBLIST**. A configuration screen displays (figure 18-2).

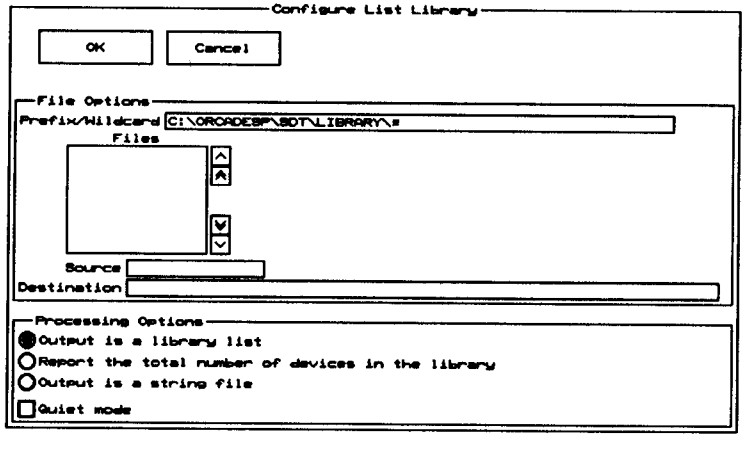


Figure 18-2. List Library's local configuration screen.

File Options **File Options** tells the library to be listed and the destination of the output.

Prefix/Wildcard Enter a pathname and wildcard to indicate which files to display in the list box with scroll buttons. The asterisk character (*) is used as a wildcard.

The default is:

```
\ORCADESP\SDT\LIBRARY\*.LIB
```

If you erase the entire field, the entry is restored to the prefix specified in **Configure Schematic Design Tools**.

Files The files that match the search filter entered in the **Prefix/Wildcard** entry box and those that match the filter in the current design directory are listed in this box. Files in the current directory are shown with .\ before their names. Use the scroll buttons at the right of the box to scroll the list of libraries up and down.

When you see the library you would like to list, select it. Its path and filename display in the **Source** entry box.

Source The **Source** names an existing compiled library file.

Specify the source file by selecting it from the list box or by simply entering its name in the **Source** entry box.

Destination The **Destination** is any valid pathname and is where the list is placed. If you name an existing file, when you run **List Library**, it asks if you want to overwrite the existing file. You cannot append to an existing file.

If a **Destination** is not specified, the output displays in the monitor box at the bottom of the screen.

Processing Options

Choose one of the following types of output files:

- ○ `Output is a library list`

  Causes **List Library** to create a list of part names in the design.

- ○ `Report the total number of devices in the library`

  Causes **List Library** to simply report how many parts are in the library, rather than create a list of parts.

- ○ `Output is a string file`

  Causes **List Library** to list the names of the parts in the design in string file format. A string file is designed to serve as the "parts affected" or left column of a stuff file with the processor **Update Field Contents.**

  Below is an example of a short string file listing the affected part name:

```
'GND'
'74LS00'
'RESISTOR'
'SW SPDT'
```

Select the following if you wish to turn quiet mode on:

- ☐ `Quiet mode`



Archive Parts in Schematic

Archive Parts in Schematic takes all the library parts used in a set of schematic files and makes a library source file or a library string file containing only those parts used in the schematic files.

Archive Parts in Schematic is a two-process tool. The first process, called LIBARCH, builds a library of all the components in the design. The second process, called COMPOSER, compiles the source file produced by LIBARCH into a form usable by **Draft**, processors, reporters, and transfer tools. Each of the two processes can be independently configured and turned on and off.

To achieve more efficient use of memory, run **Archive Parts in Schematic** on each design, turning both LIBARCH and COMPOSER on. This creates a library containing only the parts used in your design. Doing this protects the design from changes in standard libraries and results in more efficient memory use because you only have to configure one library instead of several.

Execution

With the **Schematic Design Tools** screen displayed, select **Archive Parts in Schematic** and then select **Execute** from the menu that displays.

The monitor box at the bottom of the screen displays messages while **Archive Parts in Schematic** creates the source file or the string file.

When **Archive Parts in Schematic** is complete, the **Schematic Design Tools** screen appears. You may look at the file created by **Archive Parts in Schematic** using **Edit File**.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Archive Parts in Schematic**. Select **Local Configuration** from the menu that displays. The menu below displays.

This menu includes options to configure LIBARCH and COMPOSER, as well as to turn each process on or off. When you run **Archive Parts in Library** only the processes that are turned on run. To turn a process on or off, choose the desired process from the menu. For example, to turn COMPOSER on, select **COMPOSER on** from the menu. The design environment prompts:

```
Select Configuration
Configure LIBARCH
Configure COMPOSER
LIBARCH    on
COMPOSER   on
```

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **off** to turn the process off.

To configure LIBARCH, follow the instructions in the *Configure LIBARCH* section on the next page.

To configure COMPOSER, see the *Configure COMPOSER* section in this chapter or the *Local Configuration* section of *Chapter 24: Compile Library.*

If you want to run both processes at the same time, leave both LIBARCH and COMPOSER on. Otherwise, set LIBARCH on, and COMPOSER off.

Configure LIBARCH

Select **Archive Parts in Schematic** from the **Schematic Design Tools** screen. Select **Local Configuration** from the menu that displays, and then select **Configure LIBARCH**. A configuration screen displays (figure 19-1).

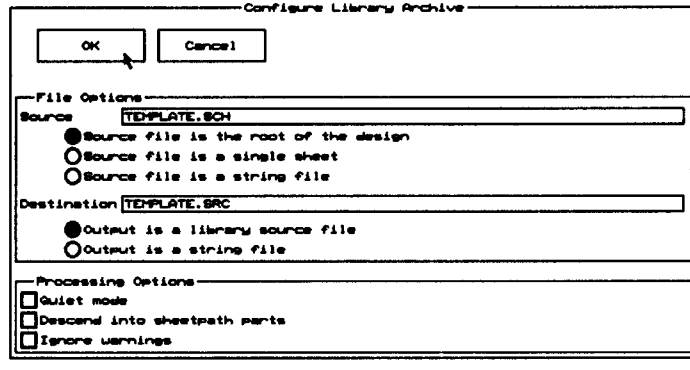


Figure 19-1. LIBARCH's local configuration screen.

File Options

File Options defines the source filename and its type. It also defines the destination filename and its type.

Source

The **Source** may be the root sheet name of a hierarchical or flat design, the filename of a one-sheet file, or a string file. If a pathname is not given, LIBARCH looks for the file in the current design directory.

After entering the source filename, select one of the following options:

- O `Source file is the root of the design`

  Specifies that the design consists of more than one worksheet. If the root sheet contains sheet symbols, the design is a hierarchy. If the root sheet contains !LINK followed by a list of files, the design is flat.

- O `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

O `Source file is a string file`

Specifies that the source file is a string file. LIBARCH creates a library source file containing the parts listed in the string file.

You can create a string file from a design by running LIBARCH with the **Output is a string file** option selected in its local configuration.

To create a special library with the string file output of LIBARCH, run LIBARCH again on the string file created the first time you ran LIBARCH. This time, make the output a library file, and the source the string file you created the first time you ran LIBARCH

Destination

The **Destination** is any valid path and filename and is where the library source file or string file is placed. If a filename is given without a pathname, LIBARCH places the resulting library source file or string file in the current design directory. The default name for the file is *design*.SRC, where *design* is the name of the current design.

If **Destination** is not specified, the output is directed to the monitor box at the bottom of the screen. If the destination is the name of an existing file, LIBARCH asks if you want to overwrite the existing file.

After entering the destination filename, select one of the following options:

O `Output is a library source file`

Tells LIBARCH to make a source file (a text file describing the parts using OrCAD's Symbol Description Language).

- ○ `Output is a string file`

  Tells LIBARCH to make a string file. A string file is a text file consisting of the names of all the parts used in a schematic file. The names are delimited with single quotes and each appears on a separate line. A string file can be used to create an include file for the **Create Bill of Materials** reporter, or to create a file to be used to create a special library.

  To create a special library with the string file output of LIBARCH, run LIBARCH again on the string file created the first time you ran LIBARCH. This time, make the output a library file, and the source the string file you created the first time you ran LIBARCH.

Processing Options

Select any combination of the following options:

- ☐ `Quiet mode`

  Turns quiet mode on.

- ☐ `Descend into sheetpath parts`

  Tells LIBARCH to descend into any parts defined as sheetpath parts. Without this option selected, LIBARCH treats the sheetpath itself as a part to be archived and does not archive the parts within a sheetpath.

- ☐ `Ignore warnings`

  Prevents LIBARCH from displaying a message when it encounters warnings.

Configure COMPOSER

Select **Configure COMPOSER**. The local configuration screen for COMPOSER displays. This is the same local configuration screen as for **Compile Library**.

For information about configuring COMPOSER, see *Local Configuration* in *Chapter 24: Compile Library*, replacing references to **Compile Library** with COMPOSER.



Edit Library

Edit Library is a graphical part editor used to create and edit library parts. This chapter describes how to configure and run **Edit Library**, tells how **Edit Library** uses vectors, and lists some part limitations you should consider when using **Edit Library**. A detailed description of the **Edit Library** commands is given in the *Command reference* section in this chapter.

About Edit Library

Use **Edit Library** to:

- Create a new part and add it to an existing or a new library.
- Create a part similar, but not identical, to a part in an OrCAD library. You run **Edit Library**, get the similar part, write it out to a new or existing library, edit it, and then save the changes to a custom library.
- Modify an existing part in an existing library.

See *Chapter 5: Creating a custom component* in the *Schematic Design Tools 386+ User's Guide* for an example that shows how to use **Edit Library** to create a new part and add it to an existing library file.

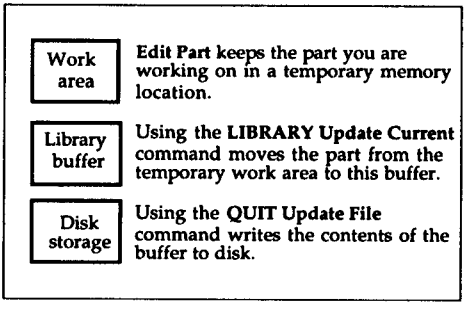
▲ *CAUTION: Avoid changing parts and saving them in libraries with the same names as OrCAD libraries. Instead, create your own custom libraries with unique names. This way you can avoid losing your parts when OrCAD updates the libraries.*

To modify parts in one of OrCAD's libraries, extract the parts (using GET PART), export them to a temporary data file, import the data from the temporary data file into a custom library, edit the parts' definitions, and then write them to the custom library.

Editing a part with Edit Library

To create or edit a part in a library, follow these steps:

1. Select **Edit Library's Local Configuration** option. Select the library to edit from the **Files** list box. If you want to create a new library, enter its name in the **Source** entry box.
2. Run **Edit Library. Edit Library** reads the library file into memory or creates the new library you named.
3. Create a part, or retrieve a part with the **GET PART** command and edit it. The changes you make are stored in a temporary work area; the copy of the library in memory remains unchanged.



4. Select the **LIBRARY Update Current** command. The copy of the library in memory gets the new or modified part. You could instead decide to create an export file and not modify the library (see the **Export** command description in this chapter).

5. Select the **QUIT Update File** command (saves the modified library to the same file) or **QUIT Write to File** command (saves the *entire* modified library to a new file you specify).

   Issuing either of these commands without previously issuing **LIBRARY Update Current** results in no modification to the library, even if you have retrieved and modified or constructed a part.

▲ *CAUTION: Update the library using the **LIBRARY Update Current** command before you select **QUIT Update File** or **QUIT Write to File** or the current part will be lost.*

Editing existing parts to create new parts

It is often convenient to create a new part by editing an existing part. To do this, follow these steps:

1. Get the part.
2. Use the **Name Add** command to give it the new name. Use the **Name Delete** command to delete the old name. Do this before other editing to protect the old part from being overwritten.
3. Edit the part using **Edit Library**'s commands.
4. Select **LIBRARY Update Current** and **QUIT Update File** as needed.

Execution

With the **Schematic Design Tools** screen displayed, select **Edit Library**. Select **Execute** from the menu that displays.

Edit Library's work area displays. Press <Enter> to display **Edit Library's** main menu. These commands and the menus and commands accessed by the main menu commands are described in the *Command reference* section in this chapter.

When you are finished editing library parts and leave the **Edit Library** tool, the **Schematic Design Tools** screen displays.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Edit Library**. Select **Local Configuration** from the menu that displays.

Select **Configure LIBEDIT**. **Edit Library's** local configuration screen appears (figure 20-1).

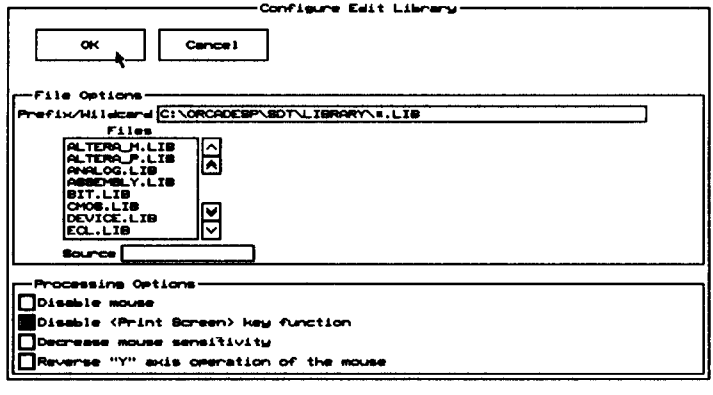


Figure 20-1. Edit Library's local configuration screen.

File Options — **File Options** defines a library for **Edit Library** to open.

Prefix/Wildcard

This entry box contains a pathname and wildcard to indicate which files to display in the list box. The asterisk character (*) is used as a wildcard.

The default is:

```
\ORCADESP\SDT\LIBRARY\*.LIB
```

If you erase the entire field, the entry is restored to the prefix as specified in the **Library Prefix** entry box on the **Configure Schematic Design Tools** screen.

Files list box

The files that match the search filter entered in the **Prefix/Wildcard** entry box and those in the current design directory that match the wildcard (for example, *.LIB) are listed in this box. Files in the current directory are shown with .\ before their names. Use the scroll buttons to scroll the list of libraries up and down.

When you see the library you would like to edit, click on it to select it. Its path and filename display in the **Source** entry box.

Source

The **Source** names a library file. The file can already exist, or you can enter a name for a new library.

To specify the source, select a name from the list box, or enter a new name in the **Source** entry box.

Processing Options You may select any combination of the following options:

- ❑ `Disable mouse`

  Disables the mouse. This option is normally used when attempting to debug mouse problems while working with OrCAD Technical Support. It may also be required when running on older PC-compatible systems.

- ❑ `Disable <Print Screen> key function`

  Disables **Edit Library**'s <Print Screen> key function. Use this option when you run other applications (usually RAM-resident) that make use of the <Print Screen> key. If this option is *not* selected, **Edit Library** uses the <Print Screen> key to capture hardcopy output and blocks other uses.

- ❑ `Decrease mouse sensitivity`

  Slows the mouse down. Used for mouse devices that are too sensitive. For example, if you move your mouse a small distance and the pointer moves a large distance on the screen, select this option to make the pointer movement respond more closely to the mouse movement.

- ❑ `Reverse "Y" axis operation of mouse`

  Changes the way the mouse responds. If this option is selected, the pointer moves up when you pull the mouse toward you, and moves down when you push the mouse away from you.

Command reference

The remainder of this chapter is a command reference for **Edit Library**.

Commands are described in alphabetical order, the order in which they appear in **Edit Library**'s main menu. Main menu commands appear at the top of the first page describing the command. Many commands display other menus. Commands on these menus are described under the main menu command.

If you are not sure of the name of a command, use table 20-1 to quickly look up the command alphabetically or use table 20-2 to identify the task you want to accomplish and then identify the command. Once you know the name of the command, look it up in the command reference.

Some commands in the main menu also appear in several other menus. These commands (such as **FIND**, **JUMP**, and **ZOOM**) are described at the main menu level only. When a command occurs in another menu, see the main menu description of the command for an explanation of its use.

Selecting commands

Select **Edit Library** commands in one of two ways:

- Press the first letter of the command name. Occasionally a menu has more than one command beginning with the same letter. When this happens, use the method of selecting commands explained below.
- Display the main menu by pressing <Enter> or clicking the left mouse button. Move the highlight over the command you wish to select and press <Enter> or click the left mouse button.

Press <Esc> to return to the previous menu.

Command	*Menu commands*		
AGAIN	*none*		
BODY	Kind of Part? Block Graphic IEEE		
BODY <Block>	Size of Body	Kind of Part	
BODY <Graphic>	Line Circle Arc	Text IEEE Symbol Fill	Delete Erase Body
BODY <IEEE>	Line Circle	Text IEEE Symbol	Delete Erase Body
CONDITIONS	*none*		
EXPORT	*none*		
GET PART	*none*		
IMPORT	*none*		
JUMP	A-H tags	X location	Y location
LIBRARY	Update Current Delete Part	List Directory Prefix	Browse
MACRO	Capture List	Delete Read	Initialize Write
NAME	Add Prefix	Delete	Edit
ORIGIN	*none*		
PIN	Add Pin-Number Move	Delete Type Jump	Name Shape
QUIT	Update File Suspend to System	Write to File Abandon Edits	Initialize Run User Commands
REFERENCE	*none*		
SET	Auto Pan Left Button Show Body Outline	Backup File Macro Prompts Visible Grid Dots	Error Bell Power Pins Visible
TAG	A-H tags		
ZOOM	Center Select	In	Out

*Table 20-1. **Edit Library** menu commands.*

Category	*Task*	*Select*
Entering and editing objects and data	Display a menu with commands you use to define a part's type and draw the part.	**BODY**
	Add, delete, or edit a part name, or activate or deactivate a part name prefix.	**NAME**
	Add, delete, name, number, or edit pins on the current part.	**PIN**
	Specify or edit a part's reference designator.	**REFERENCE**
Accessing libraries	Write a definition of the current part to a file that you then add to a library using the IMPORT command.	**EXPORT**
	Read a part from an export file created with the **EXPORT** command.	**IMPORT**
	Retrieve a part from a library for editing.	**GET PART**
	Update the current library, list the part names in the library, browse through the library, delete a part from the library, and define prefixes.	**LIBRARY**
Navigating on the screen	Move the pointer to a specified location on the screen.	**JUMP**
	Change the amount of detail you see on the screen.	**ZOOM**
Repeating repetitive or complex tasks	Repeat the last main menu command.	**AGAIN**
	Define and use macros.	**MACRO**
Setting locations and conditions	Change the origin (0,0) to the pointer's current position.	**ORIGIN**
	Set the status of **Edit Library** parameters.	**SET**
	Assign tags to specific locations on the screen.	**TAG**
Showing status or other information	Display memory available for the macro buffer, system, library, and current part.	**CONDITIONS**
Leaving the program	Save parts and libraries, abandon edits without saving, initialize **Edit Library**, suspend to system, or exit the program.	**QUIT**

*Table 20-2. **Edit Library** commands by function.*

AGAIN

AGAIN repeats the last *main menu* command executed. For example, if the last command you selected is **BODY**, you may repeat **BODY** by selecting **AGAIN**.

AGAIN repeats commands only from the main menu. For example, if you choose a **SET** command—such as **SET Backup File**, and then choose **AGAIN**, the **SET** commands display, ready for you to choose another **SET** command.

BODY

BODY displays a menu used for drawing a part. Depending on previous settings, the **BODY** command displays one of three menus: the **Block, Graphic,** or **IEEE** menu (figure 20-2).

BODY <Block>

```
Size of Body
Kind of Part
```

BODY <Graphic>

```
Line
Circle
Arc
Text
IEEE Symbol
Fill
Delete
Erase Body
Size of Body
Kind of Part
```

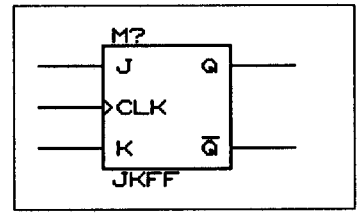
BODY <IEEE>

```
Line
Circle
Text
IEEE Symbol
Delete
Erase Body
Size of Body
Kind of Part
```

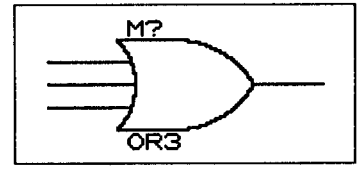
*Figure 20-2. **BODY's Block, Graphic, and IEEE menus.***

A part must be a block, a graphic, or an IEEE part. It cannot be a combination of any of these.

- A block part is one whose body is a square or rectangle. For example, a JK flip-flop is a block part. A block part can be up to 25.5 inches by 20.0 inches.

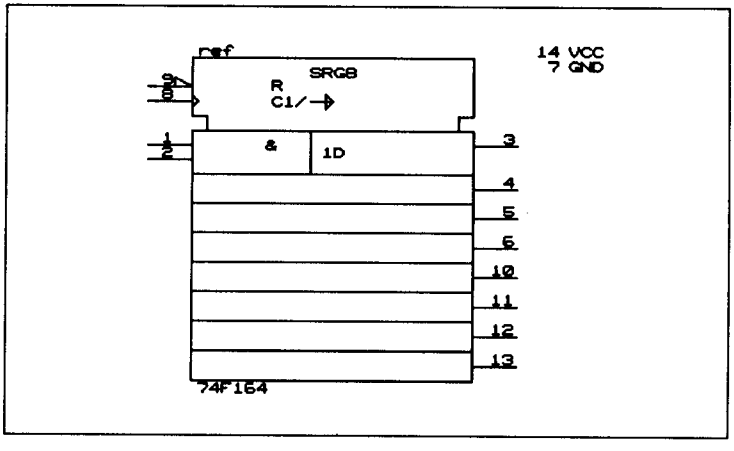


- A graphic part is one whose body contains graphical information such as circles, arcs, and filled areas. It may also contain lines and text. A graphic part can be a simple square or rectangle, or it can be something more complex, such as an OR gate. A graphic part has no visible pin names.



- An IEEE part is one that complies with the ANSI/IEEE standard.[2] IEEE parts differ from graphic parts in that they may not contain arcs or fills, and they are typically larger. An IEEE part can be up to 25.5 by 20.0 inches.

An example of an IEEE part



△ ***NOTE:*** *If you did not previously specify* ***Block, Graphic,*** *or* ***IEEE, Edit Library*** *requests the kind of part as described below. Depending on whether you select* ***Block, Graphic,*** *or* ***IEEE,*** *different questions are asked. These questions are described on the next page.*

BODY Kind of Part?

The first time you select **BODY**, the menu shown at the right displays. You must tell **Edit Library** which type of part you want to edit. Select **Block, Graphic,** or **IEEE.**

```
Kind of Part?
Block
Graphic
IEEE
```

[2] See ANSI/IEEE Std 91-1984: *IEEE Standard Graphic Symbols for Logic Functions,* © 1984, or *Graphic Symbols for Electrical and Electronics Diagrams,* ©1975; both published by The Institute of Electrical and Electronics Engineers, Inc.

BODY Kind of Part? Block

A block part is one whose body is a square or rectangle. For example, a JK flip-flop is a block part.

Number of Parts per Package

If you choose **Block**, **Edit Library** requests the number of parts per package. Choose a number from **0** to **16**.

Is Part a GRID ARRAY?

If the number of parts per package is 1, **Edit Library** displays "Is Part a GRID ARRAY?" Select **Yes** or **No**. The **Place** command displays (described on the next page).

By specifying one part per package on a block part and selecting **Yes** to this prompt, you can create a pin-grid array part. Pin-grid array parts have an alpha character for the first digit of each pin number.

BODY Kind of Part? Graphic

A graphic part is one whose body contains curves, arcs, fills, text, and lines. For example, an OR gate is a graphic part.

Is Part a GRID ARRAY?

If you choose **Graphic**, **Edit Library** displays "Is Part a GRID ARRAY?" Select **Yes** or **No**. The **Place** command displays (described on the next page).

Number of Parts per Package

If you choose **No** (the part is not a grid array), **Edit Library** requests the number of parts per package. Choose a number from **0** to **16**.

Does Graphic Part have CONVERT?

If you choose **No** (the part is not a grid array), **Edit Library** displays "Does Graphic Part have CONVERT?" Select **Yes** or **No**:

- If you choose **Yes**, **Edit Library** can create a *converted form* for your part (used to store an alternate form of the part—such as a DeMorgan equivalent symbol of the part). It is the version of the part that appears when you issue **Draft**'s **GET Convert** command.
- If you choose **No**, you can't create or store a converted form of the part.

The **Place** command displays (described on the next page).

BODY Kind of Part?
IEEE

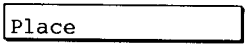
An IEEE part is one that complies with the ANSI/IEEE standard. IEEE parts differ from graphic parts in that they may not contain arcs or fills, and they are typically larger.

Is Part a GRID ARRAY?

If you choose **IEEE**, **Edit Library** displays "Is Part a GRID ARRAY?" Select **Yes** or **No**. The **Place** command displays (described on the next page).

Place

Once you select **Block**, **Graphic**, or **IEEE**, and respond to any prompts that display (as described on the previous page), the **Place** command displays, as shown at right.



To increase or decrease the part size, move the pointer. When the part size is what you want, select **Place**. **Edit Library** fixes the size of the part and displays the appropriate menu, as shown in figure 20-2.

Once you have defined the part body, use the displayed menu to place objects on the part, thereby building the part.

For most parts, you should place objects inside the part body. For graphic parts, only the area within the body displays. It is possible for objects such as circles to be placed so that they are partially outside the body. These will usually appear different on a plot. For example, the plotter often draws the entire circle.

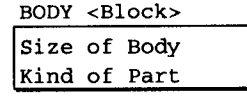
For IEEE parts, some objects have to be placed outside the body. In particular, the Active_Low IN and OUT symbols are normally placed on pins outside the body. Most objects should be inside the body in order to conform to the IEEE standard.

BODY command reference

In the following command reference, the **BODY <Block>** commands are described first, next the **BODY <Graphic>** commands are described, followed by the **BODY <IEEE>** commands.

BODY <Block> commands

The **BODY <Block>** commands are shown at the right. Descriptions of these commands follow.

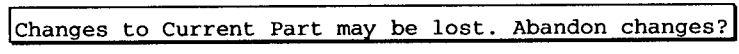


Body <Block> Size of Body

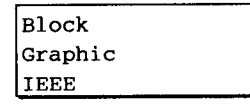
Use the **Size of Body** command to change the size of the edited part. To increase or decrease the part size, move the pointer. When the part is the desired size, select **Place**. The **BODY <Block>** menu displays (shown above).

Body <Block> Kind of Part

If you want to draw a different type of part, select **Kind of Part**. **Edit Library** displays:



Select **Yes**. The menu shown at right displays. Select the type of part to draw, as described previously in this section.



BODY <Graphic> commands

You can create graphic parts of virtually unlimited size. You can also create graphic grid array parts, which are single-element graphic parts with alphanumeric pin numbers.

The **BODY <Graphic>** commands are shown at the right. Descriptions of these commands follow.

```
BODY <Graphic>
Line
Circle
Arc
Text
IEEE Symbol
Fill
Delete
Erase Body
Size of Body
Kind of Part
```

BODY <Graphic> Line



Use the **Line** command to draw lines. This command is similar to **Draft's PLACE Wire** command.

Select **Line**. This command line displays:

```
Begin Jump Origin Tag Zoom
```

To draw a line, place the pointer where you want the line to start and select **Begin**. This command line displays:

```
Begin End New Jump Origin Tag Zoom
```

Move the pointer to draw the line. Then use the **Begin, End,** or **New** commands to finish drawing the line.

Begin

Use the **Begin** command to:

- Start drawing a line segment.
- Finish drawing a line segment and begin a new one (for example, if the line you are drawing makes a turn). You can use **Begin** over and over to draw a complex line.

End

Select **End** to end the current line and return to the **BODY <Graphic>** menu.

New

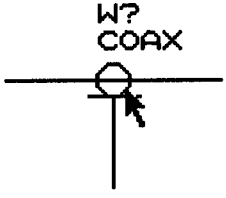
Select **End** to end the current line and continue to display the **Line** command line. You can begin a new line at a different location with **Begin.**

BODY <Graphic>
Circle

Use the **Circle** command to place a circle.

Select **Circle**. The following command line displays:

```
Center Jump Origin Tag Zoom
```



To place a circle, move the pointer to the point that will be the center of the circle. Select **Center**. The **Center** command changes to **Edge**, as shown below.

```
Edge Jump Origin Tag Zoom
```

Move the pointer. As you do this, a circle expands and contracts.

To place the circle, move the pointer to where you want the edge of the circle and select **Edge**. The **Edge** command changes back to **Center**, so you can draw another circle.

△ *NOTE: Circles may appear elliptical on your screen, depending on the type of monitor you have, because of variations between monitors. They will print or plot correctly, though.*

BODY <Graphic> Arc



Use the **Arc** command to place an arc. An arc is a section of a circle. You can draw an arc ranging from 0° to 90°.

Select **Arc**. The **Arc** command line displays:

```
Center Jump Origin Tag Zoom
```

This command line is identical to the **Circle** command line.

To place an arc, move the pointer to the center of the circle from which the arc is to be taken and select **Center**. The **Center** command changes to **Edge**, as shown below.

```
Edge Jump Origin Tag Zoom
```

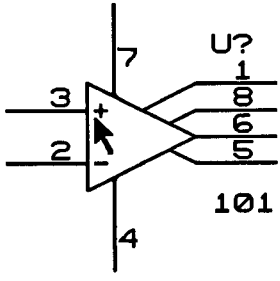
Move the pointer. As you do this, a circle expands and contracts. A radius extends from the center of the circle to the pointer. Move the pointer to where you want one end of the arc and select **Edge**. Then move the pointer to where you want the other end of the arc and select **Arc** again. This places the arc. The **Edge** command changes back to **Center**, so you can draw another circle.

➤ *Helpful hints . . .*

A circle is divided into four quadrants by invisible horizontal and vertical lines through the center. Each quadrant is 90°. An arc can only be drawn in the current quadrant, and you cannot move the pointer outside the current quadrant while you are drawing.

For finer control when drawing an arc, zoom in to ½ or ¼ scale.

BODY <Graphic> Text



Use the **Text** command to place comment text. Note that this text is not intended to be a reference designator for the part or a name for a pin.

Select **Text**. The following prompt displays:

```
Text?
```

Enter the comment text. The text displays at the pointer position. The command line shown below displays:

```
Place Larger Smaller Jump Origin Tag Zoom
```

Place

Select **Place** to place the text. The "Text?" prompt reappears, ready for you to enter more text.

Larger and Smaller

Select **Larger** or **Smaller** to make the text larger or smaller.

➤ *Helpful hints . . .*

You should normally place the text within the part outline. If you do place text outside the outline, you run the risk of a messy or ambiguous situation on the final worksheet. If you can't see the part outline, set the **SET Show Body Outline** command to **Yes**. The part outline then displays as a dotted line on the screen. **SET Show Body Outline** is described later in this chapter.

You may wish to use **BODY <Graphic> Text** to make the pin names visible on graphic parts. If you do this, the text rotates and mirrors with the graphic image. Be aware, however, that **Schematic Design Tools 386+** does not recognize the text as a pin definition.

BODY <Graphic>
IEEE Symbol

Use the **IEEE Symbol** command to place IEEE/ANSI special symbols. For more information about what these symbols mean, see *ANSI/IEEE Std 91-1984.*

Select **IEEE Symbol.** A menu listing IEEE symbols appears. This menu, the menus below it, and the IEEE symbols placed by each command are shown in figure 20-3.

△ ***NOTE: Edit Library*** *treats the negation symbol as a circle. That means a Negation symbol counts toward the maximum number of circles allowed in a part definition.*

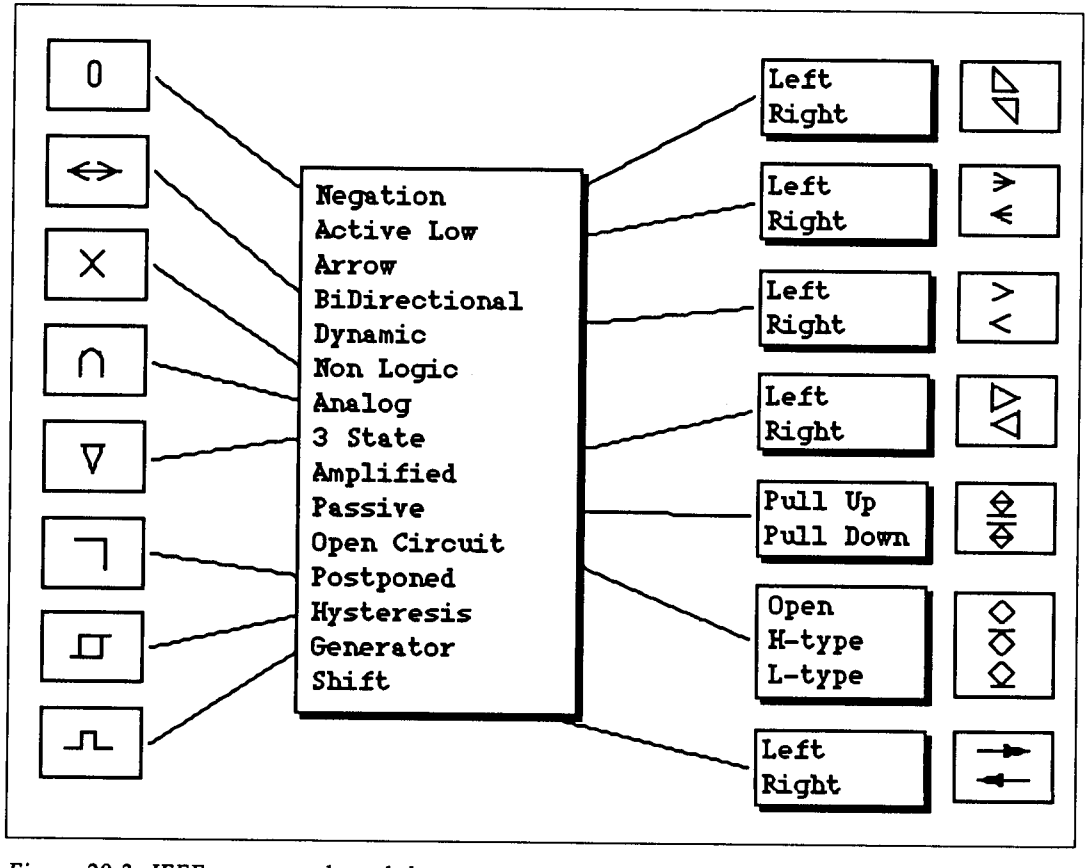


Figure 20-3. IEEE menus and symbols.

Select the command for the symbol you wish to include in your part. If another menu displays, select the desired command.

The **Place** command line displays:

```
Place Larger Smaller Jump Origin Tag Zoom
```

This command line is identical to the **BODY <Graphic> Text** command line.

Place

Select **Place** to place the symbol. You can place copies of the symbol as many times as you like.

Larger and Smaller

Select **Larger** to make the symbol larger and **Smaller** to make it smaller. The size specified by **Larger** and **Smaller** affects both IEEE symbols and text that you place with the **BODY <Graphic> Text** command. The size retains its last setting so you can place objects at the same scale.

➤ *Helpful hint . . .*

When the zoom scale is 1, you may not easily distinguish between a pointer actually on a symbol and a pointer just close to a symbol. When you zoom in, you may place objects in positions that are off grid points. At zoom scale 1, you may not be able to place the pointer on the object because the pointer stays on grid points at this scale. If so, zoom in once or twice to get the fine control you need.

BODY <Graphic> Fill

Use the **Fill** command to shade an enclosed area around the current pointer position. For example, you can Fill to darken the triangular shape in the symbol of a diode.



Select **Fill.** The following command line displays:

```
Fill Jump Origin Tag Zoom
```

To darken an enclosed area, move the pointer inside the area and select **Fill.**

➤ *Helpful hint . . .*

If you edit a part after you use **Fill,** you lose your fills. This protects against overflowing the entire part with a fill if you delete the fill boundary. Therefore, use **Fill** at the end of your editing session.

BODY <Graphic> Delete

Use the **Delete** command to delete an item.

▲ *CAUTION: Be careful when you delete an item; you cannot undo the deletion.*

Select **Delete.** The following command line displays:

```
Delete Jump Origin Tag Zoom
```

Delete

To delete an item, place the pointer on the item and select **Delete.**

➤ *Helpful hint . . .*

For the item to be deleted, the pointer must actually be on a pixel in the part. At zoom scale 1, it's not easy to distinguish between a pointer actually on an item and a pointer just close to an item.

BODY <Graphic> Erase Body

Use the **Erase Body** command to delete all objects within a part's graphic body. Use this command when you want to start over drawing the graphic body.

Select **Erase Body**. **Edit Library** asks you to confirm your choice. Select **Yes** to delete the graphic body. Select **No** to return to the **BODY** menu.

BODY <Graphic> Size of Body

Use the **Size of Body** command to change the size of the part you are editing.

Choose **Size of Body**. The **Place** command displays on the command line. Move the pointer to increase or decrease the part size. When the part is the desired size, select **Place**. The **BODY <Graphic>** menu displays.

▲ *CAUTION: Once objects are inside the body of the part, changing the size of the body can have undesirable results.*

BODY <Graphic> Kind of Part

To draw a different type of part, select **Kind of Part**. **Edit Library** displays:

```
Changes to Current Part may be lost. Continue?
```

Select **Yes**. The menu shown at right displays. Select the type of part to draw, as described previously in this section.

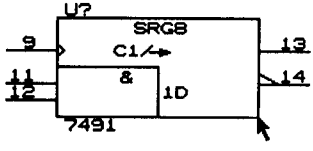
```
Kind of Part?
Block
Graphic
IEEE
```

BODY <IEEE> commands

The **BODY <IEEE>** commands are shown at the right. Descriptions of each of the commands on this menu follow.

```
Body <IEEE>
Line
Circle
Text
IEEE Symbol
Delete
Erase Body
Size of Body
Kind of Part
```

BODY <IEEE> Line



Use the **Line** command to draw lines. This command is similar to **Draft's PLACE Wire** command.

Select **Line**. The following command line displays:

```
Begin Jump Origin Tag Zoom
```

To draw a line, place the pointer where you want the line to start and select **Begin**. **Edit Library** adds the **End** and **New** commands to the command line:

```
Begin End New Jump Origin Tag Zoom
```

Move the pointer to draw the line. Then use the **Begin**, **End**, or **New** command to finish drawing the line.

Begin

Use the **Begin** command to:

- Start drawing a line segment.
- Finish drawing a line segment and begin a new one (for example, if the line you are drawing makes a turn). You can use **Begin** over and over to draw a complex line.

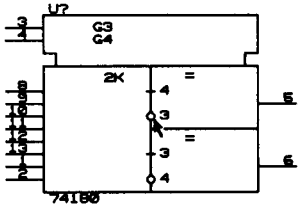
End

Select **End** to end the current line and return to the **BODY <IEEE>** menu.

New

Select **End** to end the current line and continue to display the **Line** command line. You can begin a new line at a different location with **Begin**.

BODY <IEEE>
Circle



Use the **Circle** command to place a circle.

Select **Circle**. This command line displays:

```
Center Jump Origin Tag Zoom
```

To place a circle, move the pointer to the point that will be the center of the circle. Select **Center**. The **Center** command changes to **Edge**, as shown below:

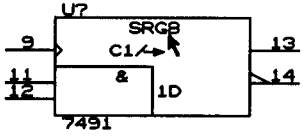
```
Edge Jump Origin Tag Zoom
```

Move the pointer. As you do this, a circle expands and contracts.

To place the circle, move the pointer to where you want the edge of the circle and select **Edge**. The **Edge** command changes back to **Center**, so you can draw another circle.

△ *NOTE: Circles may appear elliptical on your screen, depending on the type of monitor you have, because of variations between monitors. they will print or plot correctly, though.*

BODY <IEEE> Text



Use the **Text** command to place comment text. Note that this text is not intended to be a reference designator for the part or a name for a pin.

Select **Text**. The following prompt displays:

```
Text?
```

Enter the comment text. The text appears at the pointer position. The command line shown below displays:

```
Place Larger Smaller Jump Origin Tag Zoom
```

Place

Select **Place** to place the text. The "Text?" prompt reappears, ready for you to enter more text.

Larger and Smaller

Select **Larger** to make the text larger and **Smaller** to make it smaller.

➤ *Helpful hints . . .*

IEEE text is limited to ten characters per text object. If you enter more than ten characters in response to the "Text?" prompt, the extra characters are discarded. You can place two or more text objects together to create a string that is longer than ten characters.

Most of the time, you place the text within the part outline. It is sometimes necessary, however, to place text for IEEE parts *outside* the outline. If you do this, you risk a possibly messy or ambiguous situation on the final worksheet. If you can't see the part outline, set the **SET Show Body Outline** command to **Yes**. The part outline then displays as a dotted line on the screen. **SET Show Body Outline** is described later in this chapter.

You may wish to use **BODY <IEEE> Text** to make the pin names visible on IEEE parts. Be aware, however, that **Schematic Design Tools 386+** does not recognize the text as a pin definition.

BODY <IEEE> IEEE Symbol

Use the **IEEE Symbol** command to place IEEE/ANSI special symbols. For more information about what these symbols mean, see *ANSI/IEEE Std 91-1984.*

Select **IEEE Symbol.** A menu listing IEEE symbols appears. This menu, the menus below it, and the IEEE symbols placed by each command are shown in figure 20-3.

Select the command for the symbol you wish to include in your part. If another menu displays, select the desired command.

The **Place** command line displays:

```
Place Larger Smaller Jump Origin Tag Zoom
```

This command line is identical to the **Text** command line.

Use **Larger** and **Smaller** to change the size of the symbol. Move the pointer to the position where you want the symbol and select **Place.** desired. When you are done, press <Esc>. The **IEEE Symbol** menu (figure 20-2) displays.

Place

Select **Place** to place the symbol. You can place copies of the symbol as many times as you like.

Larger and Smaller

Select **Larger** to make the symbol larger and **Smaller** to make it smaller. The size specified by **Larger** and **Smaller** affects both IEEE symbols and text that you place with the **BODY <IEEE> Text** command. The size retains its last setting so you can place objects at the same scale.

➤ *Helpful hint . . .*

When the zoom scale is 1, you may not easily distinguish between a pointer actually on a symbol and a pointer just close to a symbol. When you zoom in, you may place objects in positions that are off grid points. At zoom scale 1, you may not be able to place the pointer on the object because the pointer stays on grid points at this scale. If so, zoom in once or twice to get the fine control you need.

BODY <IEEE> Delete

Use the **Delete** command to delete an item.

▲ *CAUTION: Be careful when you delete an item; you cannot undo the deletion.*

Select **Delete**. This command line displays:

```
Delete Jump Origin Tag Zoom
```

Delete

To delete an item, place the pointer on the item and select **Delete**.

➤ *Helpful hint . . .*

For the item to be deleted, the pointer must actually be on a pixel in the part. At zoom scale 1, it's not easy to distinguish between a pointer actually on an item and a pointer just close to a graphic item.

BODY <IEEE> Erase Body

Use the **Erase Body** command to delete all objects within a part's graphic body. Use this command when you want to start over drawing the body.

Select **Erase Body**. **Edit Library** asks you to confirm your choice. Select **Yes** to delete the graphic body. Select **No** to return to the **BODY** menu.

BODY <IEEE> Size of Body

Use the **Size of Body** command to change the size of the part you are editing.

Choose **Size of Body**. The **Place** command displays on the command line. Move the pointer to increase or decrease the part size. When the part is the desired size, select **Place**. The **BODY <IEEE>** menu displays.

▲ *CAUTION: Once objects are in the body of the part, changes to its body size can have undesirable results.*

BODY <IEEE>
Kind of Part

To draw a different type of part, select **Kind of Part. Edit Library** displays:

```
Changes to Current Part may be lost. Continue?
```

Select **Yes.** The menu shown at right displays. Select the type of part to draw, as described earlier in this section.

```
Kind of Part?
Block
Graphic
IEEE
```

CONDITIONS

CONDITIONS reports on the current library and its memory requirements. When you select **CONDITIONS**, a screen similar to the one shown in figure 20-4 displays. To return to the main menu level, press any key or mouse button.

The **CONDITIONS** screen has two areas: The upper area describes the current library, and the lower area describes the current part. If no part is current, only the upper area displays. The categories of information offered in the lower area is determined by the part type.

```
Press any key to continue

Library                                Used
    Objects                             634
    General Control Information       64435
    Graphic Images                    13462

Current Part          Size X = 60  Y =  40

  Part Type is: Graphic + Convert

  Pins Used                              7
  Pins Available                       248

  Normal Image
    Circles                              0
    Lines                                3
    Texts                                0
    Arcs                                 2
    Fills                                0
  Convert Image
    Circles                              2
    Lines                                2
    Texts                                0
    Arcs                                 4
    Fills                                0
```

*Figure 20-4. **CONDITIONS** screen.*

Library This area describes the current library.

Objects This shows the number of names in the current library. A library can have up to 1023 names (more if you use prefixes). Note the number represents names, not parts, since a part may have multiple names. However, extra names resulting from prefix definitions do not count. For more information about prefix definitions, see *Chapter 17: About libraries* in this guide.

General Control Information This shows the number of bytes used by library symbols. **General Control Information** symbols include part names, pin names, pin positions, block symbols, and so on.

Graphic Images This shows the number of bytes used by graphic symbols in the library. Graphic symbols include arcs, circles, lines, text, and IEEE symbols.

Current Part This area describes the current part.

Size Tells the size, in grid units, of the current part.

Type Tells the type of the current part. The types are:

- Block
- Block/GRIDARRAY
- Graphic
- Graphic/GRIDARRAY
- Graphic + Convert
- IEEE
- IEEE/GRIDARRAY

Pins Used Tells the number of pins used.

Pins Available Tells the number of pins available.

Normal Image Tells how many of these objects on graphic parts are used:

- Circles
- Lines
- Texts
- Arcs
- Fills
- IEEE symbols

Convert Image Tells how many objects on graphic parts with convert are used. The objects are the same types reported in the **Normal Image** area.

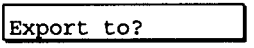
IEEE Image Tells how many of each type of object is used on IEEE parts. This table gives the memory requirement of each type of object:

Object	*Bytes*
Circle	7
Line	9
Text	7 plus 1 byte per character

EXPORT

EXPORT writes the current part definition (the one you are editing) to a file. The file is called an export file and is formatted as a library source file without a PREFIX-END statement. **EXPORT** writes only one part per export file.

Choose **EXPORT. Edit Library** displays the prompt shown at right.



Enter the name of the file to export the part to. If you don't specify a path, **Edit Library** places the part in the current design directory.

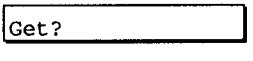
△ *NOTE: A part must have a name before you can write it to an export file. Use the NAME command to give the part a name.*

➤ *Helpful hints . . .*

To add the part in the export file to an existing library, run **Edit Library** and specify the desired destination library as the current library. Then use the **IMPORT** command to read the export file. You can then make changes to the part using **Edit Library** commands if necessary. Then, you can add the part to the library with the **LIBRARY Update Current** command. The following example illustrates this procedure.

The first part of the example demonstrates how to modify a part from a library—in this case TTL.LIB—and write it to a new file.

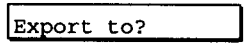
1. In **Edit Library's Local Configuration** screen, configure **Source** to be the library from which you will copy the part—in this case, TTL.LIB.
2. Run **Edit Library**.
3. Select **GET PART** from the main menu. **Edit Library** displays the prompt shown at right.



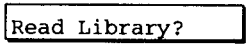
4. Choose a part from the library. For instructions on how to use **GET PART** to choose a library part, see the *GET PART* section in this chapter.

   The selected part displays on the screen.

5. Now, select **EXPORT. Edit Library** displays the prompt shown at right.



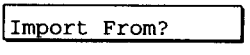
6. Enter the name of a temporary file for the data to be stored—in this case, **TEMP**.
7. Select **QUIT Initialize. Edit Library** displays the prompt shown at right.



8. Enter the new library name, for example **NEW.LIB**.

The next part of the example demonstrates how to read the part from the export file you just created.

1. Choose **IMPORT** to import the data from the temporary file. **Edit Library** displays the prompt shown at right.



2. Enter the name of the temporary data file in which you stored the exported data—in this case, **TEMP**.

The part is now ready for you to edit. When you have completed your editing, save the new library as demonstrated in the last part of the example.

1. Select **LIBRARY Update Current**. This modifies the copy of the library in memory, *not* the copy on disk.
2. Select **QUIT Update File**. This writes the latest copy of the library from memory to disk.
3. Select **Abandon Edits**.

GET PART

GET PART retrieves a part from a library for editing.

Select **GET PART. Edit Library** displays "Get?" You can use either of these two methods to retrieve parts:

Method 1	*Method 2*
Enter the desired part name exactly as it appears in the library directory. **Edit Library** searches the current library to find the part you requested. Once it finds the part, **Edit Library** displays the part on the screen. ➤ *Helpful hint . . .* To verify the spelling of a part name, use the **LIBRARY Directory** command.	Press <Enter>. **Edit Library** displays a menu listing the library parts. Move the highlight to the part name you want, then press <Enter> to retrieve the part. **Edit Library** displays the part on the screen. Type a character to snap the highlight bar to the first part whose name begins with that character. Use the <Page Up> and <Page Down> keys to scroll up and down the list of parts. The <Home> and <End> keys move the highlight to the first or last part in the list.

Getting a part by entering a part suffix

If you are using Method 1 (as described above), you can select library parts created with a prefix and short-hand string (see *Prefix definition* in *Chapter 17: About libraries*) from the library by entering the suffix. For example, suppose you want to retrieve a 74LS27 from the TTL.LIB library. After you select **GET**, you can enter any of the following examples to retrieve the part:

`Get?` **`74LS27`** The entire part name is used to retrieve the part.

`Get?` **`LS27`** The prefix "LS" (which is the shorthand string for "74LS") is combined with the suffix "27." **Edit Library** retrieves the part 74LS27.

```
Get? 27
```

Edit Library lists all parts in the library that have "27" in their names. Select the 74LS27 from the menu that displays the part names. **Edit Library** retrieves the part 74LS27.

IMPORT

IMPORT reads a part from an export file created with the **EXPORT** command. Export files contain only one part definition per file. **IMPORT** reads in the part definition as the current active part. It overwrites the existing current part.

Select **IMPORT**. **Edit Library** displays:

```
Import From?
```

Enter the name of the export file you want to import.

If you made any changes to the current part since you last used the **GET PART** or **LIBRARY Update Current** commands, a message displays that warns you about losing your changes. Select **No** or press <Esc> to cancel the **IMPORT** command.

JUMP

Use **JUMP** to quickly move the pointer to a different location in your work area. The specific locations can be tags or (X,Y) coordinates.

Select **JUMP**. The menu shown at right displays.

Jump

```
A tag
B tag
C tag
D tag
E tag
F tag
G tag
H tag
X location
Y location
```

JUMP A, B, C, D, E, F, G, H Tag

Select one of the **Tag** commands, the pointer jumps to the specified tag on the drawing (that you previously set with the **TAG** command). For information on the **TAG** command, see the *TAG* section in this chapter.

△ *NOTE: The error message "Tag does not exist" displays if the specified tag has not been set.*

JUMP X-Location

The **X-Location** command moves the pointer a specific distance along the X-axis. Each step represents 1/100 (0.01) inch on the worksheet (if the pin-to-pin spacing on the template table in the **Configure Schematic Design Tools** screen is set to the default value of 0.100 inch).

Select **X-Location**.

Edit Library displays "Jump X." Enter the number of steps to jump. The pointer jumps to the specified location and **Edit Library** returns to the main menu level.

➤ *Helpful hints . . .*

A number without a plus or minus sign moves the pointer to the actual grid coordinate, a signed positive number (+10, +25, +30, and so on) moves the pointer to the right, and a negative number (–10, –25, –30, and so on) to the left. If you enter +10, for example, the pointer jumps to the right 0.1 inch. If you enter 10, without a plus or minus sign, the pointer moves to the actual X grid coordinate 10.0.

JUMP Y-Location

The **Y-Location** command moves the pointer a specific distance along the Y-axis. Each step represents 1/100 (0.01) inch on the worksheet if the pin-to-pin spacing on the template table on the **Configure Schematic Design Tools** screen is set to the default value of 0.100 inch.

Select **Y-Location.**

Edit Library displays "Jump Y". Enter the number of steps to jump. The pointer jumps to the specified location and **Edit Library** returns to the main menu level.

➤ *Helpful hints . . .*

A number without a plus or minus sign moves the pointer to the actual grid coordinate, a signed positive number (+10, +25, +30, and so on) moves the pointer down, and a negative number (–10, –25, –30, and so on) up. If you enter +10, for example, the pointer jumps down 0.1 inch. If you enter 100, without a plus or minus sign, the pointer moves to the actual Y grid coordinate 10.0.

△ *NOTE: The coordinates used by the **JUMP X-Location** and **JUMP Y-Location** commands are dependent on the origin set with the **ORIGIN** command.*

LIBRARY

LIBRARY updates the current library, lists the part names in the library, browses through the library, deletes a part from the library, and defines prefixes.

Select **LIBRARY**. The menu shown at right displays.

Library *filename*

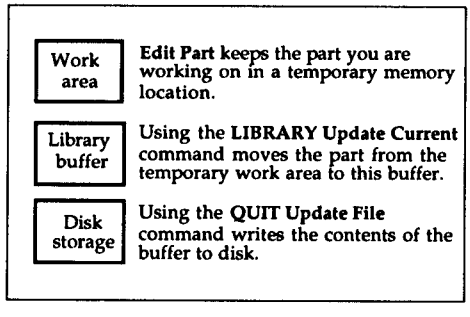
```
Update Current
List Directory
Browse
Delete Part
Prefix
```

LIBRARY Update Current

You update the library to change the definition of an existing part or to add a new part. To change the definition of an existing part, retrieve the part with the **GET PART** command, edit it, and then select **LIBRARY Update Current.**

➤ *Helpful hints . . .*

To add a new part, build the part (you may find it easier to retrieve a similar part and edit it), name the part with the **NAME** command, and select **LIBRARY Update Current.**



Updating the library with the **LIBRARY Update Current** command modifies the copy of the library in memory, *not* the copy on disk. To change the copy of the library on disk, use the **QUIT Update File** command (which saves the library on disk without renaming it) or the **QUIT Write to File** command (which saves the library on disk under a new name).

LIBRARY List Directory

List Directory lists the names of the parts in a library. The list can be displayed, printed, or saved in a file.

Select **List Directory.** The menu shown at right displays.

```
List Directory To?
Screen
File
```

Screen

Select **Screen.** The library directory displays on the screen.

File

Select **File.** The prompt "File?" displays. Enter the path and filename. The library directory is sent to a file.

➤ *Helpful hint . . .*

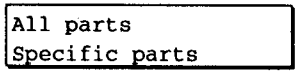
If you wish to print a library directory, use the **LIBRARY List Directory File** command to send the library directory to a file, and then use DOS's **PRINT** command to print the file. Or you can use the **List Library** tool to create the list, then DOS's **PRINT** to print it.

```
Directory of EXAMPLE.LIB
 Prefix    - 74LS     74S      74ALS    74AS     74HCT    74HC     74ACT    74AC
 Shorthand- LS        S        ALS      AS       HCT      HC       ACT      AC
 Prefix    - 74AHCT   74FCT    74F      74C      74
 Shorthand- AHCT      FCT      F        C

 00      01      02      03      04      05      06      07      08      09      10
 11      12      13      14      15      16      17      18      19      20      21
 22      24      25      26      27      28      30      32      33      34      35
 36      37      38      39      40      41      42      43      44      45      46
 47      48      49      50      51      51X     54      54X     55      56      57
 60      63      64      65      68      69      70      72      73      74      75
  .       .       .       .       .       .       .       .       .       .       .
  .       .       .       .       .       .       .       .       .       .       .
  .       .       .       .       .       .       .       .       .       .       .
 11241   11244   11245   11251   11253   11257   11258   11280   11286   11299   11323
 11352   11353   11373   11374   11378   11379   11520   11521   11533   11534   11620
 11623   11640   11643   11646   11648   11651   11652   11821   11822   11823   11824
 11825   11826   11827   11828   11833   11834   11841   11842   11843   11844   11845
 11846   11853   11854   11861   11862   11881   11882   29806   29809   29827   29828
 29861   29862   29863   29864   75188   75189
```

Figure 20-5. Partial listing of EXAMPLE.LIB.

LIBRARY Browse

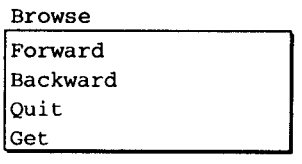
Use the **LIBRARY Browse** command to view all the parts in a library, or to select a part and view it on the screen. This command works like Draft's **LIBRARY Browse** command.



Select **Browse**. The menu shown above displays.

All parts

Select **All parts** to view parts, one at a time, starting at the beginning of the library. The menu shown at right displays.



- Select **Forward** or **Backward** to browse through the library. The parts are arranged in ascending numeric or alphabetic order.
- Select either **Quit** or **Get** to return to the main menu. The last part displayed remains on the screen, ready for you to edit.

Specific parts

Select **Specific parts** when you know the name of the part you want to view. The prompt "Part?" displays.

- If you know the name of the part you want to look at, type its name and press <Enter>. **Edit Library** gets the part and displays it on the screen.
- If you don't know the part name, press <Enter>. **Edit Library** displays a list of part names. You can then select a part from the list. To scroll through the list of parts, use the <↓> and <↑> keys or move the mouse up and down. To select the highlighted part, press <Enter> or click the left mouse button.

Once you select a part, the "Part?" prompt returns. You can display another part or a list of part names as described above. If you press <Esc> without a part name, **Edit Library** returns to the main menu level. The last part displayed remains on the screen, ready for editing.

LIBRARY Delete Part

Use **Delete Part** to delete parts from a library.

Select **Delete Part**. The following prompt displays:

```
Delete Library Part?
```

- If you know the name of the part you want to delete, type its name and press <Enter>. **Edit Library** deletes the part.
- If you don't know the part name, press <Enter>. **Edit Library** displays a list of part names. You can then select a part from the list. To scroll through the list of parts, use the <↓> and <↑> keys or move the mouse up and down. To select the highlighted part, press <Enter> or click the left mouse button. **Edit Library** deletes the part.

The **Delete Part** command only deletes a part from the library in memory. The part is not deleted from the library on disk until you select **QUIT Update File** (saves the modified library to the same file) or **QUIT Write to File** command (saves the modified library to a new file you specify).

➤ *Helpful hint . . .*

You can specify which part to delete by typing a suffix in response to the "Part?" prompt. If a part definition has multiple names (or suffixes) associated with it, this command only deletes the particular suffix you type. The part is only deleted when the last of its suffixes is deleted.

LIBRARY Prefix

Use **Prefix** to edit a library's prefix definition.

Select **Prefix. Edit Library** displays a menu, a list of prefixes, their associated short prefix, and the number of parts in the library that start with the prefix:

```
Select Index
 0          ID  Prefix    Short Prefix     In Use By
 1          0   74LS      LS                   277
 2          1   74S       S                     83
 3          2   74ALS     ALS                  241
 4          3   74AS      AS                   162
 5          4   74HCTLS   HCTLS                159
 6          5   74HCT     HCT                  121
 7          6   74HC      HC                   178
 8          7   74ACT     ACT                   89
 9          8   74AC      AC                    81
 10         9   74HCT     AHCT                 159
 11         10  74FCT     FCT                   49
 12         11  74F       F                    216
 13         12  74C       C                     71
 14         13  74PCT     PCT                   46
 15         14  74BC      BC                    30
 16         15  74                             163
            16                                   0
```

About prefix definitions

Draft uses a library's prefix definition when you get a part with the **GET** command. Instead of entering the long name of a part, you can enter just the part's short prefix.

The prefix definition is specifically designed to handle the various TTL logic families.

For example, the 74LS00, the 74S00, and the 74ALS00 have different prefixes (74LS, 74S, and 74ALS), but the same suffix (00). When you use a prefix definition, you reduce the memory needed to store multiple families of parts that have different prefixes, but the same suffix. For more information, see *Prefix definition* in *Chapter 17: About libraries*.

Up to 32 prefixes can be defined in each library.

0 through 16

Once you select **Prefix**, a menu listing the values 0 through 16 displays. These are the default values—you can specify up to 32.

To add or change a prefix or short prefix, choose the number that corresponds to the prefix or short prefix you want to add or change.

For example, in the figure on the previous page, choose **7** to change the prefix "74ACT" or the short prefix "ACT."

Once you choose a number, its corresponding line becomes highlighted and the **Prefix** and **Short Prefix** commands display in a menu, as shown below right.

Prefix and Short Prefix

Select **Prefix** or **Short Prefix**, depending on which field you want to edit. **Edit Library** displays "Prefix?" or "Shorthand Prefix?" If the prefix or shorthand prefix is already defined, it displays after this prompt.

```
Prefix Editing
Prefix
Short Prefix
```

Enter (or edit) the prefix or short prefix. **Edit Library** makes the change in the table displayed in the window.

MACRO

A macro is a set of keystrokes you assign to a particular key. You can run all the commands represented by those keystrokes by pressing just the assigned key. With the **MACRO** command, you can capture keystrokes, delete macros, write defined macros to a file, read macros from a file, list the active macro keys and erase (initialize) the macro file.

Edit Library's MACRO command works just like **Draft's MACRO** command. For more information about writing macros, see the description of the **MACRO** command in the *Chapter 2: Draft* in this guide.

To define a macro file that loads whenever **Edit Library** runs, enter the filename in the **Edit Library Macro File** entry box on the **Configure Schematic Design Tools** screen.

Initial Macro

Initial macros run automatically each time you run **Edit Library**. To define and load an initial macro, follow the instructions given in the *MACRO* section of *Chapter 2: Draft*. Enter the initial macro name in the **Edit Library Initial Macro** entry box on the **Configure Schematic Design Tools** screen.

NAME

Use **NAME** to assign a name to a part.

When you edit a part from an OrCAD-supplied library, it comes into **Edit Library** with its own name or list of names. If you update the library with the **LIBRARY Update Current** command, you overwrite the existing part.

Suppose you want, instead, to construct a new part. If the part is similar to an existing part, the best way to build it is to read in the existing part, rename the part, edit it, and *then* update the library.

A part may have a list of names. For a part to be unique, each name in the list must be unique. For example, assume you do the following:

- Retrieve a part with the names A, B, and C.
- Change one of its names (for example, C to D).
- Edit the part definition.
- Add the part (now with names A, B, and D) to the library.

The library now contains two parts: one with the names A, B, and D (the new part) and one with the names A, B, and C (the old part). Adding a new part replaces existing parts with the same name. Because the new part doesn't have the name C, one instance of the original part is kept with the name C. To delete C from the library, use the **LIBRARY Delete Part** command.

Select **NAME**. The menu shown at right displays.

Name
Add
Delete
Edit
Prefix

➤ *Helpful hint . . .*

The maximum length allowed for part names is 127 characters. However, it is recommended that you use much smaller part names. On a 640 x 480 screen at zoom scale 1, a name longer than 78 characters is clipped short. Also, many netlist formats place restrictions on the length of names. Check the name length requirements for the netlist format you are using. This information is available in *Appendix B: Netlist formats*.

NAME Add

Select **Add** to add a name to a part. The prompt "Name?" displays. Enter the name to give the part.

The prompt "Sheet Path?" displays. If the part represents a sheet, enter the name of the worksheet file it refers to. If the part does not represent a sheet, just press <Enter>. For more information about sheetpath parts, see chapter 10 in the *Schematic Design Tools 386+ User's Guide.*

NAME Delete

Select **Delete** to delete a name. All names assigned to the current part display on the screen. Select the name to delete.

NAME Edit

Select **Edit** to edit an existing name. All names assigned to the current part display on the screen.

Select the name you want to edit. The prompt "Name?" displays, followed by the current name.

Edit the name by positioning the cursor using the <←>and <→> keys and the <Home> and <End> keys. Erase characters with the <Backspace> and <Delete> keys. When you are done editing, press <Enter>.

The prompt "Sheet Path?" displays. If the part represents a sheet, enter the pathname of the worksheet file it refers to. If the part does not represent a sheet, just press <Enter>.

NAME Prefix

Select **Prefix** to determine what prefixes are defined for a part name. All names assigned to the current part display on the screen. Select the part name whose prefixes you want to define. A list of prefix strings defined for the library displays (see right). Beside each prefix string is a **Yes** or a **No**, which determines whether the prefix string is valid or invalid for the selected part name. To change the setting of a prefix string, select the prefix string. Then select **Yes** to make the string valid or **No** to make it invalid.

74LS	NO
74S	NO
74ALS	NO
74AS	NO
74HCT	NO
74HC	NO
74ACT	NO
74AC	NO
74AHCT	NO
74FCT	NO
74F	NO
74C	NO
74	NO

ORIGIN

ORIGIN sets the current pointer position as the new X,Y origin (X=0.0, Y=0.0). The default origin is the upper left of the part's body. When building a part, it is sometimes useful to redefine the origin to be at a particular point within the part outline. In this way, you can easily make relative measurements when constructing graphic parts.

PIN

Use the **PIN** command to add, delete, or edit pins.

Select **PIN. Edit Library** displays:

```
Add Delete Name Pin-Number Type Shape Move ... Repeat
```

PIN Add

To add a pin to a part, place the pointer where you want the new pin and select **Add.**

Edit Library then queries you for the Name, Pin Number, Type (Input, Output, Bidirectional, Power, Passive, 3-state, Open Collector, or Emitter), and Shape (Line, Clock, Dot, Dot Clock, or Short) of the pin you are adding.

PIN Delete

To delete a pin in a part, place the pointer on the unwanted pin and select **Delete.**

PIN Name

Use the **Name** command to edit the string definition of an existing pin. Place the pointer on the pin to name and choose **Name.**

Edit Library displays "Name?" Enter the new pin name.

To enter a name with a bar over it (indicating negation), type backslash characters after the letters. For example, type:

```
R\E\S\E\T\
```

to define the name:

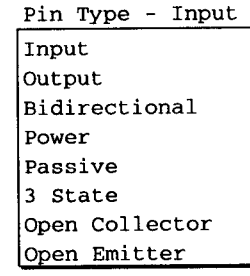
$\overline{\text{RESET}}$

PIN Pin-number

Use the **Pin-number** command to change the pin number of a pin. Place the pointer on the pin that you are assigning a number to and choose **Pin-number.**

Edit Library displays "Pin number?" Enter the new pin number.

PIN Type To change the type of a pin, move the pointer to the pin's location and select **Type**. A menu listing the available pin types appears, as shown at right. The pin's current type is shown above the menu.



Select the type you want. The available types are described below and on the next page.

Input An input pin is one to which you apply a signal. For example, pins 1 and 2 on the 74LS00 NAND gate are input pins.

Output An output pin is one to which the part applies a signal. For example, pin 3 on the 74LS00 NAND gate is an output.

Bidirectional A bidirectional pin is either an input or an output. For example, pin 2 on the 74LS245 bus transceiver is a bidirectional pin. The value at pin 1 (an input) determines the active type of pin 2 as well as others.

Power A power pin expects either supply voltage or ground. For example, on the 74LS00 NAND gate, pin 14 is VCC, and pin 7 is GND (ground). It is not a good idea to use overbars above power pins names; if you do, any netlists that you create will have invalid power pin names. Power pins are invisible

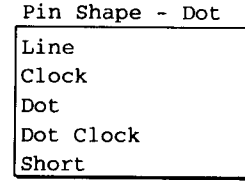
Passive A passive pin is typically connected to a passive device. A passive device does not have a source of energy. For example, a resistor lead is a passive pin.

3 State A 3-state pin has three possible states: low, high, and high impedance. When it is in its high impedance state, a state pin looks like an open circuit. For example, the 74LS373 latch has 3-state pins.

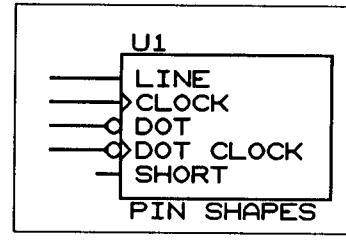
Open Collector An open collector gate omits the collector pull-up. Use an open collector to make "wire-OR" connections between the collectors of several gates and to connect with a single pull-up resistor. For example, pin 1 on the 74LS01 NAND gate is an open collector gate.

Open Emitter An open emitter gate omits the emitter pull-down. The proper resistance is added externally. ECL logic uses an open emitter gate and is analogous to an open collector gate. For example, the MC10100 has an open emitter gate.

PIN Shape To specify the shape of a pin, move the pointer to the pin location and select **Shape**. A menu listing available pin shapes appears (shown below). The pin's current shape displays above the menu. Select the shape you want.



The shape **Line** represents a normal pin with three grid unit leads. **Clock** indicates the clock symbol. **Dot** indicates the inversion bubble. **Dot Clock** represents a clock symbol with an inversion bubble. **Short** represents a pin with 1 grid unit lead.



PIN Move To move a pin, select **Move**. "Pin to Move?" displays.

Place the pointer on the pin you need to move, and click the mouse button. "New Pin Location?" displays.

Move the pointer to the new location, and click the mouse button once. **Edit Library** places the pin in the new location.

PIN Repeat

To add several pins to a part, set a few parameters, place the pointer where you want the first new pin, select **Add** to place the first pin, and then select **Repeat** for each subsequent new pin.

Edit Library increments the name, pin number, and location as you specify (using the **SET Repeat Parameters** command). **Edit Library** uses the type and shape you specify for the first pin you place with the **PIN Add** command.

QUIT

Use **QUIT** to leave **Edit Library** and return to the **Schematic Design Tools** screen. You can also write to a file, update the current library, initialize the current **Edit Library** session, and suspend to the operating system.

```
Quit LIBRARY.LIB
Update File
Write to File
Initialize
Suspend to System
Abandon Edits
```

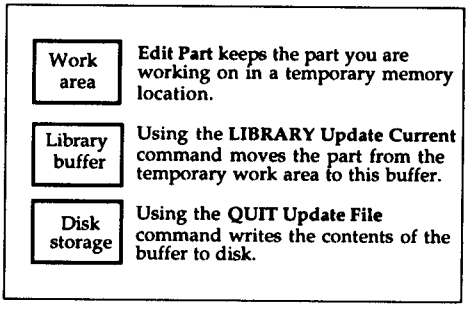
Select **QUIT**. The menu shown above displays.

QUIT Update File

The **Update File** command writes the latest copy of the library being edited to disk. This command is *not* the same as the **LIBRARY Update Current** command.

The procedure for modifying a library is as follows:

- Run **Edit Library** using the desired library. **Edit Library** then reads the library into memory.
- Get and edit a part definition, or define a new part.
- Select **LIBRARY Update Current** to modify the copy of the library in memory.



- Select **QUIT Update File** to write the copy of the library in memory to disk.

➤ *Helpful hint . . .*

QUIT Update File writes a compiled library file. It does not change any corresponding library source file. To update the library source file to reflect the changes you made with **Edit Library**, run **Decompile Library** on the new compiled library file to make a new source file.

QUIT Write to File

Write to File writes the latest copy of the library being edited to any file you specify.

Select **Write to File**. The following prompt displays:

```
Write to?
```

Enter the path and filename of the library you want to write to.

QUIT Initialize

The **Initialize** command abandons the edits since the last file update (using **QUIT Update File** or **QUIT Write to File**) and loads a new library.

Select **Initialize**. If you have made changes since the file was loaded or saved, **Edit Library** displays "Initialize – Are you sure?" Select **No** to cancel the **Initialize** command and return to the main menu level. Select **Yes** to continue.

If you select **Yes**, the current library is removed from memory, and the following prompt displays:

```
Read library?
```

If you know the name of the library containing the part you want to edit, enter its path and filename.

If you do not know the name, press <Enter> to display a list of files. All files specified in **Edit Library's Prefix/Wildcard** entry box are listed. To load a file, simply select it from this list, or to create a new library, select **New Library** at the top of the list.

QUIT Suspend to System

Suspend to System temporarily leaves **Edit Library** and returns to the operating system. Once you have suspended **Edit Library**, you may perform operating system functions, including using other software programs (as long as there is enough computer memory).

Select **Suspend to System. Edit Library** suspends operation, loads the operating system command interpreter, and adds an additional ">" to the system command prompt. This is a reminder that **Edit Library** is suspended and in the background.

To return to **Edit Library**, type **EXIT** at the operating system prompt. **Edit Library** then comes to the foreground, with the current part displayed on the screen.

QUIT Abandon Edits

Use the **Abandon Edits** command to end your **Edit Library** session and return to the **Schematic Design Tools** screen.

Select **Abandon Edits.** If you changed the library currently in memory (with the **LIBRARY Update Current** command), **Abandon Edits** asks you to confirm your decision to leave.

Select **No** to return to the main command level. Select **Yes** to exit **Edit Library.**

REFERENCE

Use **REFERENCE** to define or edit a part's reference designator.

The default reference designator is "U?". The "?" is a placeholder for the number representing the occurrence of the device. **Annotate Schematic** replaces "?" with a number indicating the instance of the part. If the part is a multiple-element part, **Annotate Schematic** adds a one-character alphabetic suffix to the reference designator and increments it A, B, C, and so on, once for each part of the package used.

Select **REFERENCE**. This prompt displays:

```
Initial Reference Designator? U
```

"U" is the current reference designator. To replace U with another letter or sequence of letters and numbers, backspace over U, type the new reference designator, and press <Enter>.

➤ *Helpful hint . . .*

If a part is zero parts per package and you delete the "?," the reference designator and part value are not shown in **Draft**. See the GND symbol in DEVICE.LIB for an example. In addition, these types of parts are not shown in the **Bill of Materials** report.

SET

Use **SET** to control the following **Edit Library** settings:

- Auto-panning mode
- Creating backup files
- Turning the error bell on or off
- Activating the left mouse button
- Macro prompting
- Displaying power pins
- Displaying a part's body
- Displaying grid dots
- Setting repeat parameters
- Selecting pointer shape

Select **SET** to change a setting. The menu shown at right appears. If you prefer settings other than the defaults, you may use an initial macro to change them automatically every time you run **Edit Library**. See the *MACRO* section in this chapter and *Chapter 1: Configure Schematic Tools* for information about initial macros.

Set

Auto Pan	YES
Backup File	YES
Error Bell	YES
Left Button	NO
Macro Prompts	YES
Power Pins Visible	NO
Show Body Outline	NO
Visible Grid Dots	NO
Repeat Parameters	
Cursor Style	

SET Auto Pan

Auto Pan controls movement past the screen boundary. While **Auto Pan** is turned on, the screen pans in the direction that the pointer moves.

Select **Auto Pan**. "PAN at the screen edge?" displays. Select **Yes** to turn on auto panning; select **No** to turn it off.

SET Backup File

Backup File controls whether or not **Edit Library** creates a backup file of the current library when you write or update files using the **QUIT** command. This is useful if you want to preserve the last version of a library in case you make a mistake and want to revert to the earlier version. The backup is given a .BAK extension. Whenever you update or write to the file with **SET Backup File** set to **Yes, Edit Library** saves the previous version of the library as the backup and the edited version as the actual file.

Select **Backup File.** "Make Backup (.BAK) file?" displays. Select **Yes** if you want backup files created; select **No** if you don't want backup files created.

SET Error Bell

Error Bell turns the error bell (your computer's speaker) on and off. When you turn this option on, error messages and errors sound the speaker.

Select **Error Bell.** "Ring Bell for errors?" displays. Select **Yes** to turn on the error bell; select **No** to turn it off.

SET Left Button

When **Left Button** is on, you can select a command line command by pressing the left mouse button, holding the button down while you highlight the desired command in the menu that displays, and then releasing the left mouse button. When **Left Button** is off, you must click the left mouse button once to display the menu and then click it again to select a command.

For example, suppose you select the **BODY <Graphic> Line** command and the "Begin Jump Origin Tag Zoom" command line displays at the top of the screen. To select **Begin** with the mouse when **SET Left Button** is disabled, you click the left button once to display a menu and once again to select **Begin**. When **SET Left Button** is turned on, you instead press the left button and hold it down while you move the highlight to **Begin**, then release the left button to select **Begin**, thus saving one button click.

Select **Left Button**. "Left Mouse Button Release does <Enter>?" displays. Select **Yes** if you want to be able to select command line commands by pressing and releasing the left mouse button. Select **No** if you want to select command line commands by clicking the left mouse button twice.

SET Macro Prompts

When **Macro Prompts** is turned on, commands making up your macros display on the screen when a macro runs. Turn **Macro Prompts** on when debugging macros or to watch the commands being performed when you run a macro.

Select **Macro Prompts**. "View Prompts during Macros?" displays. Select **Yes** if you want macro prompts to display; select **No** if you don't want them to display.

SET Power Pins Visible

When **Power Pins Visible** is set to ON, the part's power pins appear on the screen. For example, if you set **Power Pins Visible** to **Yes** and display the part 74LS00 in the TTL library, pins 14 and 7 appear. For the 74LS00, 14 is VCC and 7 is GND. Power pins may overlap existing pin names or the part name. Typically, power pins do not display. Power pins do not display in **Draft**.

Select **Power Pins Visible**. "Power Pins Visible for Editing?" displays. Select **Yes** if you want power pins to display; select **No** if you don't want them to display.

SET Show Body Outline

When **Show Body Outline** is turned on, the part's body outline appears as a dotted line. When you edit a graphic part, you must edit within the body outline or unpredictable results occur. It's a good idea to display the part's body outline unless you have a compelling reason not to.

Select **Show Body Outline**. "Show Bitmap Body Outline?" displays. Select **Yes** if you want the part's body outline to display; select **No** if you don't want it to display.

SET Visible Grid Dots

When **Visible Grid Dots** is turned on, grid dots display on the screen. The spacing of the grid dots depends on the current zoom scale.

Scale 1

The grid dots appear every grid unit.

Half scale

The grid dots still appear every grid unit but, because half scale shows twice as much detail as scale 1, the grid dots appear twice as far apart.

Quarter scale

The grid dots are placed 0.1 grid unit apart.

SET Repeat Parameters

Repeat Parameters govern the horizontal or vertical spacing, and the name and pin numbers of pins that are placed using the **Repeat** command. The **Repeat** command is available in the **Pin** menu.

X Repeat Step

X Repeat Step specifies the number of grid spaces in the X direction that one pin is offset from the previous pin. Enter a positive number to place new pins to the right of the original pin and a negative number to place new pins to the left of the original pin.

Y Repeat Step

Y Repeat Step specifies the number of grid spaces in the Y direction that one pin is offset from the previous pin. Enter a positive number to place new pins below the original pin and a negative number to place new pins above the original pin.

Name Delta

If a pin name ends with a numeric value, the **Name Delta** specifies how much it increments or decrements each time the pin is placed. For example, If the **Name Delta** is +2, and a pin is named DPIN1, each successive pin placed using the **PIN Repeat** command is named DPIN3, DPIN5, and so on.

Pin Number Delta

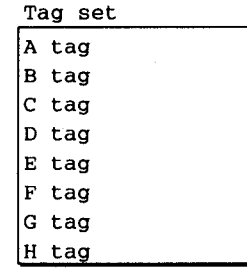
Pin Number Delta specifies how much a pin number increments or decrements each time a pin is placed using the **PIN Repeat** command.

SET Cursor Style

Cursor Style allows you to choose between two cursor styles: a simple pointer, or a pointer attached to a cross. The cross is especially useful when you align objects on the screen.

TAG

The **TAG** command identifies and remembers locations on **Edit Library's** screen. You can specify eight locations (A through H) using **TAG**. Tagged locations can be used as destinations when you use the **JUMP** command to quickly move the pointer to pre-defined locations. Tags do not appear on the part display, and they are not saved with the part.



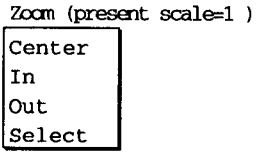
To set a tag, place the pointer at a location you want to remember, and select **TAG**. **Edit Library** displays the menu shown above.

With the pointer in the location where you want the tag assigned, choose one of the tag commands. **Edit Library** remembers the tag location.

ZOOM

ZOOM zooms in or out from the part display, changing the amount of detail you see. You can zoom in and out to three different levels: **1** (the default), **Half**, and **Quarter**. **Half** shows more detail than **1**, and **Quarter** shows more detail still.

Select **ZOOM**. The menu shown at right appears.



The number or word in parentheses after **Center** shows the current zoom scale.

ZOOM Center Choose **Center** to centers the displayed portion of the part around the pointer. This command is useful for centering a part on the screen for easy editing.

For example, if a part displays partially off the screen, you may center it by placing the pointer near the part's center and selecting **ZOOM Center**.

ZOOM In Select **In** to select the next more detailed scale (the part appears larger).

ZOOM Out Select **Out** to select the next less detailed scale (the part appears smaller).

ZOOM Select Select **Select** to choose a specific zoom level.

ZOOM Select 1 Select **1** to show the part in normal scale.

ZOOM Select Half Select **Half** to show the part in two times the drawing resolution.

ZOOM Select Quarter Select **Quarter** to show the part in four times the drawing resolution.



Decompile Library

Decompile Library takes a compiled library file and produces a library source file.

You can then edit the library source file, and use **Compile Library** to make a compiled library file. Think of **Decompile Library** as the inverse of **Compile Library**.

Execution

Select **Decompile Library** from the **Schematic Design Tools** screen. Select **Execute** from the menu that displays.

Decompile Library creates a library source file from a compiled library file.

When **Decompile Library** completes its task, the **Schematic Design Tools** screen appears.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Decompile Library**. Select **Local Configuration** from the menu that displays.

Select **Configure DECOMP**. A configuration screen appears (figure 21-1).

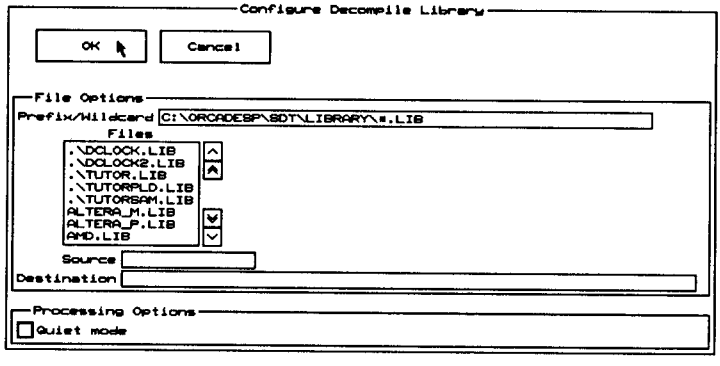


*Figure 21-1. **Decompile Library**'s local configuration screen.*

File Options — **File Options** names the library to decompile and the library source file to produce.

Prefix/Wildcard — Enter a pathname and wildcard to define which files to display in the list box with scroll buttons. The asterisk character (*) is used as a wildcard. The default is:

```
\ORCADESP\SDT\LIBRARY\*.LIB
```

If you erase the entire field, the prefix specified on the **Configure Schematic Design Tools** screen is restored.

Files — The files that match the search filter entered in the **Prefix/Wildcard** entry box and those that match the filter in the current design directory display in this box. Files in the current directory are shown with .\ before their names. Use the scroll buttons to scroll the list of libraries up and down.

Select the file to decompile by clicking on its name. Its path and filename display in the **Source** entry box.

Source

The **Source** names an existing compiled library file.

Specify **Source** by selecting a name from the **Files** list box, or enter a name by simply typing it in this entry box and pressing <Enter>.

Destination

The **Destination** is the full path and filename of the library source file that **Decompile Library** produces. It is a text file that describes the parts in the specified library. If you give the name of an existing file, **Decompile Library** asks if you want to overwrite the existing file. You cannot append to an existing file.

The **Destination** can be a complete pathname.

Processing Options

If desired, select the following option:

- ❑ `Quiet mode`

  Turns quiet mode on.



Creating a library source file with a text editor

Library source file

A library source file consists of the following:

- A prefix definition. You can only have one prefix definition per library, and it occurs at the beginning of the library.
- A series of part definitions. There are three types of part definitions: Block, Graphic, and IEEE.

Block part definitions

Block part definitions represent parts that are either square or rectangular. These parts are typically memory chips, microprocessors, peripheral controllers, and many TTL and CMOS devices.

Graphic part definitions

Graphic part definitions are used for small parts that have curved perimeters or whose function can be suggested by internal lines, circles, arcs, and fills. They include such parts as resistors, diodes, transistors, MOSFETS, relays, and many others. A graphic part can be up to 25.5 inches by 20.0 inches in size.

IEEE part definitions

IEEE part definitions represent parts that follow ANSI/IEEE standard. An IEEE part is similar to a graphic part, but it cannot contain arcs or fills. An IEEE part can be up to 25.5 inches by 20.0 inches.

△ *NOTE: For more information, see* ANSI/IEEE Std 91-1984: IEEE Standard Graphic Symbols for Logic Functions, *©1984, and* Graphic Symbols for Electrical and Electronics Diagrams, *©1975; both published by The Institute of Electrical and Electronics Engineers, Inc.*

Prefix Definition

The prefix definition is specifically designed to handle the various TTL logic families. For example, the 74LS00, the 74S00, and the 74ALS00 have different prefixes (74LS, 74S, and 74ALS), but the same suffix (00). When you use a prefix definition, you reduce the memory required to store multiple families of parts with different prefixes, but the same suffix.

All source files *must* begin with a prefix definition. If you do not want a prefix definition in your custom library, you must still supply a null prefix (see example 2 in the next section, *Constructing a prefix definition*).

Use of the prefix definition

Draft uses the prefix definition when you obtain a part with the **GET** command. Instead of entering the entire name of the part, you can enter just the suffix. **Draft** displays a menu listing all the valid part names constructed by appending the suffix you provided with the prefixes in the prefix definition. For example, if TTL.LIB is one of your libraries and you enter the suffix 04, the menu lists the parts shown in figure 22-1.

```
Get? 04
74LS04
74S04
74ASL04
74AS04
74HCT04
74HC04
74F04
7404
```

Figure 22-1. Valid part names displayed by entering the suffix "04."

Constructing a prefix definition

You can construct a prefix definition in a text editor as follows:

1. Enter the PREFIX keyword.
2. Begin the first prefix string by typing a single quote <'>. Type the prefix string. It consists of a string of text characters no more than seven characters long. **Draft** does not distinguish between upper- and lower-case. Close a prefix string with another single quote.
3. Type an equal sign <=>. To improve readability, you can enter any number of space or <Tab> characters before and after the equal sign.
4. Type the shorthand string. The shorthand string consists of no more than seven text characters. After you have entered the shorthand string, press <Enter> and type the next line. You can define a maximum of sixteen unique prefix strings.

   The shorthand string is used to bypass the prefix menu and still enter an abbreviated part name. For example, you can obtain the part 74HC04, by supplying the **GET** command with the abbreviated name HC04. This is possible because HC is a shorthand string for 74HC.
5. Close the prefix definition by typing the keyword END alone on a line, followed by <Enter>.

Example 1 This example below shows the prefix definition in the TTL.LIB library:

```
PREFIX
'74LS' = 'LS'
'74S' = 'S'
'74ALS' = 'ALS'
'74AS' = 'AS'
'74HCT' = 'HCT'
'74HC' = 'HC'
'74ACT' = 'ACT'
'74AC' = 'AC'
'74AHCT' = 'AHCT'
'74FCT' = 'FCT'
'74F' = 'F'
'74C' = 'C'
'74'
END
```

△ ***NOTE:*** *To view the prefix definition of a library file, use* ***Decompile Library*** *to convert the file to a library source file. See* Chapter 21: Decompile Library *for details.*

Example 2 This is an example of a null prefix:

```
PREFIX
END
```

If you do not want prefixes in your library source file, you must use a null prefix at the beginning of the file.

Part definition

The part definition defines the following characteristics of a part:

- Part name
- Part size (in grid unit lengths on the screen and in tenths of an inch on the printed worksheet, unless the template table changes during configuration)
- Number of parts per package
- Pin functions (input, output, open collector, etc.).

Three types of part definitions

There are three types of part definitions: block symbol definitions, graphic definitions, and IEEE definitions. You don't have to group your block definitions, graphic definitions, and IEEE definitions together. For example, your source file may contain a block definition, followed by a graphic definition, followed by another block definition.

Block, graphic, and IEEE definitions follow much the same syntax. A graphic definition looks like a block or IEEE definition followed by a vector definition.

Components of a part definition

A part definition has the following fields:

- One or more part name strings. A name is a text string enclosed in single quotes. If you have more than one part name string, delimit them with a blank space or put them on separate lines. When obtaining a part, you can use any of the name strings.
- An optional sheet path designator. You can define a schematic file as a library part. This feature is useful for frequently used circuits.
- An optional reference designator. **Annotate Schematic** automatically updates reference designators.

- ❖ Symbol size. Each unit represents a grid unit length on the screen and 0.1 inch on the printed worksheet (unless the pin-to-pin spacing was changed in the template table during configuration). The X size is given first, followed by the Y size. Also included on this line is one of the following:
  - The number of parts per package
  - The keyword GRIDARRAY if the part is a pin grid array
  - The keyword IEEE if the part is an IEEE part
- ❖ Pin definition. Each pin is defined on a separate line. A pin definition consists of the following fields:
  - Pin position.
  - Pin numbers (zero or more, as reflected by the number of parts in the package) or GRIDARRAY pin name enclosed in single quotes.
  - Optional keywords:

    DOT Puts an inversion bubble at the pin position.

    CLK Puts a clock symbol at the pin position.

    SHORT Puts leads one grid unit long at the pin position (instead of standard three-grid-unit leads).

    If DOT and CLK are used together, DOT must come first. SHORT cannot be used with DOT or CLK.
  - Pin function (*input, output, input/output, open collector, open emitter, power, passive, hi-z*).
  - Pin name string enclosed in single quotes.

An example of a pin definition is shown in figure 22-2.

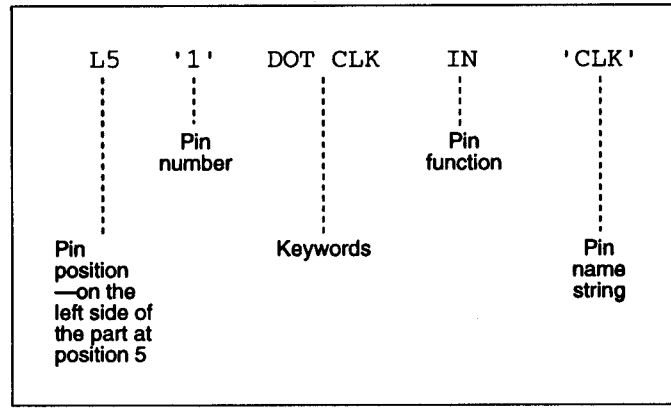


Figure 22-2. Pin definition line.

For specific details about defining a pin, see *Chapter 23: Symbol Description Language.*

- An optional bitmap definition. Bitmaps are retained for compatibility with earlier version of **Schematic Design Tools** and are ignored by **Schematic Design Tools 386+.**
- A vector definition. This describes a graphic part in vector format. All the tools that deal with graphics use these vector definitions for screen display and plotting.
- An optional converted-form bitmap definition. Bitmaps are retained for compatibility with earlier version of **Schematic Design Tools** and are ignored by **Schematic Design Tools 386+.**
- An optional converted-form vector definition. This describes the converted form of a graphic part in vector format.

Defining a block symbol

Figure 22-3 is an example of a block symbol definition. The example does not represent a real part, although it is similar to a JK flip-flop. Figure 22-4 shows the symbol produced by this block definition.

Part name string ----------
Reference designator ----------
Grid unit size *(6 x 10) and* ----------
number of parts per package *(2)*

Pin definitions ----------
Column 1 gives the pin location (i.e. L1=left side, position 1; L5=left side, position 5; L0=the top of the left side, etc). Column 2 gives the pin's number in the first part in the package (i.e. L1=3, L5=1, etc.). Column 3 gives the pin's number in the second part in the package (i.e. L1=11, L5=13, etc.) The fourth column gives an optional keyword (DOT, CLOCK, or SHORT), and the pin's function. The last column gives the pin name string.

```
'74EXAMP'
REFERENCE    'LATCH'
6      10     2

L1     3      11      SHORT IN     'J'
L5     1      13      DOT CLK IN   'CLK'
L9     2      12      SHORT IN     'K'
B3     15     14      DOT IN       'CL'
T3     4      10      DOT IN       'P'
R1     6      7       OUT          'Q'
R9     5      9       OUT          'Q\'
T0     16     16      PWR          'VCC'
B0     8      8       PWR          'GND'
```

Figure 22-3. Block symbol definition for 74EXAMP.

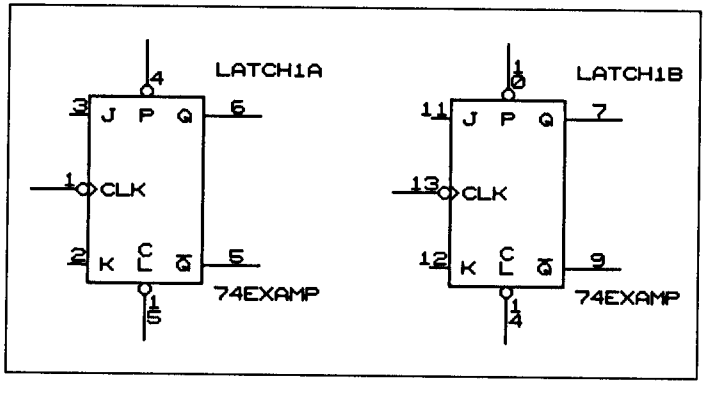


Figure 22-4. The block symbol for 74EXAMP.

Note the components making up the part definition. The first line is the part name string, 74EXAMP. The next line is the reference designator, LATCH. The third line represents the grid unit size and the number of parts per package. The remaining lines, starting with line four, are the pin definitions.

Part name string

The example on the previous page has only one part name string, 74EXAMP. If you want to refer to the same part with different names you can list a number of part names. For example, to use the same part definition for the 8031, 80C31, 8051, 80C51, and 8751 components, put all five part name strings at the beginning of the part definition, as shown in figure 22-5.

```
'8031'
'80C31'
'8051'
'80C51'
'8751'
{X Size =} 13 {Y Size =} 25 {Parts per Package =} 1
L1      31          IN  'E\A\/VP'
L3      19          IN  'X1'
L6      18          IN  'X2'
        .
        .
        .
```

Figure 22-5. Block symbol definition with multiple part names.

Notice that comments are enclosed in brackets, as in figure 22-5. The X and Y coordinates are preceded by comments, as is the number of parts per package.

Sheetpath keyword

The example in figure 22-3 does not have a SHEETPATH keyword. If you need to have a part reference a schematic sheet, place the following line after the part name strings:

SHEETPATH '*schematic_filename*'

where *schematic_filename* is the name of a schematic you have already drawn. Wherever you place a sheet-path part on your design, you actually place a reference to this schematic. You can descend into sheetpath parts using the **QUIT Enter** command. Use sheetpath parts to represent frequently used circuits.

Normally, the filename does not include a pathname. If it doesn't, **Draft** looks for the schematic in the current library directory. If the sheetpath references a sub-circuit that is common to many designs, you can include a full pathname along with the filename.

Reference keyword

If a reference designator is used, it comes after the part name string (and sheetpath designator, if there is one).

In the example in figure 22-3, the reference designator appears as "LATCH." Reference designators may also end in a question mark (such as "R?"). When you run **Annotate Schematic** on a design containing such a part, the question mark in the schematic is replaced with a number. For example, if the reference designator for a resistor is R? and there are sixteen resistors in your design, **Annotate Schematic** changes the designators to R1, R2, . . . R16. You can also manually replace the question mark when you place the part in a worksheet with **Draft**.

This example has more than one part per package, so the reference designator appears with an A after the question mark. **Annotate Schematic** then sequences the letters. **Annotate Schematic** converts the A of the second part into a B. For example, after running **Annotate Schematic**, the first two occurrences of the 74EXAMP appear as LATCH1A and LATCH1B, the next two as LATCH2A and LATCH2B, etc.

If you omit the REFERENCE line, the default designator U?A appears on the schematic.

What appears for the reference designator when you get a part is determined as follows:

1. If the device has zero parts per package and you do not specify a REFERENCE keyword, no reference or part name appear.
2. If the device has zero parts per package and you specify a REFERENCE keyword, the reference appears. It consists of the string you specified followed by a question mark. The part name also appears.

△ *NOTE: If, for some reason, you want a part without pin numbers, use a device with zero parts per package.*

3. If the device has one or more parts per package, and you do not specify a REFERENCE keyword, a default reference designator (U?A) appears. The part name also appears.

4. If the device has one or more parts per package, and you specify a REFERENCE keyword, the reference appears. It consists of the string you specified followed by ?A (assigned and displayed by **Draft**). The part name also appears.

Table 22-1 shows the relationship between the number of parts per package and the reference designator appearing on the part. You must specify a part name; the name is what **Draft** uses to get a device from the library. The REFERENCE keyword is optional, however. If the REFERENCE keyword is not specified, it defaults to U.

Source File . . .	*Number of Parts Per Package*	
	0	*1 or more*
Does not use "REFERENCE" keyword	Nothing is displayed	"U" is default reference designator
Uses "REFERENCE" keyword	Uses reference keyword you enter in source library	Uses reference keyword you enter in source library

Table 22-1. Controlling display of reference designators.

Grid unit size and parts/package

The next line in figure 22-3 contains the three numbers 6 10 2. The first two numbers (6 and 10) represent the size. The size of the part is 6X by 10Y, where each unit represents one screen unit or 0.1 inch on the printed worksheet (if the pin-to-pin spacing in the **Template Table** section of the **Configure Schematic Design Tools** screen is 0.1 inch). The third number (2) indicates there are two parts per package. If the part is a pin-grid array, supply the keyword GRIDARRAY in place of the number of parts per package. If the part is an IEEE part, supply the keyword IEEE in place of the number of parts per package.

Pin definitions

The remaining lines in the example in figure 22-3 consist of the pin definitions. Consider the second pin position as an example.

The first field L5 locates the pin on the left side of the part in the fifth position counting down from the top. The previous line in figure 22-3 defined the Y dimension as 10. This means that there are 11 possible vertical positions, 0 through 10. The first possible position on the left side of the part is L0 and the last is L10. Likewise, the right side of the part contains positions R0 through R10. The specified pin position (L5) is on the left side of the part, 5 grid units from the top of the part.

The next two fields for pin position L5 identify the pin numbers. L5 is numbered 1 on the first part of the package and 13 on the second part of the package.

The fourth field for pin position L5 specifies DOT to obtain the inversion bubble and CLK to get the clock symbol. In this case, DOT and CLK are modifiers of the pin function, IN.

Finally, the last field for pin position L5 gives the pin a name of "CLK."

Refer back to figure 22-4 and locate pin L5 on the two parts in the package. They are both labeled "CLK."

As further examples, consider R9 and B3. R9 puts a pin on the right side in the ninth position, and B3 puts a pin on the bottom in the third position counting from the left. The two power supply connections are at the top and bottom in the zero position.

Note the vertices of a part have two possible representations. For example, L0 and T0 specify the same pin location (the upper left-hand corner).

▲ *CAUTION: If you place a power pin at T0 and another power pin at L0, those pins are electrically connected.*

Figure 22-6 shows a grid representing the possible pin positions for the 74EXAMP library part.

Note the SHORT keyword in the L1 pin position in figure 22-3. Pins with the SHORT keyword have one-grid-unit leads rather than the default three grid units. The SHORT keyword, however, cannot be used with DOT or CLK keywords.

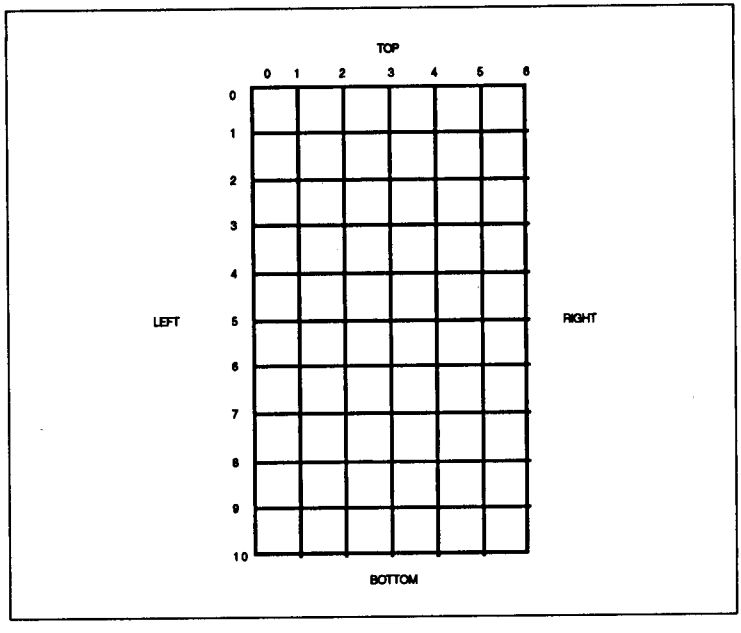


Figure 22-6. The grid of a 6X by 10Y block symbol.

Pin type

One possible pin type is PWR for power. Power pins do not appear on the screen in **Draft** (you can configure **Edit Library** to display them using the **SET Power Pins Visible** command). The connectivity database, however, does categorize all power pins connected to library parts.

If you wish to make parts with visible power pins, use the IN or PAS pin type instead of PWR. Remember to position the pin name (if it is a block part) so that it does not overlap other pin names.

Selectively displaying pins

If the device has more than one part per package, you can selectively display the pins. For example, assume you want to display the pins VCC and GND, but only on the second part of the device, not on the first. You can do this by coding the last two lines of the block symbol as follows:

```
T0  0       16      PAS  'VCC'
B0  0       8       PAS  'GND'
```

When you place this symbol on the screen, the passive pins do not display because the first column of pin locations contains a 0. When you place another symbol on the screen, it looks identical to the first. Both are called FF?A, and neither shows the power pins.

However, if you use **Draft's EDIT** command to change the reference designators to FF1A and FF1B, the power pins appear only on the second part of the device, FF1B. You can also use **Annotate Schematic** to change the reference designators.

This technique also works for other types of pins. By specifying a pin number of 0, you can cause a pin not to appear for the part of a package. But if your device has one part per package, specifying a pin number of 0 does not prevent the pin from appearing. The pin appears with a pin number of 0.

△ *NOTE: If you want a part without pin numbers, use a device with zero parts per package.*

If your device has zero parts per package, you do not specify pin numbers. Figures 22-7 and 22-8 illustrate how the number of parts per package, the pin number, and the **Annotate Schematic** tool affect the screen symbol.

The device 74ONE is identical to 74EXAMP, except it has only one part per package. Note the **Annotate Schematic** tool affects both the pinout and reference designator for 74EXAMP, but only the reference designator for 74ONE.

In both 74ONE and 74EXAMP the locations of the power pins were moved from T0 and B0 to R3 and R7, so they would not overlap existing pins.

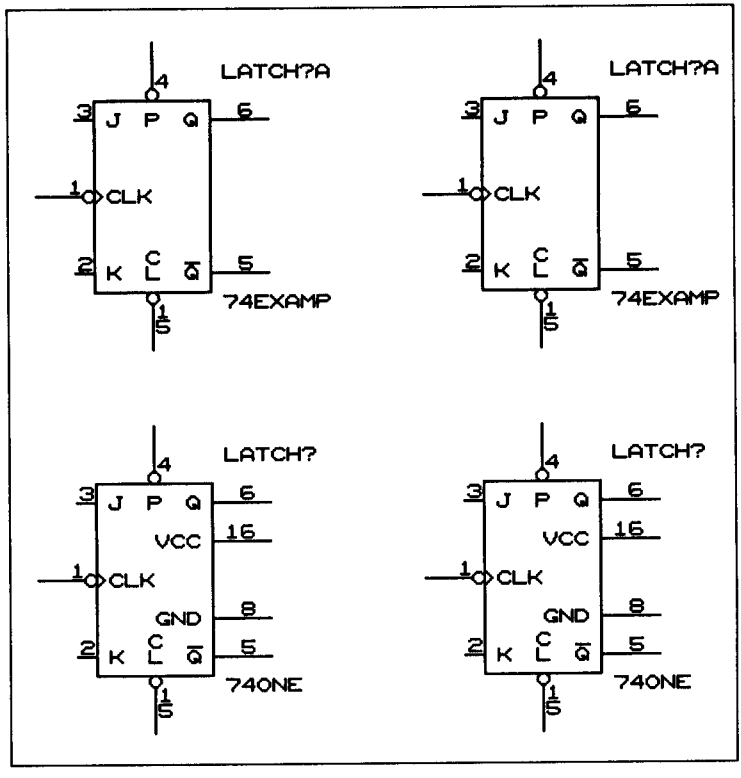


Figure 22-7. Before annotation.

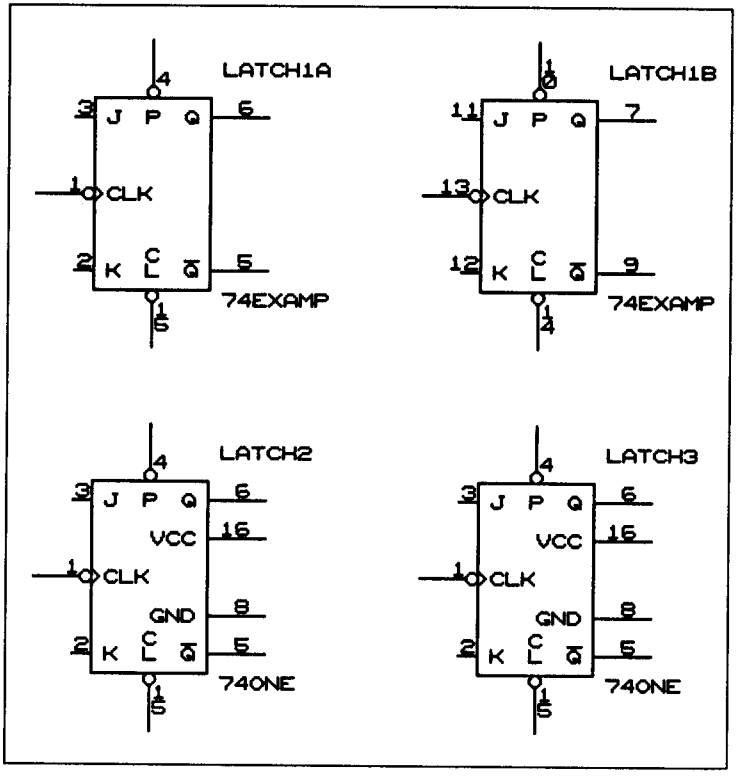


Figure 22-8. After annotation.

Pin-grid array

If the part is a pin-grid array, you supply the pin-grid array pin name instead of the pin number. A pin-grid array pin name consists of any string of up to 15 alphanumeric characters enclosed in single quotes.

Figure 22-9 is an example of a pin-grid array part definition. The example is the 68020 from the Motorola library, MOTO.LIB. The definition is quite long so only a few lines are shown. Notice the keyword GRIDARRAY on the second line, and the pin names in the second field on each pin definition line.

Figure 22-10 shows the resulting screen image.

```
'68020'
{X Size =} 16  {Y Size =} 53  {Part is a} GRIDARRAY
L1    'C2'     CLK      IN   'CLK'
L3    'J12'    DOT      IN   'IPL0'
L4    'J13'    DOT      IN   'IPL1'
L5    'H12'    DOT      IN   'IPL2'
L7    'H2'     DOT      IN   'AVEC'
L8    'A1'     DOT      IN   'BGACK'
L9    'B3'     DOT      IN   'BR'
L10   'J2'     DOT      IN   'BERR'
                .
                .
                .
```

Figure 22-9. Pin-grid array part definition.

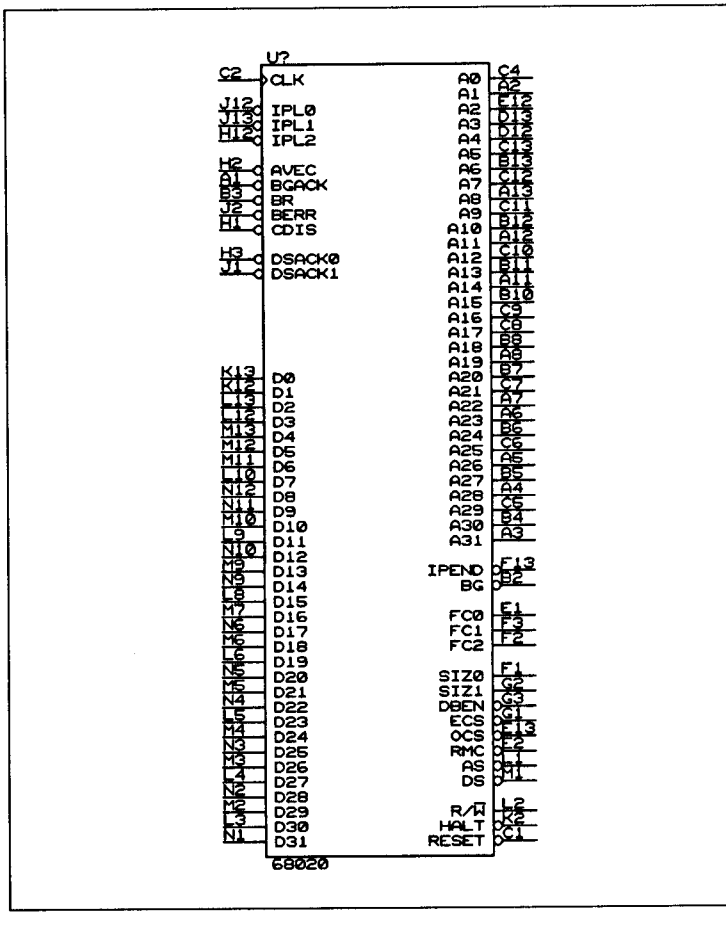


Figure 22-10. The block symbol for the 68020 defined in figure 22-9.

Pin string The pin string is delimited by single quotes. If you want a single quote as part of the pin string, you must use two single quotes. For example, "'CLK''s'" defines the string "CLK's." Also, a backslash after the pin string name puts a bar over the name. "'Q\'" produces a Q with a bar over it ($\overline{Q}$). If you have a multi-letter pin name, you must put a \ after each letter. For example, if you wanted the pin name IPL0 to have a bar above all four letters, the corresponding pin string entry in the part definition would be "'I\P\L\0\'."

Defining a graphic symbol

A graphic symbol definition is composed of a vector definition. A vector definition specifies the shape of the part in vector format.

Defining a vector

A vector definition is a set of circle, arc, line, text, and fill specifications that define the shape of the part in vector form.

Each line in the vector definition specifies one graphical piece of the part. For example, the following lines define an AND gate:

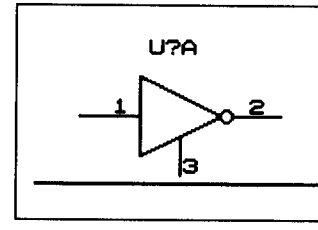
```
VECTOR
LINE    +4.0 +0.0 +0.0 +0.0  {Top}
LINE    +0.0 +0.0 +0.0 +4.0  {Left}
LINE    +0.0 +4.0 +4.0 +4.0  {Bottom}
ARC     +4.0 +2.0 +0.0 -2.0 +2.0 +0.0 +2.0
ARC     +4.0 +2.0 +2.0 +0.0 +0.0 +2.0 +2.0
END
```

The first line identifies the beginning of the vector definition. The next three lines specify the beginning and ending points of the top, left, and bottom lines in the AND gate. The next two lines define the two arcs making up the half circle on the right side of the part. Finally, the last line denotes the end of the vector definition. For details on defining vectors, see *Chapter 23: Symbol Description Language.*

Graphic symbol considerations

There are a few points you should keep in mind when creating a graphic symbol, as opposed to a block symbol.

1. You have to pay close attention to pin placement. For example, if you wanted to build the part shown at right, you would need to be very careful placing pin 3.



   The reason for this is that the pin will be placed on the part boundary. This means that your custom part will have blank space between itself and the body of the part. You will probably want a graphic line between the part boundary side of the pin and the part body.

   You will probably prefer to build this custom part in **Edit Library**.

△ *NOTE: To make your custom part look nice, place a short pin on the part boundary as you usually would. Then, draw a graphic line from the pin to the line you need to connect it to.*

2. Although you must put a pin name in the pin definition, the pin name does not appear on the screen. The pin name is, however, recognized connectivity database.

3. A graphic symbol can include a converted form of the symbol. Graphic devices always have a normal form; optionally, they can have a converted form. Usually (but not always), the converted form is the DeMorgan equivalent symbol of the normal form.

   When you use the **GET** command in **Draft**, and extract a graphic part from a library, it appears in normal form. From the menu that displays, you can also choose its converted form instead. You define what the converted form is when you create the library source file.

   Block and IEEE symbols cannot have converted forms.

4. **Edit Library** and **Plot Schematic** use vector definitions. So, although they are optional, you won't be able to edit the part with **Edit Library** or plot it with **Plot Schematic** if you don't include vector definitions.

Converted form graphic symbol

After defining a graphic symbol, you have the option of defining a converted form graphic symbol. As stated previously, the typical use of a converted form graphic symbol is to supply a DeMorgan equivalent symbol, but more generally a converted form specifies another graphic symbol to display on the screen whenever you choose the **Convert** option of the **GET** command. You can return to the original graphic symbol by selecting **Normal**.

Begin the specification of a converted form graphic symbol with the keyword CONVERT. The converted form graphic symbol consists of pin definitions followed by a vector definition. If the converted form is already defined, you can reference it by including the name of the part with the converted form in single quotes.

Here's a more detailed example showing how to use the converted form graphic symbols. The definition of the 7400 is shown first. Then, the converted form graphic symbol—the DeMorgan equivalent of the 7400—is shown. Figure 22-11 shows both the normal and converted symbols resulting from this part definition.

The 7400 has five pins, two of which are power pins not appearing in the symbol. The screen size is 6 X-units and 4 Y-units. It has four parts per package.

The converted form graphic symbol uses the same XY size and parts per package as the normal graphic symbol. You must, however, redefine the pin types. Note the DOT keyword missing from the redefinition of the pin at R2.

Also note that the converted form graphic symbol has the same number of parts per package as the normal graphic symbol. The number of parts per package determines how many columns of pin numbers appear in the definition. The converted definition must have the same number of columns as the normal definition.

```
'7400'
{X Size =)   6       {Y Size =}   4       {Parts per Package =)   4
L1       1   4   9  12           IN  'I0'
L3       2   5  10  13           IN  'I1'
R2       3   6   8  11 DOT       OUT 'O'
T0      14  14  14  14           PWR 'VCC'
B0       7   7   7   7           PWR 'GND'

VECTOR
LINE     +4.0 +0.0 +0.0 +0.0    {Top}
LINE     +0.0 +0.0 +0.0 +4.0    {Left}
LINE     +0.0 +4.0 +4.0 +4.0    {Bottom}
ARC      +4.0 +2.0 +0.0 -2.0 +2.0 +0.0 +2.0
ARC      +4.0 +2.0 +2.0 +0.0 +0.0 +2.0 +2.0
END

CONVERT
L1       1   4   9  12           IN  'I0'
L3       2   5  10  13           IN  'I1'
R2       3   6   8  11           OUT 'O'
T0      14  14  14  14           PWR 'VCC'
B0       7   7   7   7           PWR 'GND'

VECTOR
LINE     +2.5 +0.0 +0.0 +0.0
LINE     +2.5 +4.0 +0.0 +4.0
ARC      +2.5 +4.0 +3.5 -2.0 +0.0 -4.0 +4.0
ARC      -2.0 +2.0 +2.0 -2.0 +2.8 +0.0 +2.8
ARC      -2.0 +2.0 +2.8 +0.0 +2.0 +2.0 +2.8
ARC      +2.5 +0.0 +3.5 +2.0 +0.0 +4.0 +4.0
CIRCLE   +0.3 +3.0 +0.3
CIRCLE   +0.3 +1.0 +0.3
END
```

Figure 22-11. The 7400 symbol.

Defining an IEEE symbol

An IEEE symbol is composed of the following parts: a part name string, a size and type definition, a pin definition, and a vector definition.

Part name string

The part name string is defined the same as for block and graphic symbols. To refer to the same part with different names, you can list multiple part names as shown in figure 22-12.

```
'74ALS114'
'74ALS114A'
'74AS114'
'74F114'
'74HC114'
'74LS114'
'74S114'
{X Size =}  8    {Y Size =}  12     {Part is a} IEEE
T6   14          PWR   'VCC'
B6    7          PWR   'GND'
L1    1           IN   'C\L\R\'
L2   13      CLK IN   'CLK'
L5    4           IN   '1P\R\E\'
L6    3           IN   '1J'
L7    2           IN   '1K'
L9   10           IN   '2P\R\E\'
L10  11           IN   '2J'
L11  12           IN   '2K'
R6    5          OUT   '1Q'
R7    6          OUT   '1Q\'
R10   9          OUT   '2Q'
R11   8          OUT   '2Q\'
VECTOR
LINE     +0.0 +0.0 +0.0 +3.0
LINE     +0.0 +3.0 +1.0 +3.0
LINE     +1.0 +3.0 +1.0 +4.0
LINE     +0.0 +4.0 +0.0 +12.0
LINE     +0.0 +12.0 +8.0 +12.0
LINE     +8.0 +12.0 +8.0 +4.0
LINE     +8.0 +4.0 +0.0 +4.0
LINE     +7.0 +4.0 +7.0 +3.0
LINE     +7.0 +3.0 +8.0 +3.0
LINE     +8.0 +3.0 +8.0 +0.0
LINE     +8.0 +0.0 +0.0 +0.0
LINE     +0.0 +8.0 +8.0 +8.0
TEXT     -1.2 +1.0  1   ACTIVE_LOW_L
TEXT     -1.2 +2.0  1   ACTIVE_LOW_L
TEXT     -1.2 +5.0  1   ACTIVE_LOW_L
TEXT     -1.2 +9.0  1   ACTIVE_LOW_L
TEXT     +8.0 +7.0  1   ACTIVE_LOW_L
TEXT     +8.0 +11.0  1  ACTIVE_LOW_L
END
```

Figure 22-12. IEEE symbol definition for 74114 parts.

Size and type definitions

The line following '74S114' in figure 22-12 contains the numbers 8 and 12, followed by the letters IEEE. The comments inside the braces (see figure 22-12) are the X Size, the Y Size, and the part's type.

The keyword IEEE means the part is an IEEE-type symbol and, therefore, has one part per package and does not have a convert.

Pin definitions

Following the size and type definitions is the pin definitions section.

For example, look at the fourth position. The first field, L2, locates the pin on the left side of the part in the second position from the top. The next field defines the pin number: pin 13. The third field describes the function of the pin and defines how it will be drawn. "CLK IN" defines an input pin drawn with the clock symbol (>).

Finally, the last field for pin position L2 names the pin "CLK."

Vector definitions

Vector definitions are similar to those of a graphic symbol with the following exceptions:

- The allowed vector types include only: CIRCLE, LINE, and TEXT. ARC and FILL are not allowed.
- You have more freedom when drawing the vector components. In particular, some components must be placed outside the part body. For example, an input ACTIVE_LOW must be placed to the left of the part body, as shown by pins 1, 4, 10, and 13 in figure 22-13. Inputs placed to the left of the part body have a negative X offset in the vector definition. See figure 22-12.

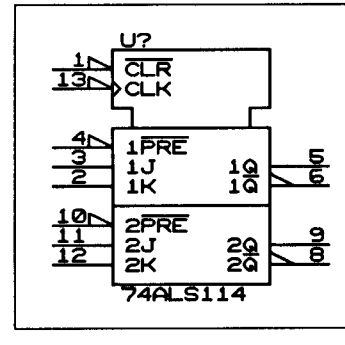


Figure 22-13. The 74ALS114 IEEE part.

Defining a vector

A vector definition for an IEEE part is a set of circle, line, and text specifications that define the shape of the part in vector form. Each line in the vector definition specifies one piece of the part's body.

The first line of the vector definition contains the keyword "VECTOR." In figure 22-3, the next twelve lines—beginning with the keyword "LINE"—specify the starting and ending points of all the lines in the body of the part.

The nine lines beginning with the keyword "TEXT" specify the locations of the text on the part. Finally, the last line denotes the end of the vector definition. For details on vectors, see *Chapter 23: Symbol Description Language.*

△ *NOTE: **Plot Schematic** draws IEEE parts using Vector commands in the order they are defined in the vector definition of the IEEE symbol definition. You can minimize plotter pen travel and Up-Down commands by careful ordering. This can save quite a bit of plotting time.*

IEEE standards

IEEE symbols are designed to create parts that meet the IEEE drawing standards in *ANSI/IEEE Std 91-1984.* Some of the requirements of this standard are not entirely intuitive. It is an excellent idea to study the standards before beginning any major effort to draw a new set of IEEE parts.

Size

IEEE symbol dimensions can be up to 25.5 by 20.0 inches. The symbols can contain many internal vector objects. All but the very largest ASIC devices can be drawn as IEEE parts.

In some cases, though, it may be more practical to use SHEET symbols on actual designs. In particular, if an ASIC device has a large number of signals which can be represented as buses, it may be easier and clearer to show the device with a small number of bus connections and a reasonable number of control signals.

Pin placement

In general, the IEEE standard wants pins placed only on the left and right side of parts: inputs on the left, and outputs on the right. **Schematic Design Tools 386+** does allow you to place pins on the top and bottom, however, to support ASIC devices with more than 255 pins. See *Size* above for information on the use of SHEET symbols for handling ASIC devices.

Building the IEEE body outline

IEEE symbols usually have a rectangular outline. They often consist of a control section on top of a main signal section. There are usually indentations in the outline near the bottom of the control section. Normally the indentations are 0.1 inch high. Figure 22-13 shows an example.

Defining this visual aspect cannot be done automatically. You have to specify the outline with LINE commands.

IEEE Vector Objects

Use TEXT commands to place IEEE objects such as those shown at right. The objects are drawn to be eight units high when used with a size argument of one. This is the same height as text of size one. It also fits between pins which are spaced ten units apart.

```
Negation
Active_Low
Arrow
BiDirectional
Dynamic
Non_Logic
Analog
3 State
Amplified
Passive
Open Circuit
Postponed
Hysteresis
Generator
Shift
```

The horizontal component was chosen to match the suggestions contained in the IEEE standard. This height and width makes the objects easy to read in **Draft** at Zoom scale 1 and on plots or hardcopy that are done at 100 dots per inch.

Placing ACTIVE_LOWs

The IEEE standard suggests an ACTIVE_LOW be drawn so that it is 1.5 times as wide as it is high. Since OrCAD's IEEE objects are 8 units high by default, this means the ideal width of and ACTIVE_LOW is 12 units. In a TEXT command, "1.2" represents 12 units. ACTIVE_LOW inputs are usually placed with –1.2 X offset in order to get the point of the triangle on the outline.

The IEEE standard implies that ACTIVE_LOWs should only be used external to a part body. Signals internal to the body are intended to be logical, not physical. Hence, Negation or CIRCLEs should be used where a logical inversion operation is needed inside a part.

△ *NOTE: When a part is IEEE type, you have considerable leeway in deciding where the vector objects should be placed. As a result of various aspects of the IEEE standard, they can be either inside or outside the part's body outline. Any time you draw a library part with objects outside the part's body, however, you increase the chances of creating a messy and possibly confusing schematic. For this reason, add Vector objects outside the part's body only if there's no other way to define the part properly.*



Symbol Description Language

This chapter describes how to define a part in a custom library. It presents OrCAD's Symbol Description Language (SDL) in the form of syntax diagrams. A syntax diagram consists of identifiers (enclosed in ovals) and tokens (enclosed in rectangles).

Syntax diagram

Figure 23-1 is an example of a syntax diagram. It represents the complete syntax for a library source file.

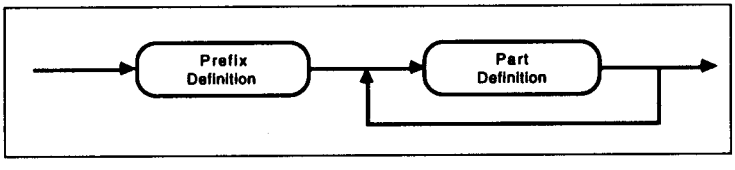


Figure 23-1. Syntax diagram for a library source file.

To read a syntax diagram, follow these rules:

1. Read the diagram from left to right.
2. The direction of the arrows on a path represent correct syntax forms.
3. Junctions represent a connection point where you have the choice of selecting another path. For example, the syntax diagram for the library source file shown in figure 23-1 has a junction after a part definition. You can choose to complete the diagram or to make another Part Definition.

4. You cannot travel a path going against the arrows. For example, in figure 23-1, you cannot return to the prefix definition after you make a part definition.
5. Text enclosed in ovals represents an identifier. Text enclosed in squares represents a token. Identifiers and tokens are described below.

Identifiers

Identifiers serve as placeholders for a more detailed level of syntax structure. They do not represent command syntax or tokens. Rather, they provide the ability to give an overview of the syntax. When you create the part, you must work down through all the nested identifiers. For example the syntax diagram for a library source file shown in figure 23-1 has two identifiers (Prefix Definition and Part Definition) and no tokens.

Tokens

Tokens are the building blocks of a library source file. Just as a sentence is made up of words, a library source file is made up of tokens. A token belongs to one of the following categories:

- **Numeric constants.** A numeric constant consists of one or more whole-number digits.

  Examples:

  ```
  15     127
   2      98
  ```

- **Character strings.** A character string consists of one or more alphanumeric characters.

  Examples:

  ```
  74ALS04          L5
  CLOCK            ZENER
  ```

- **Keywords.** A keyword is one of the following:

CLK	Represents the clock symbol in a pin definition.

CONVERT	Introduces a converted graphic symbol definition. With an argument, it refers to the converted vector definition of a graphic symbol.
DOT	Represents the inversion bubble in a pin definition.
END	Specifies the close of a prefix or vector definition.
GRIDARRAY	Specifies the device is a pin-grid array. Used in place of the number of parts per package.
HIZ	Identifies the pin as a high impedance (state) output.
IEEE	Specifies that the object is an IEEE object, and will conform to IEEE drawing standards.
IN	Identifies the pin as an input.
I/O	Identifies the pin as input/output.
OC	Identifies the pin as open collector.
OE	Identifies the pin as open emitter.
OUT	Identifies the pin as an output.
PAS	Identifies the pin as passive.
PREFIX	Specifies the beginning of a prefix definition.
PWR	Identifies the pin as a power pin. The PWR keyword prevents a pin from displaying.
REFERENCE	Takes an argument (a text string representing a reference value). Overrides the default reference value.
SHORT	Specifies the pin lead lengths be 1 grid unit instead of the standard 3 grid units.
VECTOR	Specifies the beginning of a part's vector definition.

How syntax is described in this chapter

Syntax described in this chapter is shown following these conventions:

UC	Text in uppercase represents a keyword.
italics	Text in italics represents either a character string or a numeric constant.
[]	Text enclosed in brackets is optional. You choose whether to type it in or not. Don't type the brackets.
{ }	Text enclosed in braces is ignored. You use braces to delimit comments.
,	If items within square brackets or braces are separated by commas, you choose one of them only. Don't type the comma.
. . .	Ellipses mean you can repeat the last item. How many times you can repeat the item depends on the context. Don't type the periods.

Vertical lists of options mean that you can choose any one of the options in the list.

Explanations follow each syntax definition. Within these definitions, keywords, character strings, and numeric constants are shown in the left margin with an arrow:

➨ **KEYWORD** This is an example of a keyword definition.

➨ *Character string* This is an example of a character string definition.

➨ *Numeric constant* This is an example of a numeric constant definition.

Example Here is an example of syntax represented in text:

[*pos pin#...,grid...*]

The entire line is enclosed in brackets, which means it is optional. The *pos*, *pin#*, and *grid* parameters are in italics, so each must be replaced with a valid character string or numeric constant. The comma between *pin#* and *grid* specifies either *pin#* or *grid*, but not both, may be used on the same line. The ellipses following *pin#* and *grid* specify that more than one occurrence of *pin#* or *grid* may appear on the same line.

Prefix definition

PREFIX

['*prefix string*' [= '*shorthand string*']]

.
.
.

END

➨ *prefix string* A string of up to seven text characters. You can have a maximum of sixteen prefix strings.

➨ *shorthand string* A string of up to seven text characters.

△ *NOTE: The equal sign may be separated by one or more space or tab characters.*

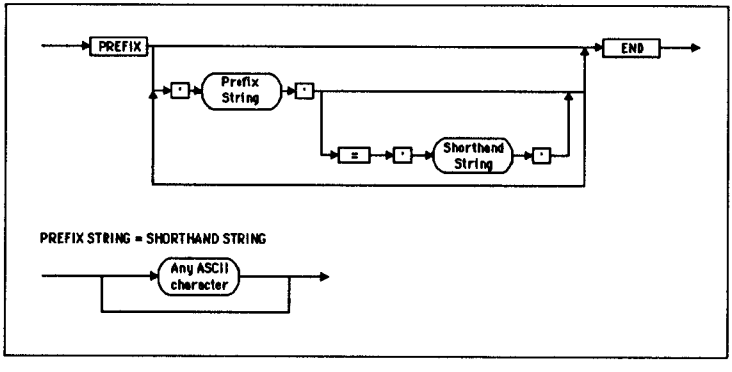


Figure 23-2. Syntax diagram for a prefix definition.

Example 1

```
PREFIX
'74LS'     =    'LS'
'74S'      =    'S'
'74ALS'    =    'ALS'
'74AS'     =    'AS'
'74HCT'    =    'HCT'
'74HC'     =    'HC'
'74ACT'    =    'ACT'
'74AC'     =    'AC'
'74F'      =    'F'
'74'
END
```

Example 2

```
PREFIX
END
```

Part definition

'part name string'
[REFERENCE *'ref string'*]
[SHEETPATH *'path and filename'*]
{*X-size Y-size parts-per-package*}
[*pin definition*]
.
.
.
[*vector definition*]
[*converted form vector definition*]

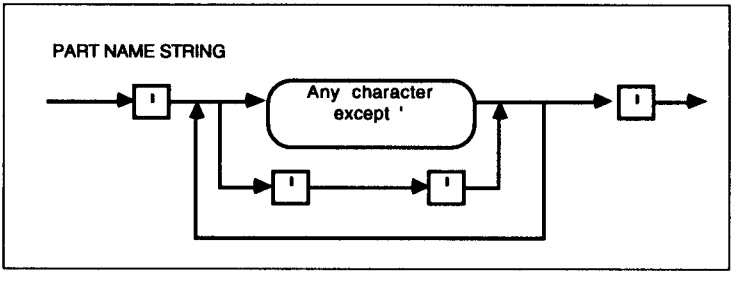


Figure 23-3. Syntax diagram of a part name string.

△ ***NOTES:*** *To type an apostrophe, use two single quotes. For example, to type:*

```
'John's'
```

type:

```
'John''s'
```

To improve readability, you can include comments within a part definition. Comment text is enclosed within braces. For example:

```
{This is a comment}
```

You can also place blank lines within a source file. Typically blank lines are placed between different part definitions.

➨ *part name string* A character string of up to 127 text characters identifying the part. This is the string used as an argument for the **GET** command in **Draft**.

It is a good idea to stay well below the maximum number of characters. Two reasons for this recommendation are:

- On a 640 x 480 screen, a name longer than 78 characters will have some characters clipped at zoom scale 1.
- While OrCAD's connectivity database is extremely flexible, some destination tools cannot accept long names. If you have a particular destination in mind, refer to the netlist format file for information about the limits of the destination tool. This information is available using **View Reference**.

➨ *ref string* A string of text characters. If present, the reference designator replaces the default reference designator.

➨ *path and filename* The name of a schematic referenced by a sheetpath part.

➨ *X size* A numeric constant in the range 1 to 255. The horizontal size of the part as it appears on a printed worksheet. Each entry corresponds to one grid unit.

➨ *Y size* A numeric constant in the range 1 to 200. The vertical size of the part as it appears on a printed worksheet. Each entry corresponds to one grid unit.

➨ *parts-per-package* A numeric constant in the range 0 to 16. If you specify a 0, the pins are not numbered on the symbol. If you specify GRIDARRAY, an alphanumeric string of up to 15 characters enclosed in single quotes is allowed as pin numbers. If you specify IEEE, you automatically get one part per package.

➨ *pin definition* See the pin definition description later in this section.

➨ *vector definition*

This definition describes the part body in vector form. See the vector definition description later in this section.

➨ *converted form vector definition*

This definition describes the part body of the part's convert (for example, its DeMorgan equivalent) in vector form. See the converted form vector definition description later in this section.

These examples shows two part name strings. These may be on the same or separate lines.

Example 1

```
'2114'   '2148'
6       14   1

pin definition
.
.
.
```

Example 2

```
'7474'
'74ALS74'
'74LS74'
'74S74'
'74HC74'
'74AC74'
6   6       2
pin definition
    .
    .
    .
```

Pin definition

[*pos* [*pin#,'grid'*] [DOT] [CLK] [SHORT] [[IN] [OUT] [I/O] [OC] [OE] [PWR] [PAS] [HIZ]] *pin name*]

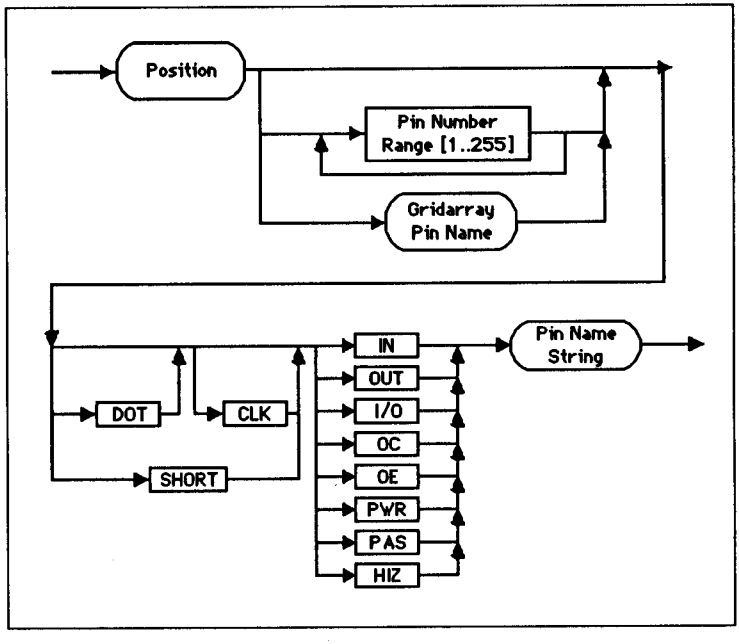


Figure 23-4. Syntax diagram for a pin definition.

➨ *pos* Defines pin position. A letter followed by a number. The letter is one of the following:

T indicates the top of the symbol.

L indicates the left side of the symbol.

R indicates the right side of the symbol.

B indicates the bottom of the symbol.

The number represents the distance along the indicated side. The distance is measured in grid unit lengths. For example, if the block symbol were 6X by 10Y the grid used for placing pins is as follows. Figure 23-5 shows the location of L3.

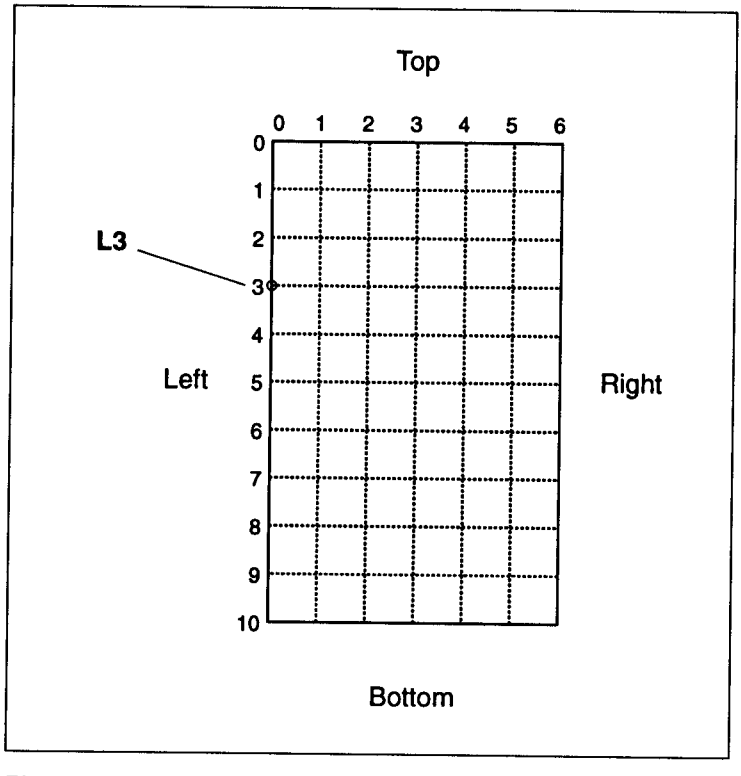


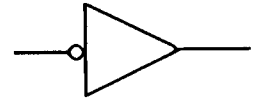
Figure 23-5. The grid of a 6X by 10Y block symbol.

➨ *pin #* If present, this is a numeric constant representing the pin number. This is the pin number appearing in the symbol. The range for the pin numbers is 0 to 255.

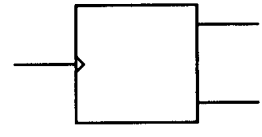
➨ *grid* A letter followed by a number. *grid* represents the pin-grid array pin number. You can only choose a pin-grid array pin number if you chose the keyword GRIDARRAY, instead of parts-per-package. A pin-grid array pin number is an alphanumeric string of up to 15 characters enclosed in single quotation marks.

➨ SHORT A keyword that places a short lead at the specified pin. A normal lead is 3 grid units long; a SHORT one is 1 grid unit long. The SHORT keyword cannot describe a pin also having either the CLK or DOT keywords.

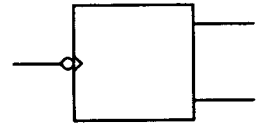
➨ DOT A keyword for placing the inversion symbol (the bubble, see right) at the specified pin location. The DOT keyword cannot describe a pin also having the SHORT keyword. The primary use of the bubble is to identify pins with logic negation, either at an input or at an output.



➨ CLK A keyword for placing the clock symbol (see right) at the specified pin location. The CLK keyword cannot describe a pin also having the SHORT keyword.



Use the CLK keyword together with the DOT keyword to produce a DOT CLK symbol (see right).



➨ IN A keyword identifying the pin as an input.

➨ OUT A keyword identifying the pin as a standard totem-pole output.

- **I/O** A keyword identifying the pin as a dual function input/output pin.

- **OC** A keyword identifying the pin as an open collector or open drain.

- **OE** A keyword identifying the pin as an open emitter.

- **PWR** A keyword identifying a power pin, such as VCC, GND, VSS, VDD, and others. Power pins are not displayed on library parts when they appear on the screen or printed worksheet. However, the connectivity database connects all power supply pins defined in library files.

- **PAS** A keyword identifying a pin as passive. Passive pins are typically pins on passive devices such as resistors, capacitors, inductors, and others.

- **HIZ** A keyword identifying a pin as a high-impedance (state) output.

➨ *pin name* A character string representing a name for the specified pin. For block symbols, this name appears on the screen or the printed worksheet. Pin names do not appear on the screen or the printed worksheet when they are part of pin definitions for a graphic symbol. However, you may still choose to use pin names in graphic symbols. The **Create Netlist** tool requires them, and you may find them useful to identify specific pins.

You can enter pin names either in upper- or in lowercase, but they always appear in uppercase.

The backslash (\) and single quote (') are special characters. A backslash (\) after a character indicates the character has a bar over it. If you want to bar multiple characters, you must place a backslash after each character. The single quote delimits the part name string. If you want a single quote as part of the pin string, you must delimit it with another single quote. Figure 23-6 shows the syntax for the pin name string.

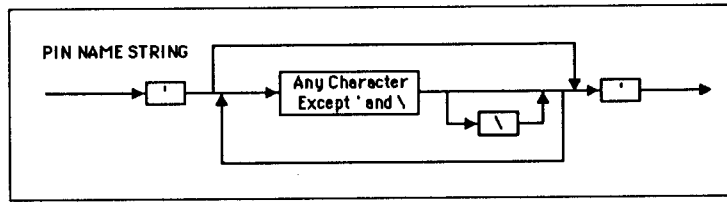


Figure 23-6. Pin name string syntax diagram.

Example 1

```
'2114'  '2148'
6       14      1
L1      5       IN      'A0'
L2      6       IN      'A1'
L3      7       IN      'A2'
L4      4       IN      'A3'
L5      3       IN      'A4'
L6      2       IN      'A5'
L7      1       IN      'A6'
L8      17      IN      'A7'
L9      16      IN      'A8'
L10     15      IN      'A9'
L12     8       IN      'C\S\'
L13     10      IN      'W\E\'
R1      14      HIZ     'D0'
R2      13      HIZ     'D1'
R3      12      HIZ     'D2'
R4      11      HIZ     'D3'
T0      18      PWR     'VCC'
B0      9       PWR     'GND'
```

Example 2

```
'68020'
15       66     GRIDARRAY
L1       'C2'   CLK IN 'CLK'
```

and so on

Example 3

```
'7474' '74ALS74' '74AS74' '74LS74'
'74S74' '74HC74' '74AC74'
6       6       2
L2      2       12      IN      'D'
L4      3       11      CLK IN 'CK'
B3      1       13      DOT IN 'CL'
```

and so on

Vector definition

VECTOR
ARC *Xcenter Ycenter Xedge1 Yedge1 Xedge2 Yedge2 radius*

CIRCLE *Xcenter Ycenter radius*

FILL *Xstart Ystart*

LINE *Xstart Ystart Xend Yend*

TEXT *Xstart Ystart height {'text'/keyword}*

END

➨ ARC — An ARC line defines a section of a circle. The first pair of X,Y coordinates specifies the center of the circle from which the arc is taken. The second pair of X,Y coordinates (edge1) defines one ending point on the arc. The third pair of X,Y coordinates (edge2) defines the other ending point on the arc. The edge1 and edge2 coordinates are relative to the center of the circle; they are not absolute coordinates. *radius* specifies the distance from the center to the outside of the arc.

➨ CIRCLE — The X,Y coordinates specify the center of the circle. *radius* specifies the distance from the center to the outside of the circle.

➨ FILL — FILL shades the enclosed area around the X,Y coordinates. One use of FILL is to darken the triangular shape in the symbol of a diode.

➨ LINE — The first pair of X,Y coordinates specifies the beginning of the line. The second pair of X,Y coordinates specifies the end of the line.

➨ TEXT The X,Y coordinates specify the point at which the lower left corner of the first character of the text starts. *height* defines the multiplier of the text size defined in the template table on the **Configure Schematic Design Tools** screen.

For example, if pin text is set to 0.06 inches in the template table, and the height is 2, then the actual plotted height is 0.12 inch. The remainder of the TEXT line can be either text enclosed in single quotes or a keyword.

keyword Keywords are IEEE/ANSI special symbols. If you use one of these keywords, the symbol corresponding to the keyword appears instead of text. Possible keywords are:

ACTIVE_LOW_L	HYSTERESIS
ACTIVE_LOW_R	NON_LOGIC
AMP_L	OPEN_O
AMP_R	OPEN_H
ANALOG	OPEN_L
ARROW_L	POSTPONED
ARROW_R	PULL_UP
BIDIRECTIONAL	PULL_DOWN
DYNAMIC_L	SHIFT_L
DYNAMIC_R	SHIFT_R
GENERATOR	THREE_STATE

For more information about these symbols, see ANSI/IEEE Std 91-1984.

Converted form definition

CONVERT
[*pin definitions*]
.
.
.
[[CONVERT '*part name*'],
[*vector definition*]]

➨ *pin definitions* — See the pin definition description earlier in this chapter.

➨ *part name* — The part name of a previously-defined part. If you specify part name with the CONVERT keyword, the definitions of the converted form of the previously defined part are used.

➨ *vector definition* — This specifies a new vector for the converted form of the part. See the vector definition description earlier in this section.

△ *NOTE: If you use the CONVERT* 'part name' *option, be sure the part name you refer to is previously defined.*



Compile Library

Compile Library takes a library source file and produces a compiled library file used by the schematic editor, processors, reporters, and transfers.

Creating a custom library with Compile Library

A custom library is a library that you modify or create. To create a custom library with **Compile Library**, follow these two steps:

1. Create a library source file. The source file is a text file containing instructions in OrCAD's Symbol Description Language (described in chapter 23).

   You can use any text editor to create the library source file. The only requirement is that it produce a text file without hidden formatting characters.

   You can also use the **Archive Parts in Schematic** tool to create a library source file. Another way to get a library source file is to use **Decompile Library** on an existing library.

2. Compile the source file using the **Compile Library** tool. It produces a compiled library file.

△ *NOTE: The maximum length for a part name is 127 characters. Use much smaller part names if you possibly can, however. On a 640 x 480 screen at ZOOM level 1, a name longer than 78 characters will be clipped. Also, many netlist formats place severe restraints on the length of names. Check the name length requirements for the netlist format you are using. This information is available using* ***View Reference.***

Execution

With the **Schematic Design Tools** screen displayed, select **Compile Library**. Select **Execute** from the menu that displays.

Compile Library creates the new library. When **Compile Library** is complete, the **Schematic Design Tools** screen appears.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Compile Library**. Select **Local Configuration** from the menu that displays.

Select **Configure COMPOSER**. A configuration screen appears (figure 24-1).

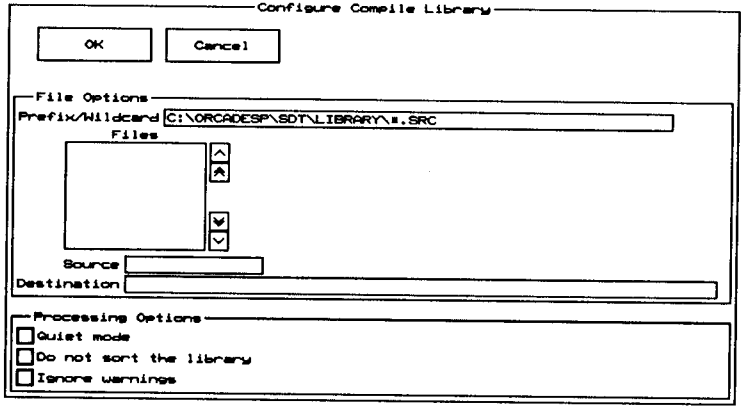


Figure 24-1. Compile Library's local configuration screen.

File Options

File Options defines the library source file and the resulting library file.

Prefix/Wildcard Enter a pathname and wildcard to indicate which files to display in the list box with scroll buttons. The asterisk character (*) is used as a wildcard. The default is:

```
\ORCADESP\SDT\LIBRARY\*.SRC
```

If you do not specify a prefix, **Compile Library** looks for files in the current design directory.

If you erase the entire field, the entry will be restored to whatever prefix is specified in **Configure Schematic Design Tools.**

Files The files that match the search filter entered in the **Prefix/Wildcard** entry box and those that match the filter in the current design directory are listed in this box. Files in the current directory are shown with .\ before their names. Use the scroll buttons to scroll the list of files up and down.

When you see the file you would like to compile, select it. Its filename displays in the **Source** entry box.

Source The **Source** is the text file describing your custom parts using OrCAD's Symbol Description Language.

Specify the source file by selecting it from the **Files** list box described above, or enter its name by simply typing it in this entry box and pressing <Enter>.

Destination The **Destination** is the library file created by **Compile Library**. If you give the name of an existing file, **Compile Library** asks if you want to overwrite the existing file. You cannot append to an existing file.

The **Destination** may be a complete pathname.

Processing Options You may select either or both of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Do not sort library`

  Tells **Compile Library** to save parts in the source file in the order they were entered (rather than sorting them alphabetically, which is the default).

- ❑ `Ignore warnings`

  Prevents **Compile Library** from displaying a message when it encounters warnings. **Compile Library** only issues a warning on one condition: it truncates pin names that exceed twenty-nine characters.

PART V: REPORTERS

Reporters are tools that produce human-readable reports, but do not modify design data in any way.

Part V describes the Reporter tools that change the design to a format you can read.

Chapter 25: *Check Electrical Rules* describes how **Check Electrical Rules** checks a design for conformity to basic electrical rules.

Chapter 26: *Cross Reference Parts* describes how **Cross Reference Parts** scans a design, gathers information for all parts used in the design, and creates a cross-reference listing telling where each part is located.

Chapter 27: *Convert Plot to IGES* describes how **Convert Plot to IGES** translates a plot of a single worksheet or a complete design into IGES format.

Chapter 28: *Plot Schematic* describes how **Plot Schematic** plots designs.

Chapter 29: *Print Schematic* describes how **Print Schematic** prints designs.

Chapter 30: *Create Bill of Materials* describes how **Create Bill of Materials** creates a summary list of all parts used in a design.

Chapter 31: *Show Design Structure* describes how **Show Design Structure** scans the worksheets in a design and displays the sheet names and their associated worksheet filenames.



Check Electrical Rules

The **Check Electrical Rules** reporter checks a design for conformity to basic electrical rules. It does not check for other types of errors in your design.

Checking for electrical errors

Check Electrical Rules scans a design and checks the sheets for unused inputs on parts and invalid connections, such as two part pins defined as outputs wired together. An example of the electrical rules checked by this reporter is included in this chapter. **Check Electrical Rules** reports two categories of electrical rules violations:

- Errors that *must* be fixed
- Warnings of situations that may or may not be all right in your design

Always carefully examine any problems reported by **Check Electrical Rules.**

You specify the conditions to be checked using the **Check Electrical Rules matrix** on the **Configure Schematic Design Tools** screen. See *How to specify conditions to check* in this chapter for more details.

Unless you tell it otherwise on the configuration screen, **Check Electrical Rules** is incremental. This means that if you run **Check Electrical Rules** twice on the same design without fixing anything, no errors are reported the second time it runs since there is nothing new to report.

If **Check Electrical Rules** finds warnings or errors, it marks them in the schematic file. If **Check Electrical Rules** reports "Program did not complete successfully" after running, go to **Draft** to fix the errors. Every warning or error is marked with a circle as shown in figure 25-1. The circle is placed on the net with the problem, but not necessarily at the exact location of the problem.

Note that the wire coming from pin 5 on U1B and the wire coming from pin 3 of U1A have circles on them.

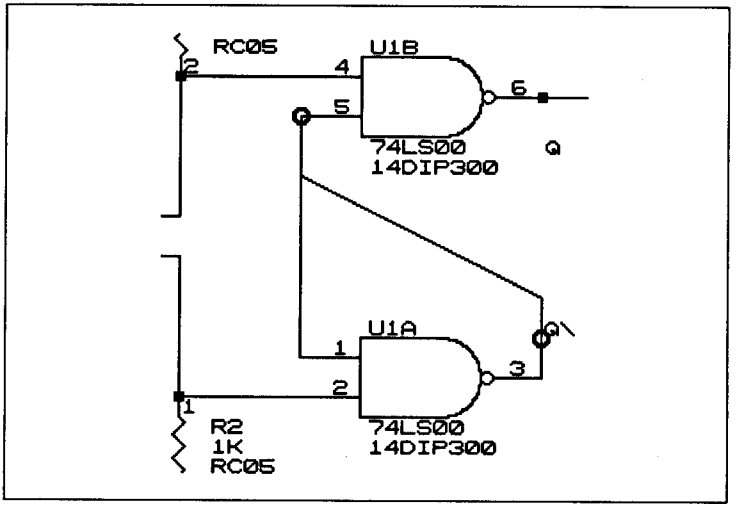


Figure 25-1. Schematic file with two warnings marked with circles.

Find and repair errors

Use the **INQUIRE** command to see the error message or warning for any trouble spots. Fix the problems, update the schematic file, and run **Check Electrical Rules** again.

Discard error markers

To discard error markers without making any changes to the schematic, select **QUIT Update File.**

Execution

Select **Check Electrical Rules** from the **Schematic Design Tools** screen. Select **Execute** from the menu that displays.

While **Check Electrical Rules** runs, messages display in the monitor box at the bottom of the screen. When it is done, the **Schematic Design Tools** screen appears.

Typical messages and resolutions

Table 25-1 lists the most common error messages produced by **Check Electrical Rules** and possible solutions to resolve the errors.

How to specify conditions to check

Figure 25-2 shows the decision matrix that tells **Check Electrical Rules** the conditions to check for when evaluating connections between pins, module ports, and sheet net names.

To configure the decision matrix to check for certain conditions, go to the **Check Electrical Rules Matrix** section of the **Configure Schematic Design Tools** screen. For more information on **Configure Schematic Design Tools**, see *Chapter 1: Configure Schematic Tools.*

The matrix shows the pins, module ports, and sheet net names in columns and rows. A connection is represented by the intersection of a row and column. The intersection of a row and column either is empty or contains a "W" or an "E." An empty intersection represents a valid connection, a **W** is a warning, and **E** represents an error.

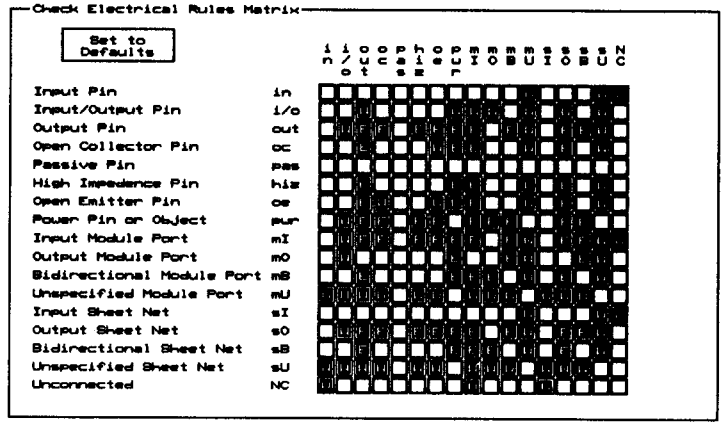


Figure 25-2. The decision matrix used by ***Check Electrical Rules*** *to check a connection.*

You can toggle between these three settings by pointing to an intersection and clicking the mouse until the desired setting appears. To return all intersections to their default settings, click the **Set to Default** button.

Suppose, for example, that you have an output pin connected to an input pin. To find the **Check Electrical Rules** value in the matrix, start in the OUT column (the third column) and go down to the IN row (the first row). The box is empty; this represents an acceptable connection. However, if you follow the OUT column down to the OUT row (the third row), you see an E, which indicates this condition will be detected as an error.

Connections prefixed with an "m" are module ports. You can have four types of module ports: input (mI), output (mO), bidirectional (mB), and unspecified (mU).

Connections prefixed with an "s" are sheet net names. As with module ports, you can have four types of sheet net names: input (sI), output (sO), bidirectional (sB), and unspecified (sU).

NC means Not Connected.

For definitions of the pin types, see the **PIN Type** command description in *Chapter 2: Draft*.

Message	*Check for*
WARNING: Unconnected MODULE PORT "....." at X=.... at Y=...	A bus on the schematic may be labeled improperly. A bus must be named in the form *busname[m..n]*. Any module port connected to a bus must also be named in the proper form *busname[m ..n]*. For further information, see *Chapter 10: Creating a netlist.*
WARNING: POWER Supplies are CONNECTED .... <->	If you have connected two power supplies, this warning appears. If this condition is acceptable in your design, you may ignore the warning. If you did not mean to connect two power supplies, this indicates a problem may exist.
WARNING: INPUT has NO Driving Source	If you did not connect wires to the input pins of a library part, this warning appears. Again, if this condition is acceptable in your design, you may ignore the warning. If wires are connected to an input pin and this message appears, you may have two wire ends overlapping or a wire overlapping a part pin. Wires must be connected end to end.
WARNING: Single Node Net	If you did not connect wires to a pin, this warning appears. If you intend the pin to be unconnected, you can place a **NoConnect** to suppress the warning or you can ignore the warning. If you passed a bus into a schematic and used some but not all members of the bus on that schematic, this warning appears. In this case you may ignore the warning.
ERROR: Module Port on a bus does not have a proper format....can not process....	When a module port is connected to a bus, it must be in the format *busname[m..n]*.
ERROR: Bus Label does not have a proper format....can not process...	When a label is placed on a bus, it must be in the format *busname[m ..n]*.
ERROR: Sheet Net on a bus does not have a proper format... can not process...	This error typically results from a bus connected to a hierarchical sheet net. The sheet net and module port must have the same name and number of members, but the sheet net and bus may differ at the root level. The form should be *busname[m ..n]*.

*Table 25-1. Error messages created by **Check Electrical Rules** and possible solutions.*

Example The following example shows the type of errors **Check Electrical Rules** looks for in schematic sheets. Figure 25-3 shows a schematic sheet containing a 74LS245 part from the TTL.LIB library. Notice pins 2, 3, 4, 7, and 8 are tied together and pins 5, 6, and 9 are tied together. These connections are electrically sound because pins 2 through 9 are bidirectional (I/O) type pins. Though this may not be a particularly useful circuit, it doesn't violate any electrical rules, so **Check Electrical Rules** doesn't report any errors.

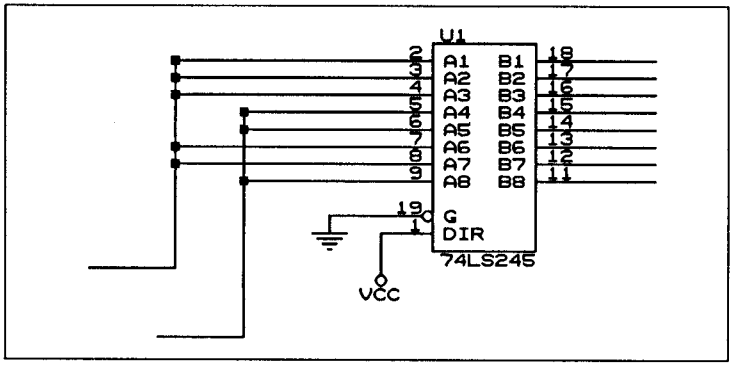


*Figure 25-3. Example circuit for **Check Electrical Rules** (without 74LS00).*

Suppose you add a 74LS00 part (also from TTL.LIB) to the sheet shown in figure 25-4. This introduces two potential electrical violations to the sheet because pin 6 on U2B and pin 3 on U2A are output type pins. Output pins usually are not connected to bidirectional pins. If you run **Check Electrical Rules** on the design with the decision matrix set to its defaults, it reports the following warnings:

```
Warning, Possible conflict I/O connected TO OUTPUT U2B,O
Warning, Possible conflict I/O connected TO OUTPUT U2A,O
WARNING - INPUT has NO Driving Source U2B,I0
WARNING - INPUT has NO Driving Source U2B,I1
WARNING - INPUT has NO Driving Source U2A,I0
WARNING - INPUT has NO Driving Source U2A,I1
```

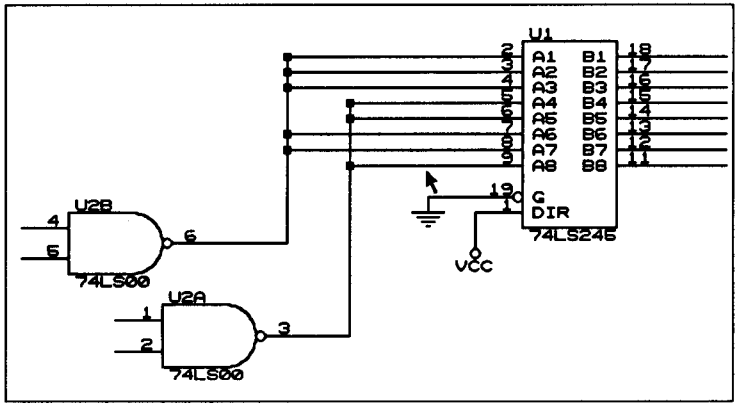


Figure 25-4. Example circuit for ***Check Electrical Rules*** *(with 74LS00).*

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Check Electrical Rules**. Select **Local Configuration** from the menu that displays.

Select **Configure ERC**. A configuration screen appears (figure 25-5).

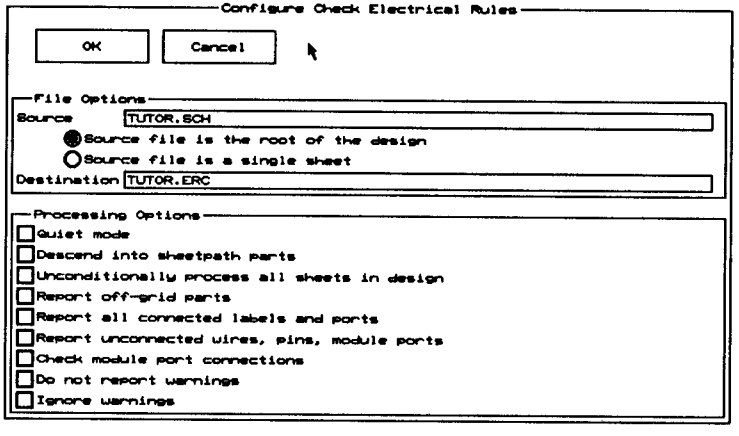


*Figure 25-5. Local configuration screen for **Check Electrical Rules**.*

File Options

File Options defines the source file, its type, and the destination file.

Source

The **Source** can be the root sheet name of the design, or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following options:

- O `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a I Link, it is a flat design.

- O `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Destination

The **Destination** can be any valid path name where the list of errors is to be placed. If a destination is not specified, the list of errors display in the monitor box at the bottom of the screen.

△ *NOTE: The errors and warnings are automatically saved to the schematic. The list of errors is useful for review, but is optional.*

Processing Options

You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Descend into sheetpath parts`

  Causes **Check Electrical Rules** to descend into any parts defined as sheetpath parts. If this option is not selected, **Check Electrical Rules** treats the sheetpath itself as a part to be cross-referenced and does not cross reference the parts within a sheetpath.

- `Unconditionally process all sheets in design`

  Forces **Check Electrical Rules** to check all sheets in the design regardless of whether or not the sheets changed since **Check Electrical Rules** ran last.

- `Report off-grid parts`

  Lists all off-grid parts in the report.

- `Report all connected labels and ports`

  Lists all connected labels and ports in the report.

- `Report unconnected wires, pins, module ports`

  Lists all unconnected wires, pins, and module ports in the report. Uses the maximum possible number of points on pins to determine whether the pins are electrically connected.

- `Check module port connections`

  Causes **Check Electrical Rules** to check all module ports and make sure they have a corresponding connection.

- `Do not report warnings`

  Causes **Check Electrical Rules** to run without testing some of the conditions for which it normally issues warnings. These conditions are always checked—unless you select this option—and cannot be changed with the **Check Electrical Rules Matrix**. These conditions are:

  - Two power objects connected
  - Single node nets
  - Input signals without a driving source

  Use this option with caution. You may end up with a netlist containing conditions that are not acceptable.

- `Ignore warnings`

  Prevents **Check Electrical Rules** from displaying a message when it encounters warnings.



Cross Reference Parts

The **Cross Reference Parts** reporter scans through the specified schematic files, gathers information for all parts used in the schematic files, and creates a cross reference listing telling you where each part is located. **Cross Reference Parts** scans a multiple-sheet file structure or a one sheet file structure.

By default, **Cross Reference Parts** produces two types of output listings. These listings differ in the order in which the parts are sorted. The first type of listing is sorted first by part values, then by reference designators. The second type of listing is sorted first by reference designators, then by part values.

Execution

Select **Cross Reference Parts** from the **Schematic Design Tools** screen. Select **Execute** from the menu that displays.

While **Cross Reference Parts** runs, messages display in the monitor box at the bottom of the screen. When the report is complete, the **Schematic Design Tools** screen appears.

Sample Output Figure 26-1 shows a sample report created by **Cross Reference Parts.**

```
Generic Netlist Example                          Revised:   October 11, 1990
OrCAD-01                                         Revision: A
Cross Reference                October   12, 1990     15:24:03   Page   1

 Item   Reference  Part      SheetName      Sheet  Filename
 ____________________________________________________________________
    1   R1         1K        <<<root>>>        1  EX1.SCH

    2   R2         1K        <<<root>>>        1  EX1.SCH

    3   S1         SW SPDT   <<<root>>>        1  EX1.SCH

    4   U1A        74LS00    <<<root>>>        1  EX1.SCH

    5   U1B        74LS00    <<<root>>>        1  EX1.SCH
```

*Figure 26-1. Report created by **Cross Reference Parts.***

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Cross Reference Parts.** Select **Local Configuration** from the menu that displays.

Select **Configure CROSSREF.** A configuration screen displays (figure 26-2).

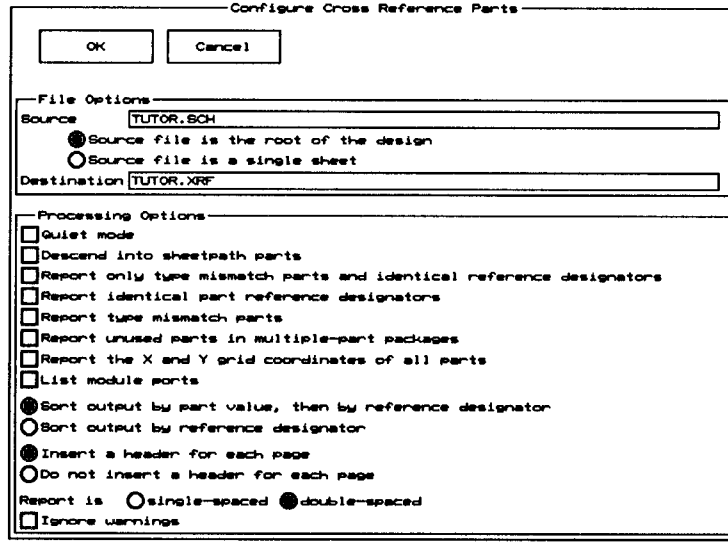


*Figure 26-2. Local configuration screen for **Cross Reference Parts.***

File Options **File Options** defines the source file, its type, and the destination file.

Source The **Source** can be the name of the root sheet of the design, or the filename of a single sheet. It may have any valid pathname.

After entering the **Source**, select one of the following:

- `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a I Link, it is a flat design.

- `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Destination The **Destination** can be any valid path name where the listings are to be placed. If a **Destination** is not specified, the listings are displayed in the monitor box at the bottom of the screen.

Processing Options You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Descend into sheetpath parts`

  Causes **Cross Reference Parts** to descend into any parts defined as sheetpath parts. If this option is not selected, **Cross Reference Parts** treats the sheetpath itself as a part to be cross referenced and does not cross reference the parts within a sheetpath.

- ❑ `Report only type mismatch parts and identical reference designators`

  Tells **Cross Reference Parts** to create a report of type mismatches and identical reference designators, but not to create the usual report.

- ❑ `Report identical part reference designators`

  Tells **Cross Reference Parts** to check for identical part references in addition to creating the usual report. You should not have identical part references in your design. Remove them by editing them with **Draft** or running **Annotate Schematic**.

- ❑ `Report type mismatch parts`

  Tells **Cross Reference Parts** to report type mismatches in addition to the usual report. Two kinds of mismatches are possible:

  - ❖ Parts with the same part name, but different numbers of parts per package.

    For example, a U1 and a U1A. This would mean that the U1 has one part per package, while the U1A has more than one part per package.

  - ❖ Parts with similar reference numbers, but different part names.

    For example, a 74LS00 with the reference number U1A and a 54LS00 with reference number U1B.

- ❑ `Report unused parts in multiple-part packages`

  Tells **Cross Reference Parts** to list all unused parts in multiple-part packages in addition to its usual report.

- ❑ `Report the X, Y coordinates of all parts`

  Tells **Cross Reference Parts** to list the X,Y coordinates for all parts.

- ❑ `List module ports`

  Tells **Cross Reference Parts** to include in its usual report a list of all module ports plus warnings about single occurrences of module port names.

Select either of the following options:

- ○ `Sort output by part value, then by reference designator`

  Tells **Cross Reference Parts** to list all parts sorted first by part values and then by reference designators.

- ○ `Sort output by reference designator`

  Tells **Cross Reference Parts** to list all parts sorted first by reference designators and then by part values.

Select either of the following options to specify whether a header is placed on each page of the report:

- ○ `Insert a header for each page`
- ○ `Do not insert a header for each page`

Select either of the following options to specify whether the report is single- or double-spaced:

`Report is`

- O `single-spaced`
- O `double-spaced`

If desired, select this option:

- ❑ `Ignore warnings`

  Prevents **Cross Reference Parts** from displaying a message when it encounters warnings.



Convert Plot to IGES

Convert Plot to IGES translates a plot of a single worksheet or a complete design into *IGES*, Initial Graphic Exchange Specification, text format. The IGES data format is application-independent and includes information about geometric and topological shapes, physical dimensions, tolerances, and other intrinsic properties.

To learn more about IGES, read the *Initial Graphics Exchange Specification* (IGES) Version 3.0, Number PB86–199759, published by the National Bureau of Standards, Washington D.C.

After you run **Convert Plot to IGES**, the file can be used with other applications that accept IGES input—such as VersaCad®—or stored on a mainframe computer.

Plot the file

Before you run **Convert Plot to IGES**, plot your worksheet with the GENERIC.DRV plotter driver configured. See *Chapter 1: Configure Schematic Tools* and *Chapter 28: Plot Schematic* for information on configuring plotter drivers and plotting.

Execution

With the **Schematic Design Tools** screen displayed, select **Convert Plot to IGES**. A menu displays. Select **Execute**.

While **Convert Plot to IGES** runs, messages display in the monitor box at the bottom of the screen. When the report is complete, the bottom of the **Schematic Design Tools** screen displays.

Sample output Figure 27-1 shows a sample report created by **Convert Plot to IGES.**

```
IGES file generated from OrCAD Schematic Design Tools                    S      1
,,21HOrCAD IGES Translator,,5HOrCAD,4HIGES,16,11,53,11,53,,1.0,3,4HINCH,G      1
30,0.46785,6H000001,,,8HIGESTRAN,5HOrCAD,6,0;                            G      2
     110       1       1       1       1                          00000000D      1
     110       1       1       1                                new       D      2
     110       2       1       1       1                          00000000D      3
     110       1       1       1                                new       D      4
     110       3       1       1       1                          00000000D      5
     110       1       1       1                                new       D      6
     110       4       1       1       1                          00000000D      7
     110       1       1       1                                new       D      8
110,0.77,0.583,.0,0.77,0.517,.0;                                        1P      1
110,0.93,0.583,.0,0.77,0.583,.0;                                        3P      2
110,0.93,0.517,.0,0.93,0.583,.0;                                        5P      3
110,0.77,0.517,.0,0.93,0.517,.0;                                        7P      4
S      1G      2D      8P      4                                        T      1
```

*Figure 27-1. Output created by **Convert Plot to IGES.***

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Convert Plot to IGES**. Select **Local Configuration** from the menu that displays.

Select **Configure PLT2IGES**. A configuration screen appears (figure 27-2).

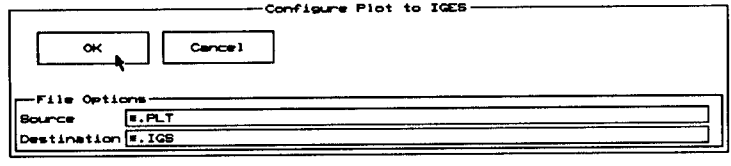


*Figure 27-2. **Convert Plot to IGES**'s local configuration screen.*

File Options

File Options defines the source file and the destination of the IGES plot.

Source

The **Source** is the name of the plot file or group of plot files to be translated into IGES format.

The source may be a single plot file or a wildcard specification. The suggested wildcard specification is `*.PLT`. When **Plot Schematic** runs, it should be configured with a destination of `?.PLT` so that all of the sheets will be ready to convert to IGES format.

Destination

The **Destination** is any valid path name where the output of **Convert Plot to IGES** is to be placed.

If the source is a wildcard specification, the destination should also be a wildcard specification. The suggested destination wildcard specification is `*.IGS`.



Plot Schematic

Plot Schematic plots schematic sheets.

There are two types of output devices that can be used with OrCAD's **Schematic Design Tools 386+**: plotters and printers. These devices are categorized by the type of input they require.

If a device accepts *vector* commands, it is considered to be a plotter. A vector is a series of points with a specific function defined. For example, a line has a beginning point and an ending point. A circle has a center and a radius. The device needs to know what the vector information is but does not need every point along the vector.

If a device accepts *raster* commands, it is a printer. A raster is an array of dots. When you draw a line to a raster device, you must specify each and every dot.

Plots are higher resolution than prints and usually take longer to produce because of the mathematical calculations needed. If **Plot Schematic** takes longer than you would like on your machine, you can use **Print Schematic** (described in chapter 29) to print working copies of your schematic during the design process. **Print Schematic** is faster, but produces output of lower quality. Use **Plot Schematic** to create final plots and scaled plots.

Plot Schematic can plot all sheets in a multiple-sheet file structure or just one sheet. It makes one copy of each sheet referenced more than once in a complex hierarchy.

Execution

Before you run **Plot Schematic**, be sure the appropriate plotter is configured on the **Configure Schematic Design Tools** screen. For more information about configuring your plotter, see *Chapter 1: Configure Schematic Tools.*

With the **Schematic Design Tools** screen displayed, select **Plot Schematic**. Select **Execute** from the menu that displays.

While **Plot Schematic** runs, messages display in the monitor box at the bottom of the screen. If **Plot Schematic** cannot find a part in the configured libraries, it lists the name of the part. When **Plot Schematic** completes the plot, the **Schematic Design Tools** screen appears.

Plots can include grid references in the output. **Plot Schematic** plots all Stimulus, Trace, Vector, Layout, and No-Connect objects.

△ *NOTE: **Plot Schematic** does not check to see if the plot will fit in the plot area of a plotter, nor does it split a design into several pages on a plotter if its image does not fit on a single page. If you choose the **Send output to printer** option, however, **Plot Schematic** prints a large image on multiple sheets.*

Suppressing the title block, border, and text

You can make **Draft** and **Configure Schematic Design Tools** cause the title block, title block text, and the design's border to not appear on the plot. These settings are summarized below.

- To cause the title block and its text to not appear on **Draft**'s screen, select **SET Title Block No** in **Draft.** Note that the design's border is not removed.
- To cause the title block or title text to not appear on a plot, display the **Color and Pen Plotter Table** of the **Configure Schematic Design Tools** screen and set the pen number to 99 for the title block or the title text. If you set the title block's pen number to 99, the design's border is also removed from the plot.

- To cause the title block or title text to not appear on **Draft's** screen, display the **Color and Pen Plotter Table** of the **Configure Schematic Design Tools** screen and set the color to black for the title block or the title text. With the title block's color set to black, the design's border does not show on **Draft's** screen.

Sample output Figure 28-1 shows a sample plot created by **Plot Schematic.**

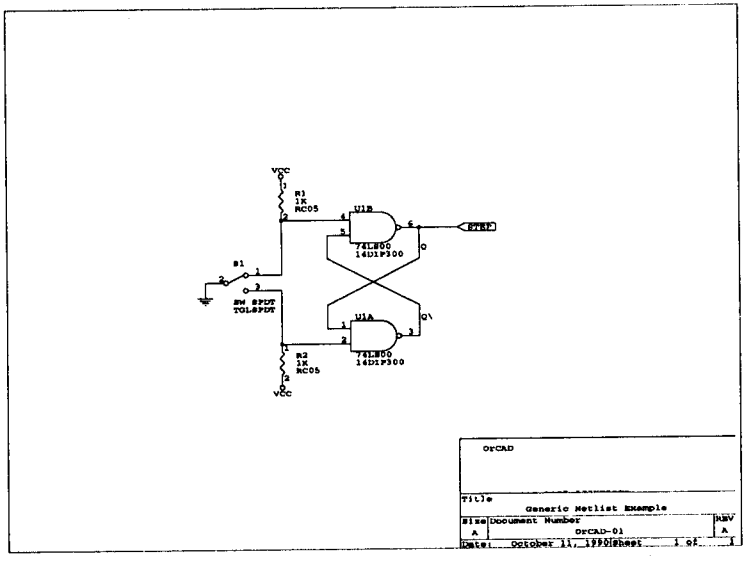


*Figure 28-1. Plot created by **Plot Schematic.***

Local Configuration

Select **Plot Schematic** from the **Schematic Design Tools** screen. A menu displays. Select **Local Configuration.**

Select **Configure PLOTALL.** A configuration screen appears (figure 28-2).

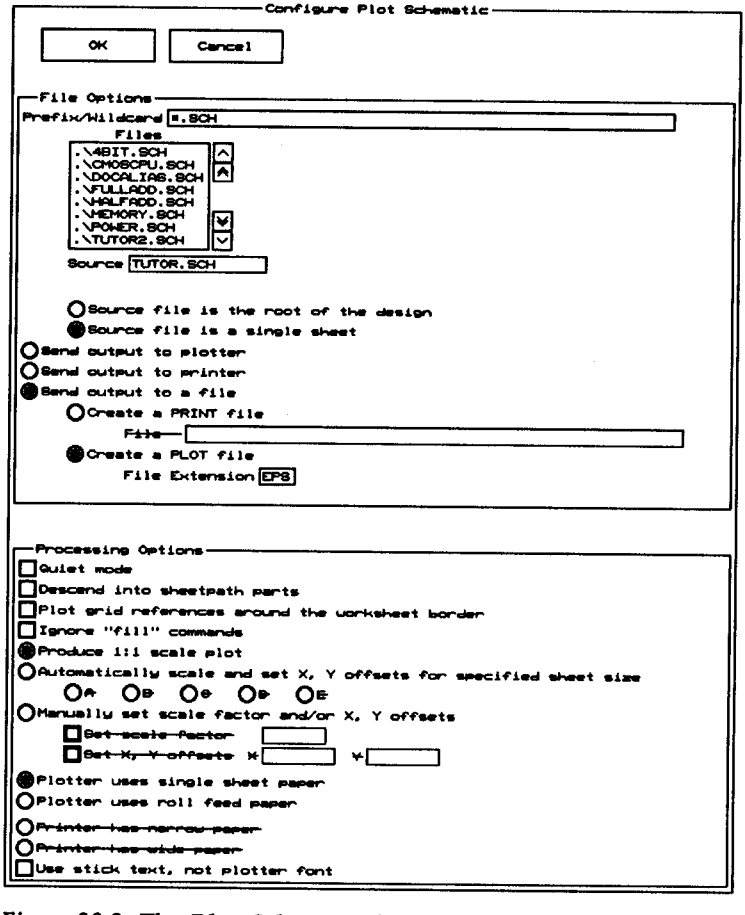


Figure 28-2. The ***Plot Schematic*** *local configuration screen.*

File Options **File Options** defines the source file and its type and the destination of the plot.

Prefix/Wildcard The **Prefix/Wildcard** defines the drive and directory containing the schematic (.SCH) files to use as source. It also serves as a "filter" of the files listed in the **Files** list box. It may have any valid pathname, including the standard DOS asterisk (*) and question mark (?) wildcards.

For example:

`Prefix/Wildcard` `*.SCH`

Lists all files in the current design directory with a .SCH extension. This is the default.

`Prefix/Wildcard` `..\MYDESN\*.SCH`

Lists all files in the MYDESN design directory with a .SCH extension.

`Prefix/Wildcard` `???*.*`

Lists all files with at least three characters in the filename, and with any extension.

Source The **Source** can be the root sheet name of the design, or the filename of a single sheet, with any valid pathname.

After entering the source filename, select one of the following options:

- ○ `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a I Link, it is a flat design.

- ○ `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Select one of the following three options:

- O `Send output to plotter`

  Use this option to create a high-quality plot of your schematic. Make sure you have configured the plotter driver on the **Configure Schematic Design Tools** screen to the plotter you want to use. See *Chapter 1: Configure Schematic Tools* for more information.

- O `Send output to printer`

  Use this option to create a scaled print of your schematic. Make sure you have configured the printer driver on the **Configure Schematic Design Tools** screen to the printer you want to use. See *Chapter 1: Configure Schematic Tools* for more information.

- O `Send output to a file`

  Sends the **Plot Schematic** output to a file, rather than to a printer or plotter.

  If the **Send output to a file** option is selected, you can select one of the following options:

  - O `Create a PRINT file`

    `File` [ ]

  - O `Create a PLOT file`

    `File Extension` [ ]

  If either of these options is selected, the output is stored in a file (or files) containing the formatting codes required by your printer or plotter. Be sure you have configured the printer or plotter driver on the **Configure Schematic Design Tools** screen for the printer or plotter you want to use. See *Chapter 1: Configure Schematic Tools* for more information.

Create a PRINT file creates one file per worksheet in the design, with the filename extension you specify in the **File Extension** entry box appended to the sheet name. You can then plot single sheets, or send the set of files to the plotter sequentially to plot the entire design.

Create a PLOT file creates one file per worksheet in the design, with the filename extension you specify in the **File Extension** entry box appended to the sheet name—*except* when the option **Plotter uses roll feed paper** is selected. (This exception is described in the next paragraph.) You can then plot single sheets, or send the set of files to the plotter sequentially to plot the entire design.

When the option **Plotter uses roll feed paper** is selected, **Create a PLOT file**, like **Create a PRINT file**, produces one large file containing plot information separated by page breaks (roll feed commands).

Processing Options

You may select any combination of the following options:

- `Quiet mode`

  Turns quiet mode on.

- `Descend into sheetpath parts`

  Causes **Plot Schematic** to descend into parts defined as sheetpath parts. Without this option selected, **Plot Schematic** treats the sheetpath itself as a part and does not plot the referenced sheets.

- `Plot grid references around the worksheet border`

  Tells **Plot Schematic** to include grid references in the plotted output.

- `Ignore "fill" commands`

  Specifies that parts with fills should not be filled in when plotted. Use this switch when you want to plot a draft version of your design quickly.

Select one of the following options:

- ○ Produce 1:1 scale plot
- ○ Automatically scale and set X, Y offsets for specified sheet size
  ○ A ○ B ○ C ○ D ○ E

  Tells **Plot Schematic** to use the X,Y offsets of the specified sheet size, as configured in the template table. Select one of the following: A, B, C, D, or E.
- ○ Manually set scale factor and/or X, Y offsets

  ❑ Set scale factor [ ]

  Tells **Plot Schematic** to scale the plot by the scale factor you enter in **Set scale factor** entry box. The scale factor is a decimal number in the form: #.### . The default scale factor is 1.000. The allowed range is 0.10 to 10.000.

  Table 28-1 shows the scale factors to use to plot worksheets on different sizes of plotter paper. The scale factors shrink or expand the image so the widest axis (horizontal or vertical) fits on the plotter paper. The narrow axis will have some blank area.

Worksheet Size	*Plotter Paper Size*				
	A	*B*	*C*	*D*	*E*
A	1.000	1.347	2.082	2.806	4.351
B	0.638	1.000	1.329	2.083	2.776
C	0.474	0.638	1.000	1.329	2.089
D	0.301	0.472	0.627	1.000	1.311
E	0.224	0.301	0.472	0.627	1.000

*Table 28-1. **Plot Schematic** scale factors.*

△ *NOTE: Table 28-1 assumes you are using the default* ***Template Table*** *values for Horizontal and Vertical paper dimensions. These values are set on the* ***Template Table*** *section of the* ***Configure Schematic Design Tools*** *screen. For more information, see* Chapter 1: Configure Schematic Tools.

To scale the size of **Plot Schematic**'s plot, find the scale factor in Table 28-1 corresponding to the worksheet size and plotter paper size you are using. Enter the number in the **Set scale factor.**

Scaling is also controlled by your plotter. Plotter scaling is typically controlled by the size of the paper used, rotation settings, and the other settings on the plotter.

If you change the size of the worksheets you are plotting (for example, plotting a C-size, then a B-size worksheet), *always RESET the plotter first* to return the plotter to a known state. If you have further scaling questions, see your plotter manual.

❑ Set X, Y offsets X [ ] Y [ ]

Specifies the plot's X and Y offset. Enter a decimal number, measured in inches from the center. The allowed range is –30.000 to 30.000.

Some plotters have their origin in the center rather than the lower left corner. If your schematic appears in the upper-right quadrant of the paper, set the X and Y offsets to negative values to move the origin to the lower left corner. Note X and Y are not affected by scaling.

Some plotter drivers (for example, HP.DRV) produce a divide error when you specify D size or E size and do not specify an X,Y offset.

The plotter driver converts inches into plotter units and for large size plots the resulting number exceeds the 16-bit integer limit. Hence, large paper plotters compensate by having the origin in the center of the paper. You must adjust the origin back to the lower left corner with the X and Y offsets. Typically, the offsets are –½ the **Horizontal** and **Vertical** dimensions shown on the template table in **Configure Schematic Design Tools**.

△ *NOTE: The X and Y custom offsets are measurements after scaling is applied. For example, if you have a C-size worksheet you want to plot on B-size paper, you must use the* ***Use scale factor*** *option and specify 0.638 as the* ***Scale factor****. However, the X and Y values are B-size measurements.*

See *Appendix E: Plotter information* for a discussion of HP, HI, and Calcomp plotters.

Select one of the following if **Send output to Plotter** or **Create a PLOT file** is selected:

O `Plotter uses single sheet paper`

O `Plotter uses roll feed paper`

Tells **Plot Schematic** that the plotter you are using has a roll feed. **Plot Schematic** inserts the commands necessary to roll the paper after each sheet.

△ ***NOTE:*** *Not all plotters have roll-feed capability.*

Select either of the following if **Send output to Printer** or **Create a PRINT file** is selected:

O `Printer has narrow paper`

O `Printer has wide paper`

Tells **Plot Schematic** whether wide (13-inch) paper or narrow (8-inch) paper is used in the printer.

Select this option if your schematic contains rotated text or special (extended ASCII) characters, and you want the text to have a uniform look:

❑ `Use stick text, not plotter font`

Tells **Plot Schematic** to draw all characters with line-drawing commands, as it does with rotated text and special characters. Without this option selected, **Plot Schematic** uses the plotter's fonts for horizontal and vertical text containing no extended ASCII characters.

This option creates larger files that take longer to plot, because each character is defined and plotted as a sequence of line-drawing commands.



Print Schematic

Print Schematic prints schematic sheets. Use **Print Schematic** to print interim (working) copies of your design during the design process. Use **Plot Schematic** to plot the schematic when design work is complete. You can also use **Plot Schematic** with **Send output to printer** selected to print a scaled schematic.

Print Schematic prints one copy of each sheet that is referenced in a complex hierarchy, even if that sheet is referenced more than once.

There are two basic types of output devices that can be used with **Schematic Design Tools 386+**: plotters and printers. These devices are categorized by the type of input they require.

If a device accepts *vector* commands, it is considered to be a plotter. A vector is a series of points with a specific function defined. For example, a line has a beginning point and an ending point. A circle has a center and a radius. The device needs to know what the vector information is but does not need every point along the vector.

If a device accepts *raster* commands, it is a printer. A raster is an array of dots. When you draw a line to a raster device, you must specify each and every dot. **Print Schematic** only addresses raster devices.

Execution

Select **Print Schematic** from the **Schematic Design Tools** screen. Select **Execute** from the menu that displays.

While **Print Schematic** runs, messages display in the monitor box at the bottom of the screen. When the report is complete, the **Schematic Design Tools** screen appears.

Sample output

Figure 29-1 shows a sample printout created by **Print Schematic.**

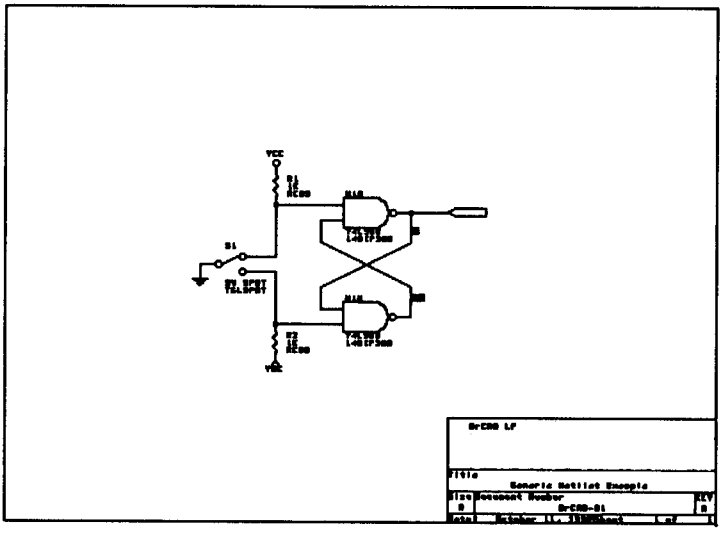


Figure 29-1. Output created by ***Print Schematic.***

Output size

The size of the printed image depends on the printer driver **Draft** uses. **Print Schematic** always produces an image at a resolution of 100 DPI (dots per inch). If the printer driver prints at some other resolution, the printed image is changed by a fixed scale factor equal to 100 divided by the printer driver resolution.

- If the driver resolution is greater than 100 DPI, the printed image is smaller.
- If the driver resolution is less than 100 DPI, the printed image is larger.

For example, if the printer driver you use prints at a resolution of 300 DPI, the image is printed at one-third (100 + 300) its original size. If the driver prints at 75 DPI, the image is printed at 1⅓ (100 + 75) its original size.

See the *Driver Options* section in *Chapter 1: Configure Schematic Tools* for information on selecting a printer driver.

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Print Schematic**. Select **Local Configuration** from the menu that displays.

Select **Configure PRINTALL**. A configuration screen appears (figure 29-2).

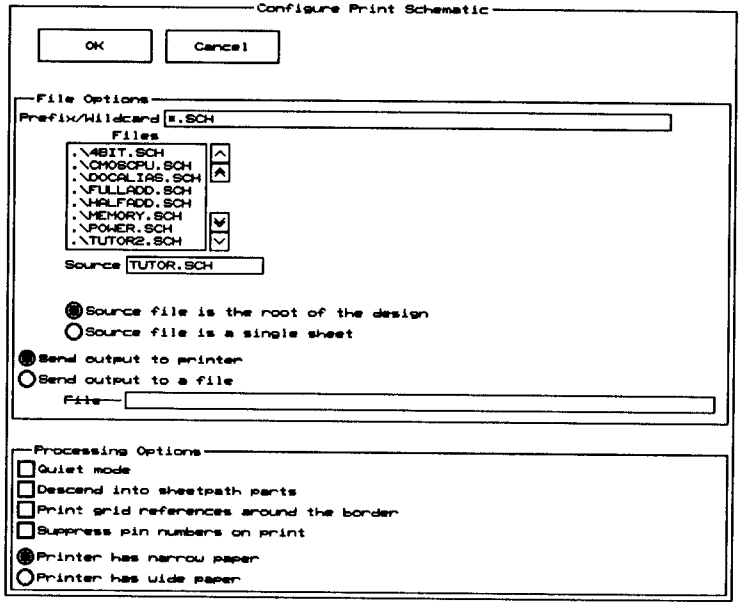


*Figure 29-2. **Print Schematic's** local configuration screen.*

File Options

File Options defines the source file and type and the destination of the print.

Source

The **Source** can be the root sheet name of the design, or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following options:

- ○ `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a |Link, it is a flat design.

- ○ `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Select one of the following options:

- ○ `Send output to printer`
- ○ `Send output to a file`

  If you select **Send output to a file**, enter the path and filename of the output file in the **File** entry box.

Processing Options

You may select any combination of the following options:

- ❑ `Quiet mode`

  Turns quiet mode on.

- ❑ `Descend into sheetpath parts`

  Tells **Print Schematic** to descend into parts defined as sheetpath parts. Without this option selected, **Print Schematic** treats the sheetpath itself as a part and does not print the referenced sheets.

- ❑ `Print grid references around the border`

  Tells **Print Schematic** to include grid references in the plotted output.

- ❑ `Suppress pin numbers on print`

  Tells **Print Schematic** to print the schematic without pin numbers.

Select either of the following options:

- O `Printer has narrow paper`
- O `Printer has wide paper`

Specifies whether wide (13-inch) paper or narrow (8-inch) paper is used to plot to a printer.



Create Bill of Materials

The **Create Bill of Materials** reporter creates a summary list of all parts used in a design. **Create Bill of Materials** scans a design or only a single sheet.

Optionally, special information specific to your application may be added in a text file called an *include file*. If this special information is included, **Create Bill of Materials** lists the parts found in the order in which they appear in the include file. Any parts not in the include file are placed at the end of the report.

Execution

With the **Schematic Design Tools** screen displayed, select **Create Bill of Materials**. Select **Execute** from the menu that displays.

While **Create Bill of Materials** runs, messages display in the monitor box at the bottom of the screen. When the report is complete, the **Schematic Design Tools** screen appears.

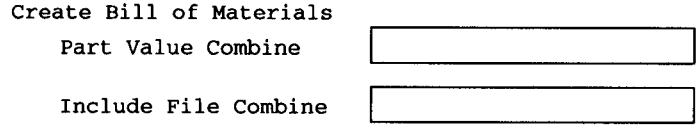
Sample output Figure 30-1 shows a sample parts list created by **Create Bill of Materials.**

```
Generic Netlist Example                          Revised:   October 11, 1990
OrCAD-01                                         Revision: A
Bill Of Materials                October 15, 1990  8:38:46          Page      1

   Item     Quantity    Reference              Part
_____________________________________________________

      1         2        R2,R1                  1K

      2         1        S1                     SW SPDT

      3         1        U1                     74LS00
```

*Figure 30-1. Bill of Materials created by **Create Bill of Materials.***

Key fields

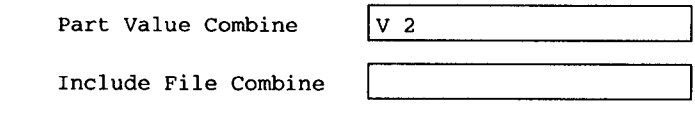
Create Bill of Materials has two key fields. They are shown in the **Key Fields** section of the **Configure Schematic Design Tools** screen as follows:



Create Bill of Materials uses the **Part Value Combine** key field to combine the part value and part fields together to become the value used in the summary list. For parts to have the same value in the **Create Bill of Materials** report, they must have the same values in the key field. If you do not specify a key field, the **Create Bill of Materials** value is the part's value.

The **Include File Combine** is used to build a lookup string to match in the Include File. As with the **Part Value Combine**, if this key field is not specified, the **Part Value** is assumed to be the match string.

For example, you configure this to include both the part value and the information in part field 2:



Create Bill of Materials is case-sensitive when checking include file match strings.

For more information about key fields, see *Chapter 1: Configure Schematic Tools.*

Local Configuration

With the **Schematic Design Tools** screen displayed, select **Create Bill of Materials**. Select **Local Configuration** from the menu that displays.

Select **Configure PARTLIST**. A configuration screen appears (figure 30-2).

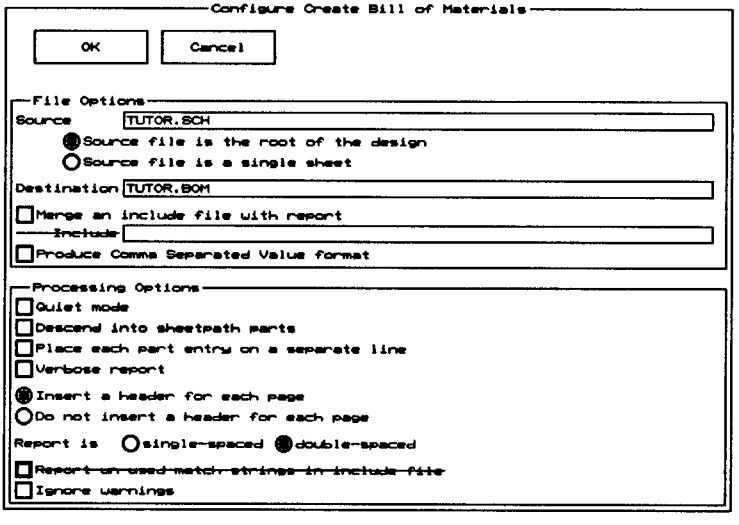


Figure 30-2. Local configuration screen for ***Create Bill of Materials.***

File Options

File Options defines the source file and its type. It also defines the destination file and whether to merge an include file with the partlist.

Source

The **Source** can be the root sheet name of the design, or the filename of a single sheet. It may have any valid pathname.

After entering the source filename, select one of the following:

○ `Source file is the root of the design`

Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a I Link, it is a flat design.

○ `Source file is a single sheet`

Specifies that the source file is a single worksheet and you want to process the single sheet only.

Destination

The **Destination** is the path and filename, and is where the output of the report is to be placed. If a destination is not specified, the output of **Create Bill of Materials** displays in the monitor box at the bottom of the screen.

Include

❑ `Merge an include file with report`

The **Include** is the path and filename of a text file containing information to be merged in the Bill of Materials. The format of this file is discussed below.

After you select this option, the **Include** entry box and the **Report un-used match strings in an include file** option change from dim to highlighted. Enter the name of the include file in the **Include** entry box.

Include file format

The include file is a text file in which you place additional part information. This part information is included in the **Create Bill of Materials** summary list. A sample include file is shown in figure 30-3.

The first line of the file is a header line. The line begins with a pair of single quotes with no characters or spaces between them. The rest of the line contains the header information you want to include.

The remainder of the file contains separate lines for each part needing additional information. Each line must begin with the part name as it appears in your worksheet. The name must be enclosed within single quotes (such as '74LS00'). Following the part name (on the same line), place the information that you want included for that part (for example, "TTL Quad Two Input NAND Gate 10004040000").

For both types of lines, header and part, the line will be aligned with the first non-space character of the information portion of the line. When **Create Bill of Materials** has finished scanning the sheets, it then scans the include file to include the rest of the line after any part name that matches.

```
''          DESCRIPTION                  Part Order Code
'1K'        Resistor 1/4 Watt 5%             10000111003
'4.7K'      Resistor 1/4 Watt 5%             10000114703
'22K'       Resistor 1/4 Watt 5%             10000112204
'1uf'       Capacitor Ceramic Disk           10000211006
'.1uf'      Capacitor Ceramic Disk           10000211007
'.01uf'     Capacitor Ceramic Disk           10000211008
'.001uf'    Capacitor Ceramic Disk           10000211009
'7400'      TTL Quad Two Input NAND Gate     10001040000
'74LS00'    TTL Quad Two Input NAND Gate     10002040000
'74S00'     TTL Quad Two Input NAND Gate     10003040000
'74ALS00'   TTL Quad Two Input NAND Gate     10004040000
'74AS00'    TTL Quad Two Input NAND Gate     10005040000
'7402'      TTL Quad Two Input NAND Gate     10001040002
'74LS02'    TTL Quad Two Input NAND Gate     10002040002
'74S02'     TTL Quad Two Input NAND Gate     10003040002
'74ALS02'   TTL Quad Two Input NAND Gate     10004040002
'74AS02'    TTL Quad Two Input NAND Gate     10005040002
```

Figure 30-3. Include file format.

File format

❑ `Produce Comma Separated Value format`

Produces the bill of materials report, which includes the reference, value, and 8 part fields for all modules, in a comma-delimited format.

When you select this option, the **Merge an include file with report** option and several of the processing options are unavailable and display as dimmed text.

Processing Options

You may select any combination of the following options:

❑ `Quiet mode`

Turns quiet mode on.

❑ `Descend into sheetpath parts`

Tells **Create Bill of Materials** to descend into any parts defined as sheetpath parts. Without this option selected, **Create Bill of Materials** treats the sheetpath itself as a part to be listed and does not list the parts within a sheetpath.

❑ `Place each part entry on a separate line`

Tells **Create Bill of Materials** to put each part entry on a separate line in the summary.

❑ `Verbose report`

Tells **Create Bill of Materials** to include the title block information in the report.

Select one of the following options:

○ `Insert a header for each page`

○ `Do not insert a header for each page`

These options specify whether or not to insert a header on each page. The header includes information such as the title of the design, the date, the document number, the revision code, the report name, the page number (if you select **Insert a header for each page**), and the time the report is created.

Select one of the following options:

`Report is`

- O `single-spaced`
- O `double-spaced`

These options specify whether the report is single- or double-spaced.

If desired, select either of these options:

- ❑ `Report un-used match strings in an include file`

  Tells **Create Bill of Materials** to report mismatches between an include file and a schematic. You can use **Edit File** to look at the text file #ESP_OUT.TXT for a list of strings in the include file which don't have a corresponding match string in the schematic worksheet. This list can be used to find duplicate match strings in an include file. The destination file lists schematic parts that don't have a corresponding match in the include file.

  This option is not available until you select **Merge an include file with report**.

- ❑ `Ignore warnings`

  Prevents **Create Bill of Materials** from displaying a message when it encounters warnings.



Show Design Structure

Show Design Structure scans the schematics in a design to display the sheet names and their associated worksheet filenames.

Execution

With the **Schematic Design Tools** screen displayed, select **Show Design Structure**. Select **Execute** from the menu that displays.

While **Show Design Structure** runs, messages display in the monitor box at the bottom of the screen. When the report is complete, the **Schematic Design Tools** screen appears.

Sample output

Figure 31-1 shows a sample report created by **Show Design Structure**.

```
<<<root>>>
[EX6.SCH]     October 11, 1990
  halfadd_A
  [ex6b.sch]     October 11, 1990
  halfadd_B
  [ex6b.sch]     October 11, 1990
```

Figure 31-1. Report created by Show Design Structure.

PART VI: TRANSFERS

Schematic Design Tools 386+ includes Transfer tools that manage the steps needed to move design information from one tool set to another. Transfers update the database then change from **Schematic Design Tools 386+** to other OrCAD tool set screens.

Part VI describes Transfer tools and provides instructions for their use.

Chapter 32: *To PLD* describes the transfer to the **Programmable Logic Design Tools** screen.

Chapter 33: *To Digital Simulation* describes the transfer to the **Digital Simulation Tools** screen.

Chapter 34: *To Layout* describes the transfer to the **PC Board Layout Tools** screen.

Chapter 35: *To Main* describes the transfer to the main ESP design environment screen.



To PLD

The **To PLD** transfer consists of three processes that update the database so that the design may be viewed by the **Programmable Logic Design Tools**. These processes are:

- FLDSTUFF (**Update Field Contents**) loads information you define into the fields of parts on a specified schematic.

  FLDSTUFF constructs a string from the key field designators for a specified field. Then, if that string equals a match field in a designated update file, it replaces the specified field with the update value.

  For more information about this process, see *Chapter 14: Update Field Contents.*

- ANNOTATE (**Annotate Schematic**) scans a design and automatically updates the reference designators of all parts in the worksheet.

  ANNOTATE updates reference designators in the order they were placed in the worksheet. When the worksheet is annotated, all parts may be assigned a new reference designator, including any manually edited parts. To selectively change reference designators and leave others unmodified, use **Draft**.

  For more information about this process, see *Chapter 6: Annotate Schematic.*

- EXTRACT (the **Extract Source from Schematic** processor on the **Programmable Logic Design Tools** screen) extracts PLD source code from a schematic for use with **Programmable Logic Design Tools**. The program reads a .SCH file and produces one or more .PLD files, adding pin information and other coding in the process.

  For more information about this process, see *Chapter 8: Using the Extract Source from Schematic processor* in the *Programmable Logic Design Tools User's Guide.*

Execution

To PLD reads the schematic database and updates the PLD source files for all devices found in the design. The design may be a complex or simple hierarchy, or a flat design.

△ *NOTE: If the source code for the PLD logic is a schematic, it should be created and edited in the **Programmable Logic Design Tools** area with the **Edit Schematic Logic** button. While technically the schematic for the internals of the PLD is part of the design, it is isolated by the fact that it represents the internal logic of the device. The schematics for the board, on the other hand, represent the external view of the logic for the design.*

Running To PLD

With the **Schematic Design Tools** screen displayed, select **To PLD**. Select **Execute** from the menu that displays.

When the transfer process is complete, the **Programmable Logic Design Tools** screen displays (figure 32-1).

The **Edit Schematic Logic** and **Convert Schematic Logic to Source** buttons are preconfigured at OrCAD with the correct libraries for logic development and the correct settings for the PLD compiler.

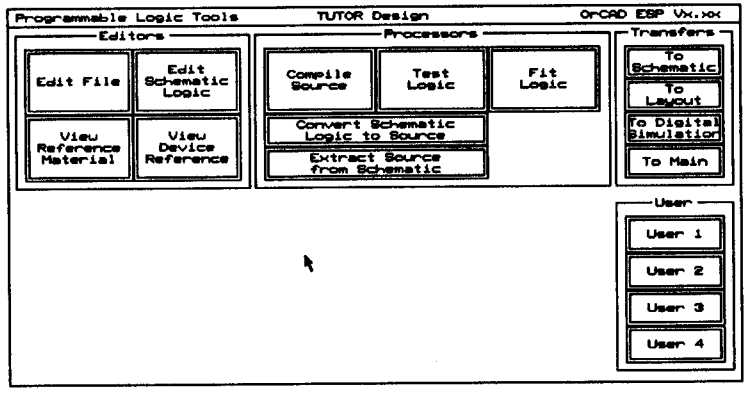


*Figure 32-1. **Programmable Logic Design Tools** screen.*

Local Configuration of To PLD

Since FLDSTUFF, ANNOTATE, and EXTRACT are each configured individually, **To PLD** has three configuration screens. To configure **To PLD**, select **To PLD**. Select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose which **To PLD** process to configure or to turn a process on or off. When you run **To PLD**, only the processes that are turned on run. In most cases, you will have EXTRACT turned on and the other processes turned off.

```
Configure FLDSTUFF
Configure ANNOTATE
Configure EXTRACT
FLDSTUFF off
ANNOTATE off
EXTRACT  on
```

To turn a process on or off, choose the desired process from the menu. For example, to turn FLDSTUFF on, select **FLDSTUFF off** from the menu. **Schematic Design Tools 386+** displays:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **on** to turn the process on.

Configure FLDSTUFF

With the **Schematic Design Tools** screen displayed, select **To PLD**. Select **Local Configuration** from the menu that displays, and then select **Configure FLDSTUFF**. The local configuration screen for FLDSTUFF displays. This is the same local configuration screen as for **Update Field Contents**. For information about configuring FLDSTUFF, see *Local Configuration* in *Chapter 14: Update Field Contents*, substituting references to **Update Field Contents** with FLDSTUFF.

Configure ANNOTATE

With the **Schematic Design Tools** screen displayed, select **To PLD**. Select **Local Configuration** from the menu that displays, and then select **Configure ANNOTATE**. The local configuration screen for ANNOTATE displays. This is the same local configuration screen as for **Annotate Schematic**. For information about configuring ANNOTATE, see *Local Configuration* in *Chapter 6: Annotate Schematic*, substituting references to **Annotate Schematic** with ANNOTATE.

Local Configuration of EXTRACT

With the **Schematic Design Tools** screen displayed, select **To PLD**. Select **Local Configuration** from the menu that displays, and then select **Configure EXTRACT**. A configuration screen appears (figure 32-2).

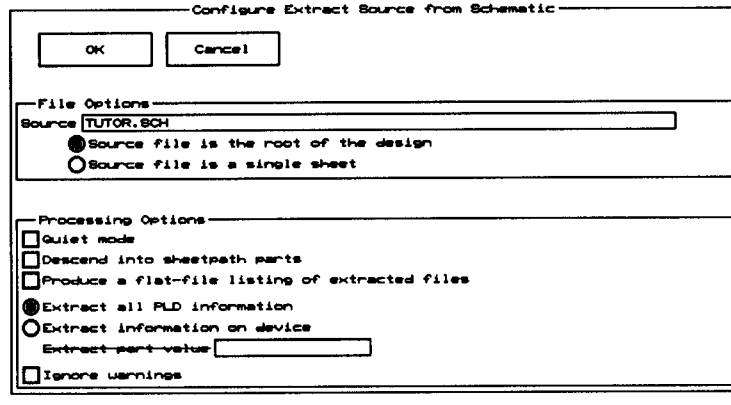


Figure 32-2. EXTRACT's local configuration screen.

File Options

File Options defines the name and type of the schematic from which to extract the PLD information.

Source

The source file is the root sheet filename of the design or the filename of a single sheet. It may have any valid pathname. After entering the source filename, select one of the following:

- `Source file is the root of the design`

  Specifies that the source file is the root sheet name of a hierarchical or flat design. If the root sheet contains sheet symbols, then the design is hierarchical. If it contains a I Link, it is a flat design.

- `Source file is a single sheet`

  Specifies that the source file is a single worksheet and you want to process the single sheet only.

Processing Options

You may select any of the following options:

- ☐ `Quiet mode`

  Turns quiet mode on.

- ☐ `Descend into sheet path parts`

  Tells EXTRACT to descend into parts defined as sheetpath parts. That is, it treats a sheetpath part as a sheet. This option, while designed for complex hierarchies, may be used in simple hierarchies. It is not recommended to be used except for FPGA, ASIC, or other designs that require a complex hierarchy.

- ☐ `Produce a flat-file listing of extracted files`

  Tells EXTRACT to produce a text file listing all the PLD files in the design. The name of the text file is taken from the name of the design (the directory) with an extension of .FLP. This file can be used as the source file in the local configuration of PLD's **Compile Source**.

Select one of the following:

- ○ `Extract all PLD information`

  Causes EXTRACT to extract all OrCAD Hardware Description Language (OHDL) source code from the schematic and produce one .PLD file for each device.

- ○ `Extract information on device`

  Causes EXTRACT to extract the OHDL source code for one specific device. Selecting this option makes the **Extract part value** entry box active. Specify the device to extract by entering its part value (the value that appears on the schematic for the particular device) in the entry box.

You may select this option:

- ☐ `Ignore warnings`

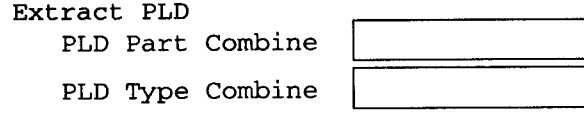
  Causes EXTRACT to continue running when it encounters warnings, instead of halting in the middle of execution.

About EXTRACT

The names and number of output files EXTRACT creates depend on specially formatted text (also called *source code*) that you place in your worksheet. This source code specifies information about a part in the **Programmable Logic Design Tools** language.

Key fields

EXTRACT uses two key fields. They are shown on the **Configure Schematic Design Tools** screen as:



When EXTRACT creates a .PLD file, it uses these key fields to fill in the Part: and Type: data fields. The **PLD Part Combine** data is enclosed in quotation marks and placed after the word "Part:" The **PLD Type Combine** data is enclosed in quotation marks and placed after the word "Type:" Figures 32-3 and 32-4 show an example.

More information about key fields for EXTRACT is found in *Chapter 1: Configure Schematic Tools.*

Unified documentation

The source code for a PLD is compiled separately from the schematic. To keep all documentation about a design together, though, it is a good idea to place the code on the schematic. It is also easier to write the source code for a PLD as part of the schematic, because EXTRACT will create pin definitions and title information for you.

In a schematic, PLD source code is placed on the same sheet as the schematic symbol for the programmable device. The source code can be anywhere on the sheet.

Make a custom symbol

Unlike nonprogrammable devices that have the same behavior every time they are used in a design, the behavior of programmable devices varies as the internal logical configuration changes. So, to represent a PLD in a design, you first need to create an appropriate symbol for it. The schematic library MEMORY.LIB contains generalized symbols for programmable devices, with values like `16R6` and pin names like `I1`, `I2`, and `O1`.

To customize a part to fit your application, use **Edit Library** to modify the pin names and device names to be the values for this specific application. When you create a custom part, it is always best to store it in your own custom library. That way, if you receive new libraries from OrCAD, you can replace your old libraries without fear of erasing your custom parts.

Defining the PLD's internal logic

Using OrCAD's **Programmable Logic Design Tools**, you can define logic in several different ways. You can use Boolean equations, indexed equations, numerical maps, state machine procedures, truth tables, streams, or schematics. See the *Programmable Logic Design Tools Language Guide* for more information.

The logic definition for the PLD consists of a series of text objects placed on the schematic. The text may be placed line-by-line with **Draft's PLACE Text** command or may be created using **Edit File** and placed on the schematic using the **BLOCK ASCII Import** command.

PLD logic definitions must begin with a key statement:

|PLD ***part***

Use **Draft's PLACE Text** command to add the statement to the schematic.

This is done by moving the pointer to an open area on your schematic. Select **PLACE**, then **Text**. Type the pipe or vertical bar symbol (|) followed by **PLD** and the name of the custom part:

|PLD ***name***

If more than one PLD is defined on this schematic, create a |PLD key statement for each one. EXTRACT creates a .PLD file for each part specified in the schematic. Each .PLD file contains the source code associated with its corresponding part.

For example, suppose you have included the source code for two different PLDs (A and B) in your schematic. When you run EXTRACT, two files are created: A.PLD and B.PLD. If either file already exists, EXTRACT renames the original file with a .BAK extension before creating the new .PLD file. For example, if A.PLD already exists, it is renamed A.BAK before the new A.PLD is created.

You can add comments, too. Just enter each line individually and place it in the schematic so that the first character of the line is in the same column as the pipe symbol of the device name line (see figure 32-7). Comment lines (those lines that do not start with a pipe character) are ignored by the **Programmable Logic Design Tools** compiler.

Select a device

Selecting a device is a decision based primarily on two factors: design complexity and cost. You base your decision on the number of inputs and outputs needed, how complex the logic is that the device needs to handle, and how much the device costs. Just exactly *when*, in the design process, you have to choose a device, is a matter for you, the designer, to determine. For example, you may want to define the logic completely, test it to be sure it performs as you expect, and then pick a device that accommodates the logic. On the other hand, you may have a device in mind before you begin designing the logic—one that for cost or availability reasons is your definite choice.

Record part type and value on the schematic

After you have selected a device, record your choice on the schematic in the device symbol's part fields. Typically, the **1st Part Field** is used to specify the PLD type (22V10, for example) and the **2nd Part Field** is used to specify the manufacturer's exact part number—but you can use any pair of the eight part fields to hold this information. EXTRACT uses this information to create a PLD header file.

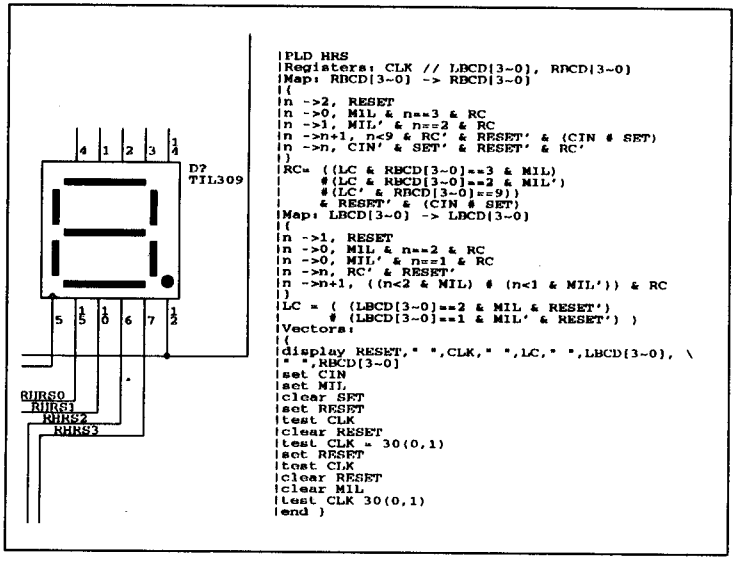


Figure 32-3. Programmable logic device.

When you run EXTRACT on the schematic containing the part, it creates a file called HRS.PLD. The contents of HRS.PLD are shown in figure 32-4. In the HRS.PLD output, the **PLD Type Combine** key field **(1st Part Field)** is shown in ***bold italics***, and the **PLD Part Combine** key field **(2nd Part Field)** is shown in **bold**.

```
|"HRS"    1:CLK,
|         2:-,
|         3:-,
|         4:-,
|         5:-,
|         6:-,
|         7:-,
|         8:-,
|         9:MIL,
|        10:SET,
|        11:CIN,
|        13:RESET,
|        23:LC,
|        22:LBCD3,
|        21:LBCD2,
|        20:LBCD1,
|        19:LBCD0,
|        18:RC,
|        17:RBCD3,
|        16:RBCD2,
|        15:RBCD1,
|        14:RBCD0
|
|Value:    "HRS"
|Type:     "22V10"
|Part:     "PALC22V10-35"
|Library: "EXAMPLE.LIB"
|
|Title:    "Example schematic"
|Title:    " November  5, 1990"
|
|Registers: CLK // LBCD[3~0], RBCD[3~0]
|Map: RBCD[3~0] -> RBCD[3~0]
|{
|n ->2, RESET
|n ->0, MIL & n==3 & RC
|n ->1, MIL' & n==2 & RC
|n ->n+1, n<9 & RC' & RESET' & (CIN # SET)
|n ->n, CIN' & SET' & RESET' & RC'
|}
|RC= ((LC & RBCD[3~0]==3 & MIL)
|     #(LC & RBCD[3~0]==2 & MIL')
|     #(LC' & RBCD[3~0]==9))
|     & RESET' & (CIN # SET)
|Map: LBCD[3~0] -> LBCD[3~0]
|{
|n ->1, RESET
```

Figure 32-4. Sample EXTRACT output (continued on next page).

```
|n ->0, MIL & n==2 & RC
|n ->0, MIL' & n==1 & RC
|n ->n, RC' & RESET'
|n ->n+1, ((n<2 & MIL) # (n<1 & MIL')) & RC
|}
|LC = ( (LBCD[3~0]==2 & MIL & RESET')
|      # (LBCD[3~0]==1 & MIL' & RESET') )
|Vectors:
|{
|display RESET," ",CLK," ",LC," ",LBCD[3~0], \
|" ",RBCD[3~0]
|set CIN
|set MIL
|clear SET
|set RESET
|test CLK
|clear RESET
|test CLK = 30(0,1)
|set RESET
|test CLK
|clear RESET
|clear MIL
|test CLK 30(0,1)
|end }
```

Figure 32-4. Sample EXTRACT output (continued from previous page).



To Digital Simulation

The **To Digital Simulation** transfer consists of four processes that update the connectivity database and simulation directives for the design. These processes are:

- ANNOTATE (**Annotate Schematic**) scans a design and automatically updates the reference designators of all parts in the worksheet.

  ANNOTATE updates reference designators in the order they were placed in the worksheet. When annotating the worksheet, you may assign all parts a new reference designator, including any manually edited parts. To selectively change reference designators and leave others unmodified, use **Draft**.

  For more information about this process, see *Chapter 6: Annotate Schematic*.

- INET (**Create Netlist**) creates or updates the incremental connectivity database for the design. INET is an incremental processor. It updates the database only for those worksheets that have changed.

- IBUILD (**Build Simulation Specification File**) extracts the trace, stimulus, and test vector information from the connectivity database, creating text trace (.ATR) and stimulus (.AST) files.

- ASCTOVST (**Compile Simulation Specification File**) reads the text file IBUILD creates, converts it to binary format, and copies the results to a breakpoint, stimulus, or trace file. ASCTOVST displays twice on the configuration screen, so you can convert both the stimulus and trace files when transferring to **Digital Simulation Tools**.

Execution

To Digital Simulation updates the schematic database, the connectivity database, and the simulation directives. The design may be a complex or simple hierarchy, or a flat design.

Running To Digital Simulation

With the **Schematic Design Tools** screen displayed, select **To Digital Simulation**. Select **Execute** from the menu that displays.

During the transfer process, a monitor box displays at the bottom of the screen where messages report the progress of transfer tasks.

When the transfer process is complete, the **Digital Simulation Tools** screen displays (figure 33-1).

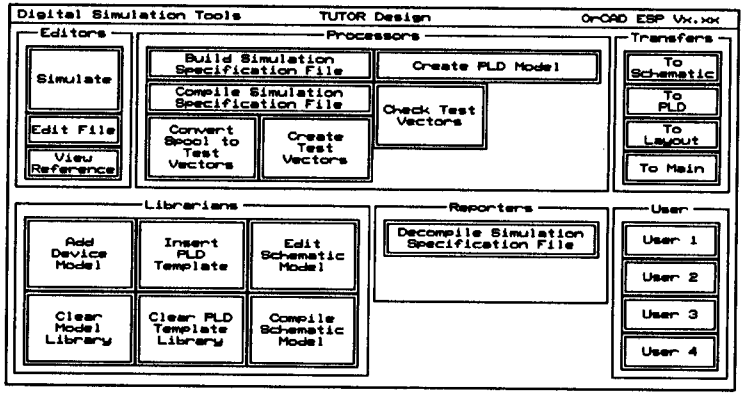


Figure 33-1. Digital Simulation Tools screen.

Local Configuration of To Digital Simulation

Since you can configure ANNOTATE, INET, IBUILD and each ASCTOVST individually, **To Digital Simulation** has five configuration screens. To configure **To Digital Simulation**, select **To Digital Simulation**. Select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose which **To Digital Simulation** process to configure or to turn a process on or off. When you select **To Digital Simulation**, only the processes that are turned on will run.

```
Configure ANNOTATE
Configure INET
Configure IBUILD
Configure ASCTOVST
Configure ASCTOVST
ANNOTATE  off
INET      on
IBUILD    off
ASCTOVST  off
ASCTOVST  off
```

For most cases, you will have INET turned on and the other processes turned off. If you are making extensive changes to trace, stimulus, or vector objects in the schematic, turn the IBUILD and ASCTOVST processes on.

To turn a process on or off, choose the desired process from the menu. For example, to turn ANNOTATE on, select **ANNOTATE off** from the menu. **Schematic Design Tools 386+** displays:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **on** to turn the process on.

Configure ANNOTATE

With the **Schematic Design Tools** screen displayed, select **To Digital Simulation**. Select **Local Configuration** from the menu that displays, and then select **Configure ANNOTATE**. The local configuration screen for ANNOTATE displays. This is the same local configuration screen as for **Annotate Schematic**. For information about configuring ANNOTATE, see *Local Configuration* in *Chapter 6: Annotate Schematic*, substituting references to **Annotate Schematic** with ANNOTATE.

Configure INET

With the **Schematic Design Tools** screen displayed, select **To Digital Simulation**. Select **Local Configuration** from the menu that displays, and then select **Configure INET**. The local configuration screen for INET displays. This is the same local configuration screen as for INET in the **Create Netlist** tool. For information about configuring INET, see *Configure INET* in *Chapter 11: Create Netlist*.

Configure IBUILD

With the **Schematic Design Tools** screen displayed, select **To Digital Simulation.** Select **Local Configuration** from the menu that displays, and then select **Configure IBUILD.** A configuration screen displays (figure 33-2).

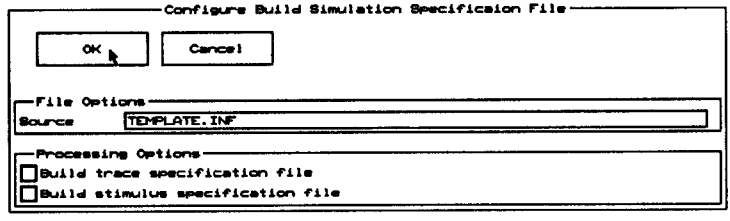


Figure 33-2. IBUILD's local configuration screen.

File Options — **File Options** defines the source file from which the simulation directives are to be obtained. This source file is the incremented connectivity database.

Source — The **Source** is the connectivity database. It has an extension of .INF and a base name that is the same as the project. It may have any valid pathname.

Processing Options — Select any combination of the following:

- ❑ `Build trace specification file`

  Causes the trace objects in the schematic database to be updated as a new trace specification file for the simulation.

- ❑ `Build stimulus specification file`

  Causes the stimulus and vector (a vector is a form of a stimulus specification) objects in the schematic database to be updated as a new stimulus specification file for the simulator.

Configure ASCTOVST

With the **Schematic Design Tools** screen displayed, select **To Digital Simulation.** Select **Local Configuration** from the menu that displays, and then select the one of the two instances of ASCTOVST. A configuration screen displays.

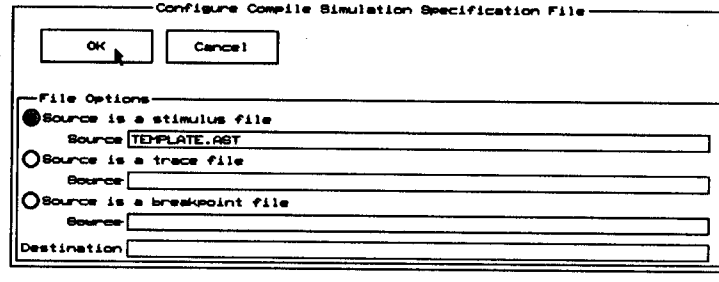


Figure 33-3. ASCTOVST's local configuration screen.

ASCTOVST displays twice so you can convert both the stimulus and trace files when transferring to **Digital Simulation Tools.** Usually, you configure the first instance to select **Source is a stimulus file** and the second instance to select **Source is a trace file.**

File Options **File Options** specifies the source and destination filenames.

Source Select one of the following options:

- ○ `Source is a stimulus file`

  The source is a stimulus file. Enter the destination path and filename. The default is your design directory name followed by an .AST extension. Select this default file option for the first instance.

- ○ `Source is a trace file`

  The source is a trace file. Enter the destination path and filename. The default is your design directory name followed by an .ATR extension. Select this file option for the second instance.

O `Source is a breakpoint file`

The source is a breakpoint file. Enter the destination path and filename. The default is your design directory name followed by an .ABR extension.

Destination **The directory path and filename for the output file. Depending on the Source option selected, the default Destination changes, as shown in table 33-1.**

Option	*Source*	*Destination*
O `Source is a stimulus file`	`FILE.AST`	`FILE.STM`
O `Source is a trace file`	`FILE.ATR`	`FILE.TRC`
O `Source is a breakpoint file`	`FILE.ABR`	`FILE.BRK`

*Table 33-1. **Source and Destination** automatically defined on **Compile Simulation Specification File's** local configuration screen.*



To Layout

The **To Layout** transfer consists of three processes that are used to update the connectivity database and the layout directives given in the schematics. These processes are:

- INET updates the connectivity database for the design. INET updates the database only for those worksheets that have been changed.
- ILINK links the incremental database into a flattened database. This flattened database contains information on the connectivity, parts, fields, pin typing information, and the layout directives.
- IFORM creates a netlist in any of over thirty netlist formats, and then displays the **PC Board Layout Tools 386+** screen.

Execution

To Layout updates the schematic database, the connectivity database, and the layout directives used in **PC Board Layout Tools 386+**. The design may be a simple hierarchy or a flat design. If the design is a complex hierarchy, the **Complex to Simple** processor in **Design Management Tools** must be used to simplify the design hierarchy before you select **To Layout**.

Running To Layout

With the **Schematic Design Tools** screen displayed, select **To Layout**. Select **Execute** from the menu that displays.

When the transfer process is complete, the **PC Board Layout Tools** screen displays.

Local Configuration of To Layout

Since INET, ILINK and IFORM are configured individually, **To Layout** has three configuration screens. To configure **To Layout**, select **To Layout**. Select **Local Configuration** from the menu that displays.

The menu shown at right displays. Use this menu to choose which **To Layout** process to configure and to turn a process on or off. When you select **To Layout**, only the processes that are turned on run. In most cases, you will have all three processes turned on.

```
Configure INET
Configure ILINK
Configure IFORM
INET      on
ILINK     on
IFORM     on
```

To turn a process on or off, choose the desired process from the menu. For example, to turn INET off, select **INET on** from the menu. **Schematic Design Tools 386+** displays:

```
Select the new status of the executable item
```

A menu with the options **on** and **off** displays. Select **off** to turn the process off.

Configure INET

With the **Schematic Design Tools** screen displayed, select **To Layout**. Select **Local Configuration** from the menu that displays, and then select **Configure INET**. The local configuration screen for INET displays. This is the same local configuration screen as for INET in the **Create Netlist** tool. For information about configuring INET, see *Configure INET* in *Chapter 11: Create Netlist*.

Configure ILINK

With the **Schematic Design Tools** screen displayed, select **To Layout**. Select **Local Configuration** from the menu that displays, and then select **Configure ILINK**. The local configuration screen for ILINK displays. This is the same local configuration screen as for ILINK in the **Create Netlist** tool. For information about configuring ILINK, see *Configure ILINK* in *Chapter 11: Create Netlist*.

Configure IFORM

With the **Schematic Design Tools** screen displayed, select **To Layout**. Select **Local Configuration** from the menu that displays, and then select **Configure IFORM**. The local configuration screen for IFORM displays. This is the same local configuration screen as for IFORM in the **Create Netlist** tool.

Be sure to select FEDIF.EXE as the netlist format when you transfer a design to **PC Board Layout Tools 386+**. For more information about configuring IFORM, see *Configure IFORM* in *Chapter 11: Create Netlist*.



To Main

The **To Main** tool transfers from the **Schematic Design Tools** screen to the ESP design environment's main screen. This tool does not process or create any files and has nothing for you to configure.

Execution

With the **Schematic Design Tools** screen displayed, select **To Main**. Select **Execute** from the menu that displays. The ESP design environment's main screen (figure 35-1) displays.

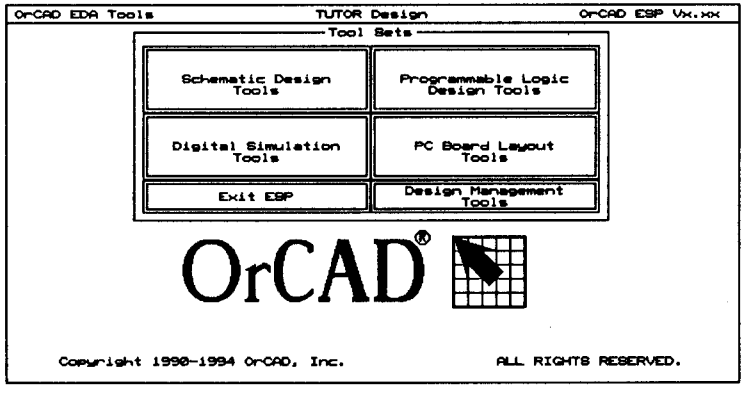


Figure 35-1. ESP design environment main screen.

The Appendixes provide reference information in the following areas:

Appendix A: *Command line controls* cross references command line commands and their switches with their corresponding tools and local configuration buttons.

Appendix B: *Netlist formats* defines each of the custom netlist format files provided with **Schematic Design Tools 386+.**

Appendix C: *Interpreting connectivity databases* describes the format of the incremental connectivity database created by INET.

Appendix D: *Creating a custom netlist format* describes how to create a custom netlist format file.

Appendix E: *Plotter information* provides additional information you may need to set up, configure, and use your plotter with **Schematic Design Tools 386+.**

Appendix F: *Files and file extensions* gives descriptions of the file extensions used by **Schematic Design Tools 386+.**

Appendix G: *Converting designs, worksheets, and libraries* tells how to translate files created in **Schematic Design Tools IV** to the format used by **Schematic Design Tools 386+.**



Command line controls

This appendix cross references command line commands and their switches with their corresponding tools and local configuration buttons. This appendix is organized in alphabetical order by command.

△ ***NOTE:*** ***Schematic Design Tools 386+*** *does not support the /F switch. Many version III utilities used the /F switch to indicate that the source file was a text file containing a list of files to be processed in a flat file structure.* ***Schematic Design Tools 386+*** *uses the !LINK command to list the files. For more information, see* The !LINK command *in* Chapter 10: Creating a netlist.

△ ***NOTE:*** *Many of these tools allow you to display the SDT configuration screen using the /C switch. This option is operational only if the OrCAD display driver (ORCADDD) environment variable identifies the location and name of the display driver used to display the configuration screen. Your AUTOEXEC.BAT file should include a line similar to the following:*

```
set orcaddd=c:\orcadesp\drv\ega16e.drv
```

Syntax

The syntax in this appendix follows this format:

- Parameters that you must enter exactly as shown are given in `monospace` font.
- Variables that you must supply, such as filenames, are shown in *italic* text.
- Items in brackets are optional, and you include them only in specific circumstances. Do not type the brackets.

AIM32 *source* [/Q]

Corresponding tool: **Compile Schematic**

Switch	*Description*	*Local configuration button*
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode

AEX32 *source* [*switches*]

Corresponding tool: **Decompile Schematic**

Switch	*Description*	*Local configuration button*
/D	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/O	Treat this file as a single sheet.	❑ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode

ANNOTATE *source [switches]*

Corresponding tool: **Annotate Schematic**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/H	Reset reference numbers to begin with 1 for each sheet of the hierarchy.	❑ Reset reference numbers to begin with 1 on each sheet of the hierarchy
/L	Create a report listing the last reference designators assigned by **ANNOTATE**. If a destination file is not specified, the report is placed in a text file with the same filename as the root worksheet and a file extension of .END.	❑ Report the last assigned reference values
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/R	Unannotate the schematic. All reference numbers are set to "?" and all reference letters (for multiple parts per package) are set to "A."	❑ Unannotate Schematic
/S	Do not change the sheet number in the title block. If this switch is not set then the sheet numbers are changed to reflect the current sheet in the design.	❑ Do NOT change the sheet number
/U	Change references unconditionally. **Annotate** updates reference designators in the order in which they were placed on the worksheet. If you add a part and run **Annotate /U**, all the reference designators are updated.	○ Unconditional annotation (update all reference designators)
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

BACKANNO *source was/is* [*switches*]

Corresponding tool: **Back Annotate**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

CLEANUP *source* [*destination*] [*switches*]

Corresponding tool: **Cleanup Schematic**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/E	Remove error markers placed on the schematic by INET /W.	❑ Remove error objects from schematic sheet(s)
/G	Report items found to be off grid.	❑ Report off-grid items
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

COMPOSER *source destination [switches]*

Corresponding tool: **Compile Library**

Switch	*Description*	*Local configuration button*
/N	Do not sort library as it is compiled. Instead save devices in the order in which they appear in the source file.	❑ Do not sort the library
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered. The only warning COMPOSER will encounter is that it will truncate pin names that exceed twenty-nine characters and then issue a warning.	❑ Ignore warnings

CROSSREF

Corresponding tool: **Cross Reference Parts**

CROSSREF no longer exists as a separate program. It is now `PARTLIST /X`. See *PARTLIST* in this appendix for details.

DECOMP *library source [/Q]*

Corresponding tool: **Decompile Library**

Switch	*Description*	*Local configuration button*
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode

DRAFT *filename [switches]*

Corresponding tool: **Draft**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/M	Disable the mouse.	❑ Disable mouse
/P	Disable **Draft**'s <Print Screen> key function.	❑ Disable <Print Screen> key function
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/S	Slow the mouse. Used for mice that are too sensitive.	❑ Decrease mouse sensitivity
/Y	Reverse the "Y" axis operation of the mouse.	❑ Reverse "Y" axis operation of the mouse

ERC

Corresponding tool: **Check Electrical Rules**

ERC no longer exists as a separate program. It is now `INET /W`. See *INET* in this appendix for details.

EXTRACT *source* [*switches*]

Corresponding tool: **Extract Source from Schematic, in To PLD**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/D	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/O	Treat this file as a single sheet.	○ Source file is single sheet
/P	Extract PLD information.	○ Extract all PLD information
/Q	Run in the "quiet" mode.	❑ Quiet mode
/R	Create a flat file listing of extracted files.	❑ Produce a flat-file listing of extracted files
/S <arg>	Extract information about the device specified by <arg>.	○ Extract information on device Extract part value [ ]
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

FLDATTRB *source partField* [*switches*]

Corresponding tool: **Select Field View**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/I	Set specified field to invisible.	○ Set the specified field to invisible
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/U	Unconditionally set visibility parameter, even if nothing is in the field.	❑ Unconditionally set attribute
/V	Set specified field to visible.	○ Set the specified field to visible
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

FLDSTUFF *source partFieldToUpdate updateFile [reportFile] [switches]*

Corresponding tool: **Update Field Contents**

Switch	*Description*	*Local configuration button*
/C	Display the configuration screen.	None
/F	Do not convert key field match string to uppercase.	❑ Do not convert key field match string to uppercase
/I	Specify that this field will be invisible.	○ Set the specified field to invisible
/K	Maintain the current visibility setting of updated fields.	○ Leave visibility of specified field unaltered
/N	Do not convert update string to uppercase.	❑ Do not convert update string to uppercase
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode.	❑ Quiet mode
/R	Create a report of all **FLDSTUFF** activity. Enter the name of the report file to use after entering the name of the stuff file.	❑ Create an update report Destination [ ]
/U	Unconditionally change this field. Normally it is only changed if it is empty.	❑ Unconditionally update field (Normally stuffed only if empty)
/V	Specify that this field will be visible.	○ Set the specified field to visible
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

HFORM *source format destination [destination2] [switches]*

Corresponding tool: **Create Hierarchical Netlist**

HFORM is Create Hierarchical Netlist's incremental hierarchical netlist formatter. An extension of .INF is assumed for the source, so you need not furnish an extension. The format file is the path and filename of the netlist format file. OrCAD hierarchical netlist format filenames begin with "H" and have an extension of .EXE. A destination must be specified. If the netlist format requires two output files, destination 2 must be present on the command line.

Some of the HFORM options apply only to specific netlists, shown in parentheses.

Switch	*Description*	*Local configuration button*
/A	(EDIF only) Allow characters which are not standard to the EDIF format. Requires an EDIF reader which can process non-standard characters.	❑ Allow nonEDIF characters
/B	Bypass the .INX check and force a link even if it is not required.	❑ Force HFORM to always create a formatted netlist
/E	(EDIF only) Do not list libraries that are not defined in the EDIF netlist.	❑ Do not create the 'external' library declaration
/I	(SPICE only) Assign all unconnected pins a unique net.	❑ Include unconnected pins
/K	(EDIF only) Replace the field name (such as PartField1) with the user-defined property name. These part field strings must have the form *propertyName=value*. Use this option if you are creating a netlist for an EDIF reader other than **PC Board Layout Tools 386+.**	❑ Convert PartFields to Named Properties
/L	(SPICE, EDIF) Do not add the sheet number to label descriptions.	❑ Do not append sheet number to labels
/N	(SPICE only) Use node names.	❑ Use node names
/N	(EDIF only) Do not include timestamps and Part Fields 1 through 7 plus Module Value in the netlist.	❑ Do not output additional properties
/P	(EDIF only) Output pin numbers instead of pin names.	❑ Output pin numbers (instead of pin names)
/Q	Run in "quiet" mode.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

IFORM *source format [destination] [destination2] [switches]*

Corresponding tool: **Create Netlist**

IFORM is **Create Netlist**'s incremental netlist formatter. An extension of .INF is assumed for the source, so you need not furnish an extension. The format file is the path and filename of the netlist format file. OrCAD flat netlist format filenames begin with "F" and have an extension of .EXE. A destination must be specified. If the netlist format requires two output files (for example, RacalRedac), destination 2 must be present on the command line.

Some of the IFORM options apply only to specific netlists, shown in parentheses.

Switch	*Description*	*Local configuration button*
/A	(Wirelist only) Abbreviate label descriptions.	❑ Abbreviate label descriptions
/A	(EDIF only) Allow characters which are not standard to the EDIF format. Requires an EDIF reader which can process non-standard characters.	❑ Allow nonEDIF characters
/B	Bypass the .INX check and force a link even if it is not required.	❑ Force IFORM to always create a formatted netlist
/E	(EDIF only) Do not list libraries that are not defined in the EDIF netlist.	❑ Do not create the 'external' library declaration
/F	(EDIF, PADS2K, PADSPCB) Create a persistent netname (.PNN) file. If the .PNN file already exists, match any unnamed nets in the new version with their counterparts in the existing version. See the description of F*formatName*.EXE in *Appendix B: Netlist formats* for more information.	❑ Keep system generated net names
/I	(SPICE AlteraADF, IntelADF, Case, Hilo, PCAD) Assign all unconnected pins a unique net.	❑ Include unconnected pins
/K	(EDIF only) Replace the field name (such as PartField1) with the user-defined property name. These part field strings must have the form *propertyName=value*. Use this option if you are creating a netlist for an EDIF reader other than **PC Board Layout Tools 386+.**	❑ Convert PartFields to Named Properties

continued on next page

IFORM *source format* [*destination*] [*destination2*] [*switches*]

(continued from previous page)

Switch	*Description*	*Local configuration button*
/K	(FutureNet only) Create CON* symbols for module ports.	❑ Create CON* symbols for module ports
/L	(All formats except DUMP, EEDesigner, PDUMP, and VSTModel) Do not append the sheet number to labels.	❑ Do not append sheet number to labels
/M	(FutureNet only) Assign SIG* attributes to module ports.	❑ Assign SIG* attributes to module ports
/N	(FutureNet only) Create a netlist instead of a pinlist.	❑ Create a netlist (instead of a pinlist)
/N	(AlteraADF, IntelADF, and VSTModel) Suppress comments.	❑ Suppress comments
/N	(EDIF only) Do not include time stamps and Part Fields 1 through 7 plus Module Value in the netlist.	❑ Do not output additional properties
/N	(SPICE only) Use node names.	❑ Use node names
/P	(EDIF only) Output pin names instead of pin numbers.	❑ Output pin names (instead of pin numbers)
/P	(FutureNet only) Output pin numbers instead of pin names.	❑ Output pin numbers (instead of pin names)
/P	(PLDNET only) Convert IN, OUT, and BIDIRECTIONAL schematic module ports into a SignalList associated with the OHDL PINS port name. Convert UNSPECIFIED schematic module ports into a SignalList associated with the OHDL NODES port name.	❑ Only use pins and nodes keywords (do not use in, out, and i/o)
/P	(Wirelist only) Skip any non-numerical pin numbers.	❑ Do not output pin numbers for Grid Array parts
/Q	Run in "quiet" mode.	❑ Quiet mode
/V	(FutureNet only) Assign FutureNet power attributes to power objects.	❑ Assign FutureNet power attributes to power objects
/W	(PLDNET only) Create a PLD source file in addition to a netlist. Any PLD vectors extracted from the schematic are placed in the PLD source file.	❑ Create a complete PLD source file
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

ILINK *source* [*switches*]

Corresponding tool: **Create Netlist**

ILINK is the incremental netlist linker. The source is the .INF file of the incremental netlist files to be linked. The linker creates two files with the name of the source and extensions of .INS (instance file) and .RES (joined resolved file). These files are used by **IFORM** to produce the final version of the netlist.

Switch	*Description*	*Local configuration button*
/B	Bypass the .INX check and force a link even if it is not required.	❑ Force ILINK to always link the database
/C	Display the SDT configuration screen.	None
/N	Reports a warning for nets with only one connection.	❑ Report single net nodes
/Q	Run in "quiet" mode.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

INET *source* [*destination*] [*switches*]

Corresponding tool: **Create Netlist** and **Create Hierarchical Netlist.**

Switch	*Description*	*Local configuration button*
/A	Do not check for the following error conditions: two power objects connected, single node nets, and input signals without a driving source. Instead check for rules as indicated in the **Check Electrical Rules Matrix.** Can only be used with /W.	❑ Do not report warnings
/B	Build the file stack from schematics rather than the .INX file.	❑ Rebuild file stack
/C	Display the SDT configuration screen.	None
/D	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/E	Generate a warning message if the Module Value field, or Part Field 8, is empty.	❑ Report empty module values
/G	Check worksheet for parts, sheets, labels, module ports, and power objects placed off grid. If a destination is supplied, then the output is placed in the indicated file, otherwise standard out is used.	❑ Report off-grid parts
/I	Assign a net name to all pins, even unconnected ones.	❑ Assign a net name to all pins
/L	Report all connected labels and module ports. If a destination is supplied, then the output is placed in the indicated file, otherwise standard out is used.	❑ Report all connected labels and ports
/M	Maintain any module ports which do not connect to a signal on another schematic. When this option is not selected, INET discards module ports which do not match a signal name on another schematic.	❑ Include unused signals connected to module ports
/N	Do not rebuild the .INF file.	❑ Do NOT create .INF files (Report only)
/O	Process one sheet only.	❑ Process one sheet only (This forces Rebuild file stack on next run)

continued on next page

INET *source* [*destination*] [*switches*]

(continued from previous page)

Switch	*Description*	*Local configuration button*
/Q	Run in "quiet" mode.	❑ Quiet mode
/R	Use the reference designator alone, rather than a combination of the reference designator and the part value, as the sheet name in the .INF file. This option can only be used in conjunction with the **Descend into sheetpath parts** option.	❑ Use reference designator as sheetname for sheetpath parts
/S	Netlist all sheet parts as though they were regular parts; do not descend into the sheets that they represent.	None
/T	Rebuild the entire database, recreate all connecting database files.	❑ Unconditionally process all sheets in design
/U	Report all unconnected wires and pins. This switch also checks for pins, module ports, and power objects that might be overlapping and reports them. The report is included in the destination file or is displayed on the screen.	❑ Report unconnected wires, pins, module ports
/W	Run an electrical rules check on all files that are netlisted. If a destination is supplied, then the output is placed in the indicated file, otherwise standard out is used. Module ports are checked for correctness after all incremental netlisting and electrical rules checking is complete.	❑ Run ERC on all sheets processed
/X	Check module ports against sheet nets for correctness. This is only done after all database and electrical rules checking is complete.	❑ Check module port connections
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

LIBARCH *source [destination] [switches]*

Corresponding tool: **Archive Parts in Schematic**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/D	Descend into any parts defined as sheetpath parts.	❑ Descend into sheetpath parts
/L	Make an ASCII file consisting of the names of all parts used in a schematic. This file is called a "string file." The names are delimited with single quotes, and each exists on a separate line.	○ Output is a string file
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/S	Treat the source file as a "string file." **LIBARCH** constructs a library source file, ready for use by **COMPOSER**. IEEE parts are not supported.	○ Source file is a string file
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

LIBEDIT *filename.LIB [switches]*

Corresponding tool: **Edit Library**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/M	Disable the mouse.	❑ Disable mouse
/P	Disable **LIBEDIT**'s <Print Screen> key function.	❑ Disable <Print Screen> key function
/S	Slow the mouse. Used for mice that are too sensitive.	❑ Decrease mouse sensitivity
/Y	Reverse the "Y" axis operation of the mouse.	❑ Reverse "Y" axis operation of mouse

LIBLIST *filename.LIB [destination] [switches]*

Corresponding tool: **List Libraries**

Switch	*Description*	*Local configuration button*
None	Create a list of parts by name, arranged in alphabetical order.	○ Output is a library list
/L	Create a string file as the report from the library. The single quotes around part names makes this report useful with FLDSTUFF.	○ Output is a string file
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/S	Print a report showing total number of devices in a library.	○ Report the total number of devices in the library

PARTLIST *source* [*destination*] [*include*] [*switches*]

PARTLIST performs the functions for either **Cross Reference Parts** (if the /X switch is present) or **Create Bill of Materials** (absence of the /X switch). Some of the switches described in the table below apply only when running a cross reference report, only when running a part list report, or when running either report. The columns at the right of the table (labeled XRF for **Cross Reference Part** and BOM for **Create Bill of Materials**) indicate when each switch applies.

Switch	*BOM*	*XRF*	*Description*	*Local configuration button*
/A	✓		Report the reference, value, and 8 part fields for all modules in comma-delimited format.	❑ Produce Comma Separated Value format
/C	✓	✓	Display the SDT configuration screen.	None
/D	✓	✓	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/E	✓		Report unused match strings in an include file.	None
/I	✓		Specify an include file to be added to the part list.	❑ Merge an include file with report Include [ ]
/K		✓	Create a report of type mismatches and identical reference designators. Do not create a cross-reference report.	❑ Report only type mismatch parts and identical reference designators
/L	✓		Specifies that each part starts on a new line.	❑ Place each part entry on a separate line
/M		✓	Append a report of name, type, page, and filename for all module ports. Issue warnings about any module name that appears once rather than twice. This report is always created for linked designs, and is optional for single-sheet and hierarchical designs.	❑ List module ports
/N		✓	Sort output by name, then reference designator. If neither /R nor /N is present, then both reports are done (one sorted by name and the other sorted by reference).	○ Sort output by part value, then by reference designator

continued on next page

PARTLIST *source [destination] [include] [switches]*

(continued from previous page)

Switch	*BOM*	*XRF*	*Description*	*Local configuration button*
/O	✓	✓	Treat this file as a single sheet.	○ Source file is a single sheet
/P		✓	Output the X,Y coordinates for all parts.	❑ Report the X and Y grid coordinates for all parts
/Q	✓	✓	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/R		✓	Sort output by reference designator, then name. If neither /R or /N is present, then both reports are done (one sorted by name and the other sorted by reference).	○ Sort output by reference designator
/S	✓	✓	Produce a single-spaced report rather than a double.	Report is ○ single-spaced
/T		✓	Report type mismatches for parts, such as combinations of single and multiple parts in a package (U1 and U1A).	❑ Report type mismatch parts
/U		✓	List all unused parts in multiple-part packages.	❑ Report unused parts in multiple-part packages
/V	✓		Output report in a verbose format.	❑ Verbose report
/W		✓	Check for identical part references. There should be none.	❑ Report identical part reference designators
/X		✓	Print cross reference report.	None
/Y	✓	✓	Turns off the header at the beginning of each page of the report.	○ Do not insert a header for each page
/Z	✓	✓	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings

PLOTALL *source* [*destination*] [*switches*]

Corresponding tool: **Plot Schematic**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/D	Descend into sheet path parts.	❑ Descend into sheetpath parts
/G	Plot grid references around the border of output.	❑ Plot grid references around the worksheet border
/N	Instruct **PLOTALL** to ignore "fill" commands.	❑ Ignore "fill" commands
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/P	Direct output to printer instead of plotter.	○ Send output to printer
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/R	Specify plotter using roll feed paper.	○ Plotter uses roll feed paper
/S	Specify a scale factor for plot output "##.###".	○ Manually set scale factor and/or X,Y offsets ❑ Set Scale factor [ ]
/U	Use offsets for another paper size (for example, /UA means to use A-size paper).	○ Automatically scale and set X, Y offsets for specified sheet size ○ A ○ B ○ C ○ D ○ E
/V	Plot rotated text and extended ASCII characters in the font used for "normal" text.	❑ Use stick text, not plotter font
/W	Specify wide paper in printer.	○ Printer has wide paper
/X	Specify an "X" offset for plot output "##.###".	○ Manually set scale factor and/or X,Y offsets ❑ Set X, Y offsets X [ ]
/Y	Specify a "Y" offset for plot output "##.###".	○ Manually set scale factor and/or X,Y offsets ❑ Set X, Y offsets Y [ ]

PRINTALL *source* [*destination*] [*switches*]

Corresponding tool: **Print Schematic**

Switch	*Description*	*Local configuration button*
/C	Display the SDT configuration screen.	None
/D	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/G	Print grid references around the border of the output.	❑ Print grid references around the worksheet border
/N	Suppress pin numbers.	❑ Suppress pin numbers on print
/O	Treat this file as a single sheet.	○ Source file is a single sheet
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/W	Specify wide paper in printer.	○ Printer has wide paper

SIMPLE *source destination [switches]*

Corresponding tool: **Complex to Simple in Design Management Tools**

You must run SIMPLE from the directory you are converting. *source* must be the root schematic of the design you are converting. The *destination* must already exist; SIMPLE will not create it.

Switch	*Description*	*Corresponding options (no local configuration)*
/C	Display the SDT configuration screen.	None
/N	Do not descend into sheetpath parts.	None
/Q	Run in "quiet" mode without echoing tracking information on the screen.	None
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	None

TREELIST *source destination [switches]*

Corresponding tool: **Show Schematic Structure**

Switch	*Description*	*Local configuration button*
/C	Display SDT configuration information.	None
/D	Descend into sheetpath parts.	❑ Descend into sheetpath parts
/Q	Run in "quiet" mode without echoing tracking information on the screen.	❑ Quiet mode
/Z	Cause warning messages to be ignored. In batch files, do not set ERRORLEVEL when warnings are encountered.	❑ Ignore warnings



Netlist formats

Usage

This appendix describes the various netlist format files that OrCAD provides. These format files are selected in the local configurations of **Create Netlist** and **Create Hierarchical Netlist**. See *Chapter 10: Creating a netlist*, *Chapter 11: Create Netlist*, and *Chapter 12: Create Hierarchical Netlist* for information on creating netlists in these formats.

To use these netlist formats and get predictable results, you must follow certain rules when you create schematics. *Chapter 3: Guidelines for creating designs* explains these rules clearly.

This appendix contains descriptions and examples of the flat and hierarchical netlist format files supplied with **Schematic Design Tools 386+**. Each section includes the characteristics of the netlist format and explains its configuration options, if any.

Many of these formats impose restrictions on the length of various names. Some netlist format files truncate long node names, for example, while others construct valid names from the corresponding node numbers.

Some formats also restrict the set of legal characters that you use. Where appropriate, IFORM and HFORM check the names for validity. When illegal names are encountered, IFORM and HFORM attempt to recover without losing information. Where this is possible, a warning is issued; where it is not, an error is issued.

Types of netlist format files

Two versions of each netlist format file are included: a compiled version and an uncompiled version written in the C programming language. You can view and edit the uncompiled netlist format files with **Edit File.**

The uncompiled versions are useful as a basis for customized netlist format files. Once you compile a custom netlist format file, you can use it as described in *Chapter 11: Create Netlist* and *Chapter 12: Create Hierarchical netlist.*

△ *NOTE: The compiled netlist format files are normally installed in \ORCADESP\SDT\NETFORMS. The two subdirectories NETSRCF and NETSRCH contain the uncompiled, or* source, *versions.*

Each uncompiled netlist format file includes a comment section near the top that lists the restrictions imposed by that particular netlist format. The same information is included in this appendix.

Table B-1 lists the filenames used for netlist format files.

Filename	*Type of netlist format file*
F*formatName*.EXE	Compiled flat
H*formatName*.EXE	Compiled hierarchical
formatName.C	Uncompiled flat
formatName.C	Uncompiled hierarchical

Table B-1. Netlist format file types and names.

On-line documentation

The file NETFORMS.DOC contains a complete list of netlist formats provided by OrCAD. You can use **View Reference** to read NETFORMS.DOC, which is normally installed in the \ORCADESP\SDT directory.

Configuring for netlists

The examples in each section assume the **Create Netlist** key fields in the **Key Fields** area of **Configure Schematic Design Tools** are configured this way:

```
Create Netlist
    Part Value Combine      V
    Module Value Combine    8
```

In each of the example netlists, the **Part Value Combine** key field (part value, represented by "V" in the example above) is shown in **bold** and the **Module Value Combine** key field (the 8th part field, represented by "8" in the example above) is shown in ***bold italics.***

For more information about configuring key fields, see the section *Key Fields* in *Chapter 1: Configure Schematic Tools.*

Flat netlists

To create a netlist in one of these flat formats, it is assumed that your schematic has already been simplified (if necessary), converted to an incremental netlist file, and linked. These steps are fully explained in *Chapter 11: Create Netlist.*

Flat netlist format files are distinguished by a filename beginning with "F." They cannot be used by **Create Hierarchical Netlist.**

Example schematics

Five sample schematics are used to create the flat netlists in this section. The first schematic, figure B-1, is a general example suitable for digital-oriented designs. The schematic in figure B-2 is used by the two ADF netlists. Figure B-3 shows how to create a model for a part for OrCAD's **Digital Simulation Tools** using a schematic as the source. Figure B-4 is an analog design that demonstrates the SPICE netlist. Figure B-5 shows how to define the programming for a PLD for OrCAD's **Programmable Logic Design Tools** using a schematic as the source. The schematic in figure B-6 is used for the OrCAD Hardware Description Language (OHDL) netlist.

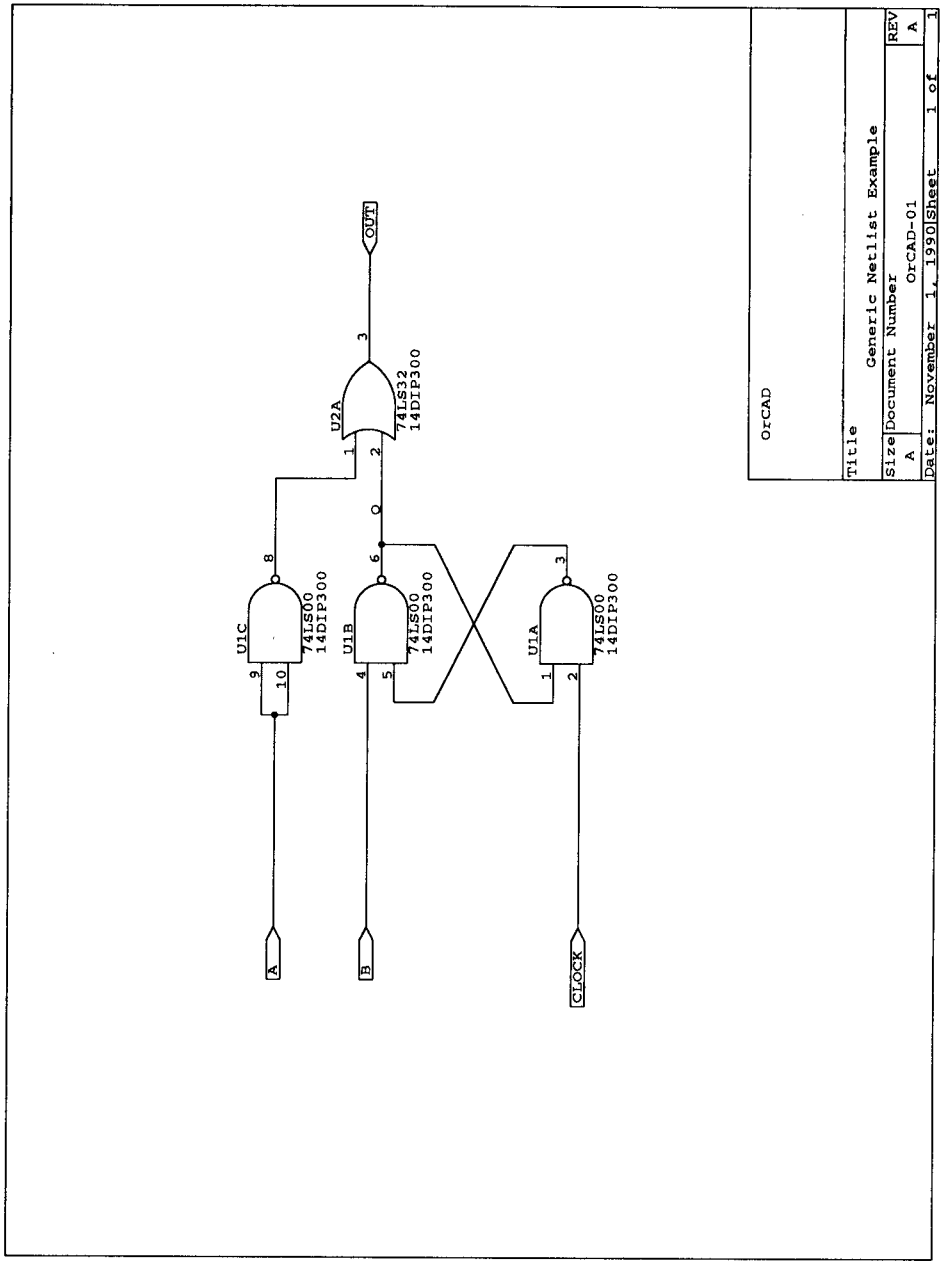


Figure B-1. Used by most of the flat netlist formats.

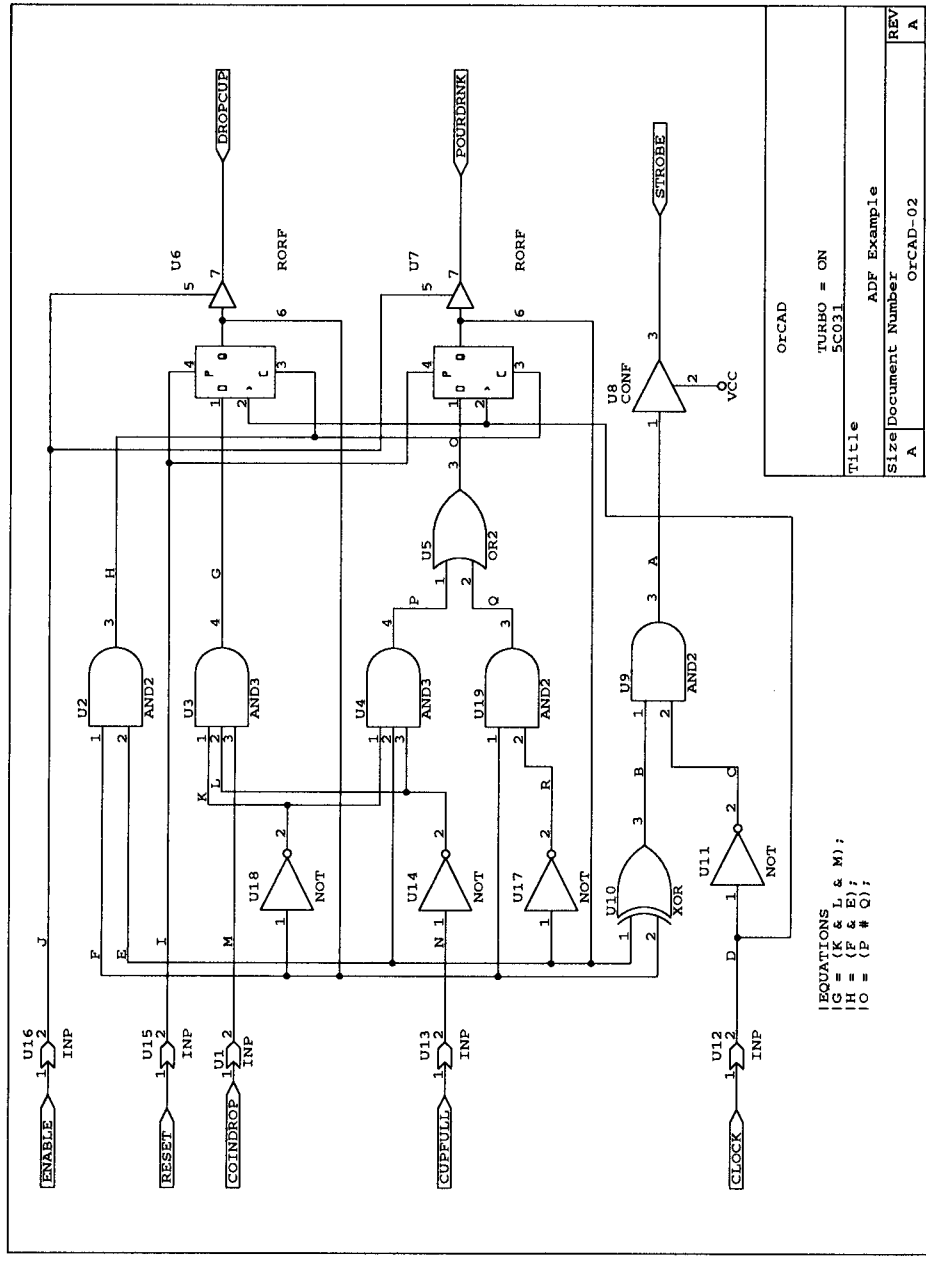


Figure B-2. Used by the AlteraADF and IntelADF formats.

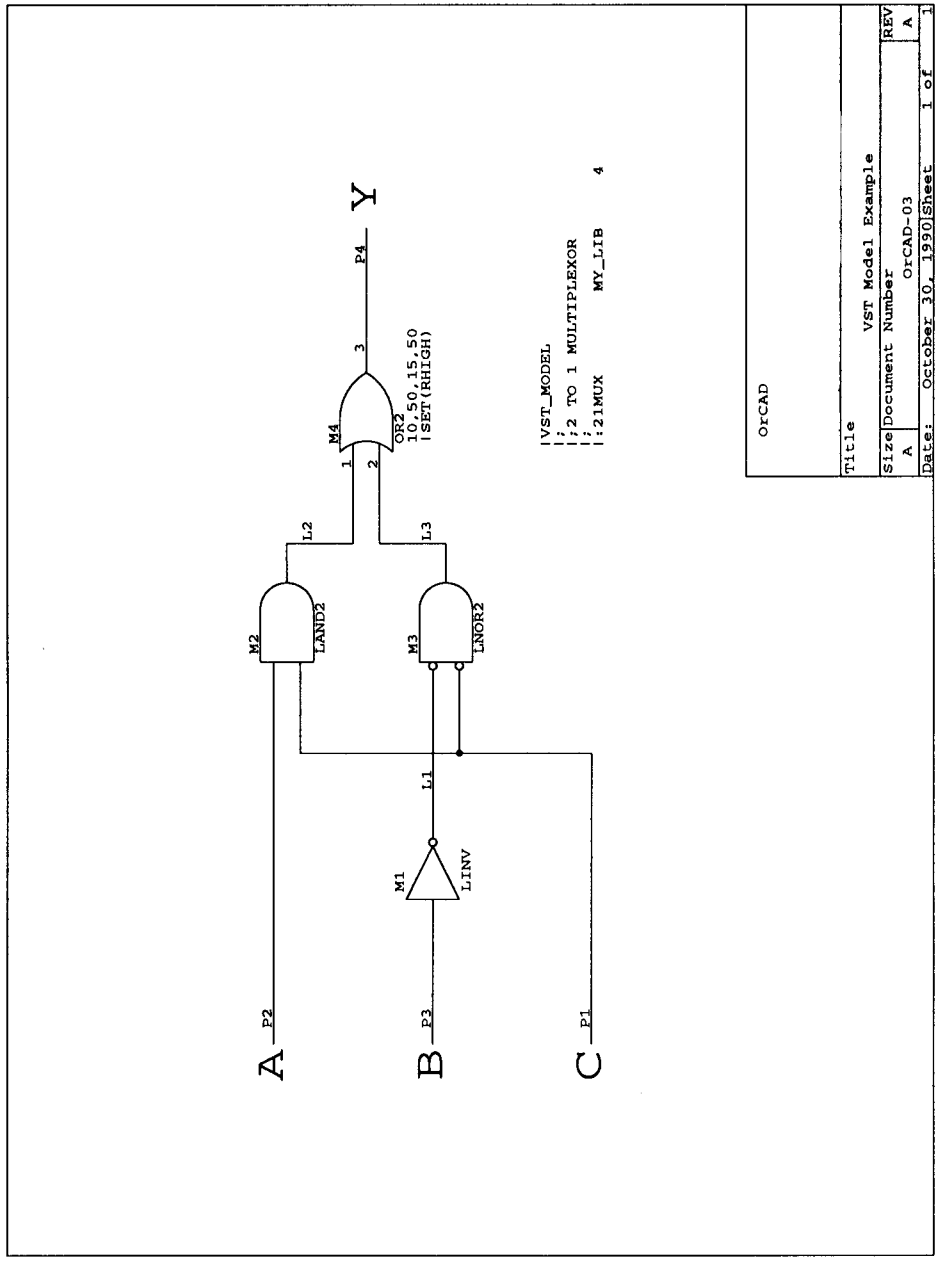


Figure B-3. Used by the OrCAD/VST Model format.

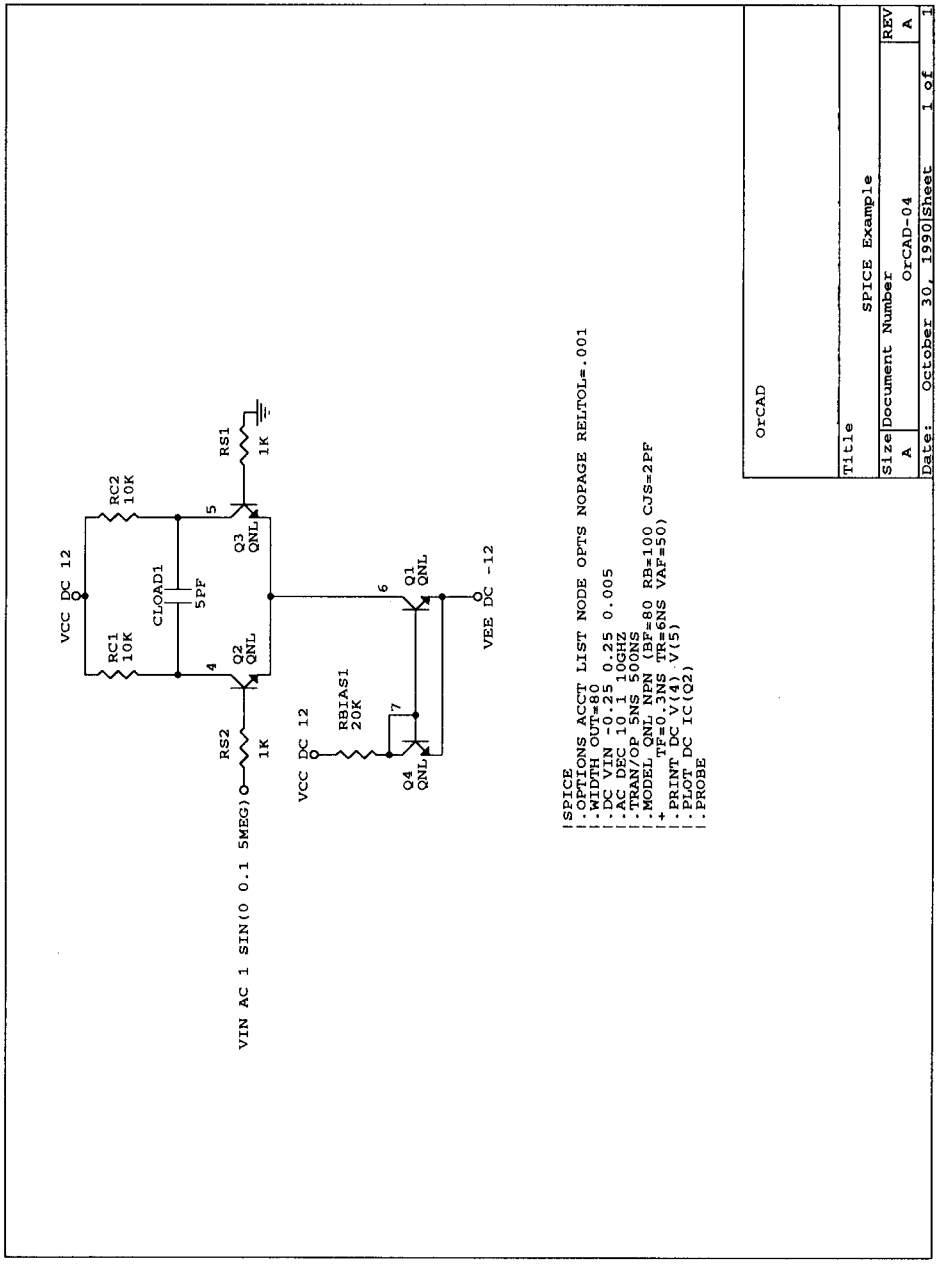


Figure B-4. Used by the flat SPICE format.

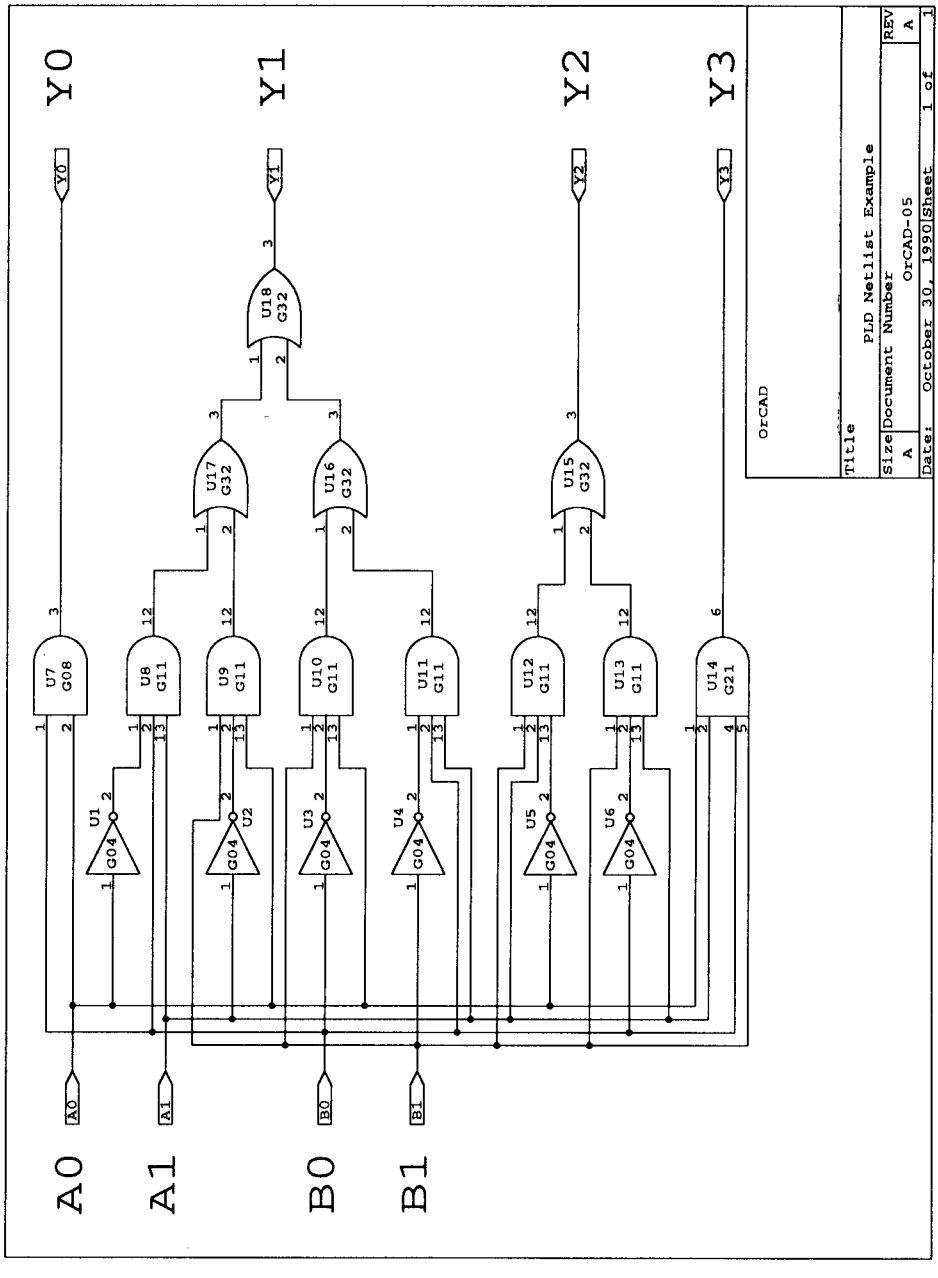


Figure B-5. Used by the OrCAD/PLD netlist format.

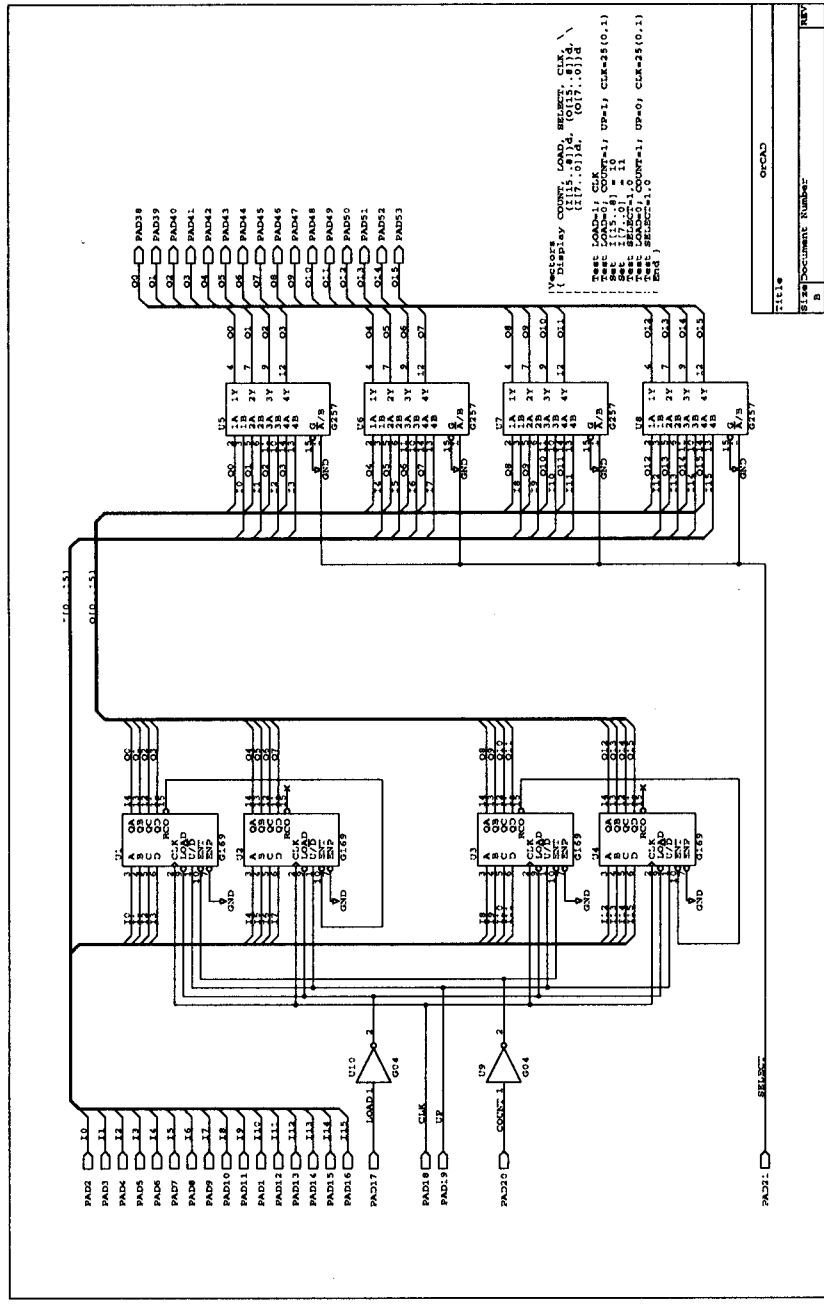


Figure B-6. Used by the OrCAD Hardware Description Language netlist format.

Algorex (FALGOREX.EXE)

The FALGOREX.EXE format file is used to produce netlists in the Algorex format.

The Algorex format has these characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to all labels.

Example

The netlist in figure B-7 was created—with no options selected—from the schematic in figure B-1.

```
N00001
          U1 (14DIP300)-8,
          U2 (14DIP300)-1
Q-1
          U1 (14DIP300)-6,
          U2 (14DIP300)-2,
          U1 (14DIP300)-1
N00003
          U1 (14DIP300)-5,
          U1 (14DIP300)-3
VCC
          U2 (14DIP300)-
14,
          U1 (14DIP300)-14
A
          U1 (14DIP300)-9,
          U1 (14DIP300)-10
GND
          U2 (14DIP300)-7,
          U1 (14DIP300)-7
B
          U1 (14DIP300)-4
OUT
          U2 (14DIP300)-3
CLOCK
          U1 (14DIP300)-2
```

Figure B-7. Example netlist in the Algorex format.

Allegro (FALLEGRO.EXE)

The FALLEGRO.EXE format file is used to produce netlists in the Allegro format.

The Allegro format has these characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a slash (/) and the sheet number to all labels.

Example

The netlist in figure B-8 was created—with no options selected—from the schematic in figure B-1.

```
$PACKAGES
14DIP300! 74LS00; U1
14DIP300! 74LS32; U2
$NETS
N00001;  U1.8 U2.1
Q/1;  U1.6 U2.2 U1.1
N00003;  U1.5 U1.3
VCC;  U2.14 U1.14
A;  U1.9 U1.10
GND;  U2.7 U1.7
B;  U1.4
OUT;  U2.3
CLOCK;  U1.2
$END
```

Figure B-8. Example netlist in the Allegro format.

AlteraADF (FALTERAA.EXE)

The FALTERAA.EXE format file is used to produce netlists in the AlteraADF format.

The AlteraADF format has these characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- All ASCII characters are legal.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a period (.) and the sheet number to any labels.

- ☐ `Suppress comments`

  Tells IFORM not to put comments in the netlist file. Comments in the AlteraADF format are delimited by the percent (%) character.

- ☐ `Include unconnected pins`

  Tells IFORM to assign all unconnected pins a unique net. If you do not select this option, IFORM does not assign a net and reports a warning.

Title block information

Title block information is placed in the first 10 lines of the netlist. Table B-2 shows an example netlist header and the title block information from which the header was extracted. Header information in **bold** is text entered in the schematic's title block.

Line	*Example Header*	*Title Block Field*
1	**ADF Example**	Title of sheet
1	June 30, 1994	Date
2	**OrCAD-02**	Document Number
2	**A**	Revision Code
3	**OrCAD**	Organization Name
4	**9300 SW Nimbus Avenue**	1st Address Line
6	**Turbo = ON**	3rd Address Line
7	**5C031**	4th Address Line

Table B-2. Title block information in AlteraADF netlists.

Pipe commands

You can place equations in your schematic to be included in the netlist. To place these equations in your worksheet, use **Draft's PLACE Text** command. You can also use a text editor to put the equations in an ASCII file, then use **Draft's BLOCK ASCII Import** command to import the contents of the file and place it on the worksheet.

Each equation must start with the *pipe* character (|). The first line must be:

```
|EQUATIONS
```

This tells IFORM that some AlteraADF equations need to be included in the netlist. The equations can contain any information you want to include in the netlist.

Constraints

When you create an AlteraADF netlist, you must configure **Schematic Design Tools 386+** to use the OrCAD-supplied ALTERA_P.LIB and ALTERA_M.LIB libraries. You can only use the parts provided in the Altera libraries to create the schematic.

Inputs and outputs are handled differently in **Schematic Design Tools 386+** and the Altera software. **Schematic Design Tools 386+** defines inputs and outputs with module ports and an input/output library object. Altera defines inputs and outputs with a library object which is then tagged with the appropriate pin number. In figure B-2, the CLOCK signal is an input and the STROBE signal is an output.

Additionally, library objects with unused pins default to predefined levels in the Altera software. Because **Schematic Design Tools 386+** does not default unconnected pins to any particular level, you must tie all unused pins to the appropriate level.

Example The netlist in figure B-9 was created—with no options selected—from the schematic in figure B-2.

```
ADF Example                              Revised:   June 30, 1994
OrCAD-02                                 Revision: A
OrCAD
9300 SW Nimbus Avenue

TURBO = ON
5C031

OPTIONS:TURBO = ON
PART:5C031

INPUTS:
     ENABLE
     RESET
     COINDROP
     CUPFULL
     CLOCK

OUTPUTS:
     DROPCUP
     POURDRNK
     STROBE

NETWORK:
H.1=AND(F.1,E.1) % SYM 1 %
A.1=AND(B.1,C.1) % SYM 2 %
Q.1=AND(F.1,R.1) % SYM 3 %
G.1=AND(K.1,L.1,M.1) % SYM 4 %
P.1=AND(K.1,E.1,L.1) % SYM 5 %
STROBE=CONF(A.1,VCC) % SYM 6 %
M.1=INP(COINDROP) % SYM 7 %
D.1=INP(CLOCK) % SYM 8 %
N.1=INP(CUPFULL) % SYM 9 %
I.1=INP(RESET) % SYM 10 %
J.1=INP(ENABLE) % SYM 11 %
C.1=NOT(D.1) % SYM 12 %
L.1=NOT(N.1) % SYM 13 %
R.1=NOT(E.1) % SYM 14 %
K.1=NOT(F.1) % SYM 15 %
O.1=OR(P.1,Q.1) % SYM 16 %
DROPCUP,F.1=RORF(G.1,D.1,H.1,I.1,J.1) % SYM 17 %
POURDRNK,E.1=RORF(O.1,D.1,H.1,I.1,J.1) % SYM 18 %
B.1=XOR(E.1,F.1) % SYM 19 %

EQUATIONS:
G = (K & L & M);
H = (F & E);
O = (P # Q);
END$
```

Figure B-9. Example netlist in the AlteraADF format.

AppliconBRAVO (FAPPLBRA.EXE)

The FAPPLBRA.EXE format file is used to produce netlists in the AppliconBRAVO format.

AppliconBRAVO netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to all labels.

Example

The netlist in figure B-10 was created—with no options selected—from the schematic in figure B-1.

```
*** Desig 14DIP300
U1
*** Desig 14DIP300
U2
*** NET N00001
U1 8
U2 1
*** NET Q-1
U1 6
U2 2
U1 1
*** NET N00003
U1 5
U1 3
*** NET VCC
U2 14
U1 14
*** NET A
U1 9
U1 10
*** NET GND
U2 7
U1 7
*** NET B
U1 4
*** NET OUT
U2 3
*** NET CLOCK
U1 2
```

Figure B-10. Example netlist in the AppliconBRAVO format.

AppliconLEAP (FAPPLLEA.EXE)

The FAPPLLEA.EXE format file is used to produce netlists in the AppliconLEAP format.

AppliconLEAP netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to all labels.

Example

The netlist in figure B-11 was created—with no options selected—from the schematic in figure B-1.

```
*** NET N00001
U1 8 14DIP300
U2 1 14DIP300
*** NET Q-1
U1 6 14DIP300
U2 2 14DIP300
U1 1 14DIP300
*** NET N00003
U1 5 14DIP300
U1 3 14DIP300
*** NET VCC
U2 14 14DIP300
U1 14 14DIP300
*** NET A
U1 9 14DIP300
U1 10 14DIP300
*** NET GND
U2 7 14DIP300
U1 7 14DIP300
*** NET B
U1 4 14DIP300
*** NET OUT
U2 3 14DIP300
*** NET CLOCK
U1 2 14DIP300
```

Figure B-11. Example netlist in the AppliconLEAP format.

Cadnetix (FCADNETI.EXE)

The FCADNETI.EXE format file is used to produce netlists in the Cadnetix format.

Cadnetix netlists have the following characteristics:

- Part names can contain up to 17 characters.
- Module names can contain up to 15 characters.
- Reference strings plus pin numbers can contain up to 12 characters.
- Node names can contain up to 16 characters.
- Pin numbers can contain up to 3 digits.
- Pin names are not used.
- Node numbers are not checked for length.
- All ASCII characters are legal.

Format Specific Options

- ❑ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to all labels.

Example The netlist in figure B-12 was created—with no options selected—from the schematic in figure B-1.

```
PARTS LIST

74LS00              14DIP300          U1
74LS32              14DIP300          U2
EOS
NET LIST

NODE      1                    $
    U1            8  U2            1
NODENAME Q_1                   $
    U1            6  U2            2  U1          1
NODE      3                    $
    U1            5  U1            3
NODENAME VCC                   $
    U2           14  U1           14
NODENAME A                     $
    U1            9  U1           10
NODENAME GND                   $
    U2            7  U1            7
NODENAME B                     $
    U1            4
NODENAME OUT                   $
    U2            3
NODENAME CLOCK                 $
    U1            2
EOS
```

Figure B-12. Example netlist in the Cadnetix format.

Calay (FCALAY.EXE)

The FCALAY.EXE format file is used to produce netlists in the Calay format. This is the older of two Calay netlist formats. The next section describes the newer format.

Calay netlists have the following characteristics:

- Part names, module names, and reference strings can each contain up to 19 characters.
- Node names can contain up to eight characters. Legal characters for node names are:

  ```
  +   -   0..9   A..Z   a..z
  ```

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- Pin numbers are not checked for length.
- All ASCII are legal except as noted for node names.

File Options

In addition to the netlist file, IFORM also creates a component file when you select the Calay format. You must enter a the component filename in the **Destination 2** entry box on the **Configure Netlist Format** screen.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to all labels.

Example The netlist in figure B-13 and its corresponding component file in figure B-14 were created—with no options selected—from the schematic in figure B-1.

```
/N00001    U1(8) U2(1);
/Q-1       U1(6) U2(2) U1(1);
/N00003    U1(5) U1(3);
/VCC       U2(14) U1(14);
/A         U1(9) U1(10);
/GND       U2(7) U1(7);
/B         U1(4);
/OUT       U2(3);
/CLOCK     U1(2);
```

Figure B-13. Example netlist in the Calay format.

```
74LS00          U1          14DIP300          000     000     0
74LS32          U2          14DIP300          000     000     0
```

Figure B-14. Example component file in the Calay format.

Calay (FCALAY90.EXE)

The FCALAY.EXE format file is used to produce netlists in a new Calay format.

Calay netlists in the new format have the same characteristics and options as the original Calay format discussed in the previous section of this appendix.

Example

The netlist in figure B-15 and its corresponding component file in figure B-16 were created—with no options selected—from the schematic in figure B-1.

```
N00001    U1('8) U2('1);
Q-1       U1('6) U1('1) U2('2);
N00003    U1('5) U1('3);
VCC       U2('14) U1('14);
A         U1('9) U1('10);
GND       U2('7) U1('7);
B         U1('4);
OUT       U2('3);
CLOCK     U1('2);
```

Figure B-15. Example netlist in the new Calay format.

```
74LS00          U1          14DIP300          000     000     0
74LS32          U2          14DIP300          000     000     0
```

Figure B-16. Example component file in the new Calay format.

Case (FCASE.EXE)

The FCASE.EXE format file is used to produce netlists in the Case format.

Case netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "X" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

- ❑ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to all labels.

- ❑ `Include unconnected pins`

  If you select this option, IFORM assigns all unconnected pins a unique net. If you do not select this option, IFORM does not assign a net and reports a warning.

Example

The netlist in figure B-17 was created—with no options selected—from the schematic in figure B-1.

```
ASSERTIONS=OFF;VERSION=400;LOCATION=LOC;
 [SIZE=1;TIMES=1;LOC=(U1);PLOC=U1;SHAPE=14DIP300]
 1=Q_1;
 2=CLOCK;
 3=X00003;
 4=B;
 5=X00003;
 6=Q_1;
 7=GND;
 8=X00001;
 9=A;
 10=A;
 11=NC;
 12=NC;
 13=NC;
 14=VCC;
 ;
 [SIZE=1;TIMES=1;LOC=(U2);PLOC=U2;SHAPE=14DIP300]
 1=X00001;
 2=Q_1;
 3=OUT;
 4=NC;
 5=NC;
 6=NC;
 7=GND;
 8=NC;
 9=NC;
 10=NC;
 11=NC;
 12=NC;
 13=NC;
 14=VCC;
 ;
;
```

Figure B-17. Example netlist in the Case format.

CBDS (FCBDS.EXE)

The FCBDS.EXE format file is used to produce netlists in the CBDS format.

CBDS netlists have the following characteristics:

- Part names, module names, reference strings, and pin numbers are not checked for length.
- Node names can contain up to 20 characters. These characters are legal:

  `/ - 0..9 a..z A..Z`

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except as noted for node names.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to all labels.

Example

The netlist in figure B-18 was created—with no options selected—from the schematic in figure B-1.

```
.SEARCH P,C
.DD  U1 14DIP300
.DD  U2 14DIP300
.S,N00001,U1,8,U2,1
.S,Q-1,U1,6,U2,2,U1,1
.S,N00003,U1,5,U1,3
.S,VCC,U2,14,U1,14
.S,A,U1,9,U1,10
.S,GND,U2,7,U1,7
.S,B,U1,4
.S,OUT,U2,3
.S,CLOCK,U1,2
```

Figure B-18. Example netlist in the CBDS format.

ComputerVision (FCOMPVIS.EXE)

The FCOMPVIS.EXE format file is used to produce netlists in the ComputerVision format.

ComputerVision netlists have the following characteristics:

- Part names, module names, reference strings, and pin numbers are not checked for length.
- Node names can contain up to 19 characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a slash (/) and the sheet number to all labels.

Example

The netlist in figure B-19 was created—with no options selected—from the schematic in figure B-1.

```
0001 N00001                U1-8  U2-1
0002 Q/1                   U1-6  U2-2  U1-1
0003 N00003                U1-5  U1-3
0004 VCC                   U2-14  U1-14
0005 A                     U1-9  U1-10
0006 GND                   U2-7  U1-7
0007 B                     U1-4
0008 OUT                   U2-3
0009 CLOCK                 U1-2
```

Figure B-19. Example netlist in the ComputerVision format.

DUMP (FDUMP.EXE) The FDUMP.EXE format file is used to produce a flat netlist containing all the information on the schematic sheets. No information is omitted or changed. You can use this netlist format when troubleshooting a design.

EDIF (FEDIF.EXE)

Two EDIF (Version 2 0 0) formats are supplied with **Schematic Design Tools 386+**: FEDIF.EXE, used to create flat EDIF netlists (described here), and HEDIF.EXE, used to create hierarchical EDIF netlists (described in the *Hierarchical netlists* section of this appendix). EDIF netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Legal characters are:

  `0..9  a..z  A..Z  _` (underscore)

  Case is not significant. Unless you select the **Allow nonEDIF characters** option, IFORM issues a warning and makes the following changes when it encounters an illegal character:

  - Changes "-" to "MINUS"
  - Changes "+" to "PLUS"
  - Changes "\" and "/" to "BAR"
  - Changes all other illegal characters to "_"

For more information about the EDIF format, contact EDIF 2 0 0 Specification, E.I.A. , Engineering Department, 2001 Eye Street N.W., Washington D.C. 20006.

Format Specific Options

Some of the options specific to the EDIF netlist format are included to support **PC Board Layout Tools 386+**. If you are creating a netlist for use with **PCB 386+**, select these options:

- ❑ `Allow nonEDIF characters`

If you are creating a netlist for use with any other EDIF reader, you may want to select these options:

- ❑ `Convert PartFields to Named Properties`
- ❑ `Do not output additional properties`
- ❑ `Output pin names (instead of pin numbers)`

This section describes all the EDIF format-specific options.

- `Keep system generated netnames`

  If you select this option, IFORM creates a persistent netname (.PNN) file. If the .PNN file already exists, IFORM compares any unnamed nets in the two versions. For each unnamed net in the new file, if more than half of the nodes in that net are in a single unnamed net in the old file, IFORM assigns that new net the same name as its counterpart in the old file. *Chapter 10: Creating a netlist* contains more information about persistent netnames and the effect of this option.

  Without this option, IFORM does not create a .PNN file or check for an existing .PNN file. Instead, IFORM generates names for any unnamed nets as they are found.

- `Allow nonEDIF characters`

  Select this option if you are creating a netlist for **PC Board Layout Tools 386+** or another EDIF reader that allows non-EDIF characters. This option prevents FEDIF.EXE from checking for legal EDIF characters.

- `Convert PartFields to Named Properties`

  Select this option if you are creating a netlist for some EDIF reader other than **PC Board Layout Tools 386+**. If you select this option, IFORM replaces the field name (such as PartField1) with the user-defined property name.

  Part field strings used this way must have the form *propertyName=value*.

- `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore(_) and the sheet number to all labels.

- ❑ `Do not output additional properties`

  Select this option if you are creating a netlist for use with an EDIF reader that doesn't require properties. When you select this option, IFORM outputs only the module value, so it takes less time and creates a smaller netlist file. Without this option, IFORM includes time stamps and part field strings in the EDIF netlist.

  Do not select this option to create a netlist for **PC Board Layout Tools 386+.**

- ❑ `Output pin names (instead of pin numbers)`

  Select this option if you want the netlist to contain pin names instead of pin numbers. Most EDIF readers expect pin names instead of pin numbers. Do not use this option to create a netlist for **PC Board Layout Tools 386+.**

- ❑ `Do not create 'external' library declaration`

  Use this option to prevent IFORM from creating an EDIF `external` statement in the netlist file. **Schematic Design Tools 386+** uses `external` statements to identify OrCAD as the source of the library parts in the netlist, but some EDIF readers do not accept external statements.

Example The netlist in figure B-20 was created—with the **Do not output additional properties** option selected—from the schematic in figure B-1.

```
(edif &EX1
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap (keywordLevel 0))
 (status
  (written
   (timeStamp 0 0 0 0 0 0)
   (program "IFORM.EXE")
   (comment "Original data from OrCAD/SDT schematic"))
  (comment "Generic Netlist Example")
  (comment " November  7, 1990")
  (comment "OrCAD-01")
  (comment "A")
  (comment "OrCAD")
  (comment "9300 SW Nimbus Avenue")
  (comment "Beaverton, OR  97008")
  (comment "(503) 671-9500 Sales & Administration")
  (comment "(503) 671-9400 Technical Support"))
 (external OrCAD_LIB
  (edifLevel 0)
  (technology
   (numberDefinition
    (scale 1 1 (unit distance))))
  (cell &74LS00
   (cellType generic)
   (comment "From OrCAD library TTL.LIB")
   (view NetlistView
    (viewType netlist)
    (interface
     (port &I0_A (direction INPUT))
     (port &I0_B (direction INPUT))
     (port &I0_C (direction INPUT))
     (port &I0_D (direction INPUT))
     (port &I1_A (direction INPUT))
     (port &I1_B (direction INPUT))
     (port &I1_C (direction INPUT))
     (port &I1_D (direction INPUT))
     (port &O_A (direction OUTPUT))
     (port &O_B (direction OUTPUT))
     (port &O_C (direction OUTPUT))
     (port &O_D (direction OUTPUT))
     (port &VCC (direction INPUT))
     (port &GND (direction INPUT)))))))
```

Figure B-20. Example netlist in the EDIF format (continues).

```
  (cell &74LS32
   (cellType generic)
   (comment "From OrCAD library TTL.LIB")
   (view NetlistView
    (viewType netlist)
    (interface
     (port &I0_A (direction INPUT))
     (port &I0_B (direction INPUT))
     (port &I0_C (direction INPUT))
     (port &I0_D (direction INPUT))
     (port &I1_A (direction INPUT))
     (port &I1_B (direction INPUT))
     (port &I1_C (direction INPUT))
     (port &I1_D (direction INPUT))
     (port &O_A (direction OUTPUT))
     (port &O_B (direction OUTPUT))
     (port &O_C (direction OUTPUT))
     (port &O_D (direction OUTPUT))
     (port &VCC (direction INPUT))
     (port &GND (direction INPUT))))))
(library MAIN_LIB
 (edifLevel 0)
 (technology
  (numberDefinition
   (scale 1 1 (unit distance))))
 (cell &EX1
  (cellType generic)
  (view NetlistView
   (viewType netlist)
   (interface
    (port &A (direction INPUT))
    (port &B (direction INPUT))
    (port &OUT (direction OUTPUT))
    (port &CLOCK (direction INPUT)))
   (contents
    (instance &U1
     (viewRef NetlistView
      (cellRef &74LS00
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS00"))
     (property ModuleValue (string "14DIP300")))
    (instance &U2
     (viewRef NetlistView
      (cellRef &74LS32
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS32"))
     (property ModuleValue (string "14DIP300")))
    (net N00001
     (joined
      (portRef &O_C (instanceRef &U1))
      (portRef &I0_A (instanceRef &U2))))
```

Figure B-20. Example netlist in the EDIF format (continues).

```
      (net &Q_1
       (joined
        (portRef &O_B (instanceRef &U1))
        (portRef &I0_A (instanceRef &U1))
        (portRef &I1_A (instanceRef &U2))))
      (net N00003
       (joined
        (portRef &I1_B (instanceRef &U1))
        (portRef &O_A (instanceRef &U1))))
      (net &VCC
       (joined
        (portRef &VCC (instanceRef &U2))
        (portRef &VCC (instanceRef &U1))))
      (net &A
       (joined
        (portRef &A)
        (portRef &I0_C (instanceRef &U1))
        (portRef &I1_C (instanceRef &U1))))
      (net &GND
       (joined
        (portRef &GND (instanceRef &U2))
        (portRef &GND (instanceRef &U1))))
      (net &B
       (joined
        (portRef &B)
        (portRef &I0_B (instanceRef &U1))))
      (net &OUT
       (joined
        (portRef &OUT)
        (portRef &O_A (instanceRef &U2))))
      (net &CLOCK
       (joined
        (portRef &CLOCK)
        (portRef &I1_A (instanceRef &U1))))))))
  (design &EX1
   (cellRef &EX1
    (libraryRef MAIN_LIB))))
```

Figure B-20. Example netlist in the EDIF format (continued).

EEDesigner (FEEDESIG.EXE)

The FEEDESIG.EXE format file is used to produce netlists in the EEDesigner format.

EEDesigner netlists have the following characteristics:

- Part names, module names, and pin numbers are not checked for length.
- Reference strings are limited to eight characters.
- Node names are *not* supported.
- Node numbers are limited to three digits following the "UN" prefix.
- Pin names are not used.
- All ASCII characters are legal.

The EEDesigner format has no format-specific options.

Example

The netlist in figure B-21 was created from the schematic in figure B-1.

```
(
PATH,OrCAD()
(COMPONENTS
U1   ,14DIP300
U2   ,14DIP300
)
(NODES
(UN001
U1   , 8
U2   , 1
)
(UN002
U1   , 6
U2   , 2
U1   , 1
)
(UN003
U1   , 5
U1   , 3
)
(UN004
U2   , 14
U1   , 14
)
(UN005
U1   , 9
U1   , 10
)
(UN006
U2   , 7
U1   , 7
)
(UN007
U1   , 4
)
(UN008
U2   , 3
)
(UN009
U1   , 2
)
)
),OrCAD
```

Figure B-21. Example netlist in the EEDesigner format.

FutureNet (FFUTURE.EXE)

The FFUTURE.EXE format file is used to produce netlists in the FutureNet format.

FutureNet netlists have the following characteristics:

- Part names are limited to 16 characters.
- Module names, node names, and pin numbers are not checked for length.
- Reference strings are limited to six characters.
- Node numbers are limited to eight digits following the "****" prefix.
- Characters are not checked for legality.

Format Specific Options

❑ `Create a netlist (instead of a pinlist)`

The FutureNet system has two connectivity output formats: a netlist, and a pinlist. The netlist format lists each net in the schematic and the part pins that belong in that net. The pinlist format is a list of each pin on a part, and the net in which that pin belongs. The FutureNet format can create both netlists and pinlists.

If you select the **Create a netlist (instead of a pinlist)** option, a netlist is created instead of a pinlist. The two formats contain the same information, hence, neither has any inherent advantages over the other. You need to decide which format is best suited to your needs.

❑ `Output pin numbers (instead of pin names)`

Not all parts have both pin numbers and pin names defined. If you select this option and a pin without a number is found, then the pin name is used. If you need a netlist consisting entirely of pin numbers, you may need to modify the OrCAD-supplied libraries, which is explained in the SPICE netlist section.

- ❑ `Assign SIG* attributes to module ports`

  Normally, all module ports (input, output, and bidirectional) are assigned the attribute of 5 (signal name). If you select this option, IFORM substitutes the more precise attributes 10, 11, and 12 (input signal, output signal, and bidirectional signal, respectively) for the generic signal attribute 5.

- ❑ `Create CON* symbols for module ports`

  Tells IFORM to create FutureNet SYMbol objects for the module ports. The new SYMbol objects (input, output, and bidirectional) are assigned the attributes 24, 25, and 26, respectively; and have the names CONI, CONO, and CONB, respectively, assigned in their data fields.

- ❑ `Assign FutureNet power attributes to power objects`

  If you select this option, IFORM assigns FutureNet power attributes to **Schematic Design Tools 386+** power objects. The pins on the following **Schematic Design Tools 386+** power objects (which are matched by name) are assigned FutureNet power attributes.

OrCAD Pin Value	*FutureNet Attribute*
GND	100
+5V	101
+12V	105
-12V	106
VEE	107

Table B-3. FutureNet power attribute equivalents.

- ❑ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore(_) and the sheet number to all labels.

Examples

The example pinlist in figure B-22 was created—with no options selected—from the schematic in figure B-1. The netlist in figure B-23 was created from the schematic in figure B-1 with the **Create a netlist (instead of a pinlist)** option selected.

```
PINLIST,2
(DRAWING,ORCAD.PIN,1-1
(SYM,1
DATA,2,U1
DATA,3,74LS00
DATA,4,14DIP300
PIN,,Q_1,1-1,5,23,I0_A
PIN,,CLOCK,1-1,5,23,I1_A
PIN,,***000003,1-1,5,21,O_A
PIN,,B,1-1,5,23,I0_B
PIN,,***000003,1-1,5,23,I1_B
PIN,,Q_1,1-1,5,21,O_B
PIN,,GND,1-1,5,23,GND
PIN,,***000001,1-1,5,21,O_C
PIN,,A,1-1,5,23,I0_C
PIN,,A,1-1,5,23,I1_C
PIN,,UN000001,1-1,5,21,O_D
PIN,,UN000002,1-1,5,23,I0_D
PIN,,UN000003,1-1,5,23,I1_D
PIN,,VCC,1-1,5,23,VCC
)
(SYM,2
DATA,2,U2
DATA,3,74LS32
DATA,4,14DIP300
PIN,,***000001,1-1,5,23,I0_A
PIN,,Q_1,1-1,5,23,I1_A
PIN,,OUT,1-1,5,21,O_A
PIN,,UN000004,1-1,5,23,I0_B
PIN,,UN000005,1-1,5,23,I1_B
PIN,,UN000006,1-1,5,21,O_B
PIN,,GND,1-1,5,23,GND
PIN,,UN000007,1-1,5,21,O_C
PIN,,UN000008,1-1,5,23,I0_C
PIN,,UN000009,1-1,5,23,I1_C
PIN,,UN000010,1-1,5,21,O_D
PIN,,UN000011,1-1,5,23,I0_D
PIN,,UN000012,1-1,5,23,I1_D
PIN,,VCC,1-1,5,23,VCC
)
```

Figure B-22. Example pinlist in the FutureNet netlist format (continues).

```
SIG,Q_1,1-1,5,Q_1
SIG,VCC,1-1,5,VCC
SIG,A,1-1,5,A
SIG,GND,1-1,5,GND
SIG,B,1-1,5,B
SIG,OUT,1-1,5,OUT
SIG,CLOCK,1-1,5,CLOCK
)
```

Figure B-22. Example pinlist in the FutureNet netlist format (continued).

```
NETLIST,2
(DRAWING,ORCAD.NET,1-1
DATA,50,Generic Netlist Example
DATA,51,OrCAD-01
DATA,52,A
DATA,54,  June 30, 1994
)
(SYM,1-1,1
DATA,2,U1
DATA,3,74LS00
DATA,4,14DIP300
DATA,23,I0_A
DATA,23,I1_A
DATA,21,O_A
DATA,23,I0_B
DATA,23,I1_B
DATA,21,O_B
DATA,23,GND
DATA,21,O_C
DATA,23,I0_C
DATA,23,I1_C
DATA,21,O_D
DATA,23,I0_D
DATA,23,I1_D
DATA,23,VCC
)
(SYM,1-1,2
DATA,2,U2
DATA,3,74LS32
DATA,4,14DIP300
DATA,23,I0_A
DATA,23,I1_A
DATA,21,O_A
```

Figure B-23. Example netlist in the FutureNet netlist format (continues).

```
DATA,23,I0_B
DATA,23,I1_B
DATA,21,O_B
DATA,23,GND
DATA,21,O_C
DATA,23,I0_C
DATA,23,I1_C
DATA,21,O_D
DATA,23,I0_D
DATA,23,I1_D
DATA,23,VCC
)
(SIG,,***000001,1-1,5,***000001
PIN,1-1,1,U1,21,O_C
PIN,1-1,2,U2,23,I0_A
)
(SIG,,Q_1,1-1,5,Q_1
PIN,1-1,1,U1,21,O_B
PIN,1-1,2,U2,23,I1_A
|PIN,1-1,1,U1,23,I0_A
)
(SIG,,***000003,1-1,5,***000003
PIN,1-1,1,U1,23,I1_B
PIN,1-1,1,U1,21,O_A
)
(SIG,,VCC,1-1,5,VCC
PIN,1-1,2,U2,23,VCC
PIN,1-1,1,U1,23,VCC
)
(SIG,,A,1-1,5,A
PIN,1-1,1,U1,23,I0_C
PIN,1-1,1,U1,23,I1_C
)
(SIG,,GND,1-1,5,GND
PIN,1-1,2,U2,23,GND
PIN,1-1,1,U1,23,GND
)
(SIG,,B,1-1,5,B
PIN,1-1,1,U1,23,I0_B
)
(SIG,,OUT,1-1,5,OUT
PIN,1-1,2,U2,21,O_A
)
(SIG,,CLOCK,1-1,5,CLOCK
PIN,1-1,1,U1,23,I1_A
)
```

Figure B-23. Example netlist in the FutureNet netlist format (continued).

HiLo (FHILO.EXE)

The FHILO.EXE format file is used to produce netlists in the HiLo format.

HiLo netlists have the following characteristics:

- Part names, module names, reference strings, and pin numbers are not checked for length.
- Node names are limited to 14 characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dollar sign ($) and the sheet number to any labels.

- ☐ `Include unconnected pins`

  If you select this option, IFORM assigns all unconnected pins a unique net. If you do not select this option, IFORM does not assign a net and reports a warning.

Example

The netlist in figure B-24 was created—with no options selected—from the schematic in figure B-1.

```
** Generic Netlist Example        Revised:   June 30, 1994
** OrCAD-01                       Revision: A
** OrCAD
** 9300 SW Nimbus Avenue
** Beaverton, OR  97008
** (503) 671-9500 Sales & Administration
** (503) 671-9400 Technical Support
CCT ORCAD (
** Please put your circuit interface definition here
        );
14DIP300
U1 (
     Q$1,
     CLOCK,
     N00003,
     B,
     N00003,
     Q$1,
     GND,
     N00001,
     A,
     A,
     ,
     ,
     ,
     VCC
     );

14DIP300
U2 (
     N00001,
     Q$1,
     OUT,
     ,
     ,
     ,
     GND,
     ,
     ,
     ,
     ,
     ,
     ,
     VCC
     );
```

Figure B-24. Example netlist in the HiLo format.

IntelADF (FINTELAD.EXE)

The FINTELAD.EXE format file is used to produce netlists in the IntelADF format.

IntelADF netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are not used.
- All ASCII characters are legal.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a period (.) and the sheet number to any labels.

- ☐ `Suppress comments`

  If you select this option, IFORM does not put comments in the resulting netlist file. Comments in the IntelADF format are delimited by the percent (%) character.

- ☐ `Include unconnected pins`

  If you select this option, IFORM assigns all unconnected pins a unique net. If you do not select this option, IFORM does not assign a net and reports a warning.

Title block

Title block information is placed in the first 10 lines of the netlist. Table B-4 shows an example netlist header and the title block information from which the header was extracted. Header information in **bold** is text you enter in the schematic's title block.

Line	*Example Header*	*Title Block Field*
1	**ADF Example**	Title of sheet
1	June 30, 1994	Date
2	**OrCAD-02**	Document Number
2	**A**	Revision Code
3	**OrCAD**	Organization Name
4	**9300 SW Nimbus Avenue**	1st Address Line
6	**Turbo = ON**	3rd Address Line
7	**5C031**	4th Address Line

Table B-4. Title block information in IntelADF netlists.

Pipe commands

You can place equations in your schematic to be included in the netlist. To place these equations in your worksheet, use the **Draft's PLACE Text** command. You can also use a text editor to put the equations in an ASCII file, then use **Draft's BLOCK ASCII Import** command to import text.

Each equation must start with the *pipe* character (|). The first line must be:

```
|EQUATIONS
```

This tells IFORM that some IntelADF equations need to be included in the netlist. The equations can contain any information you want to include in the netlist.

Constraints

To create an IntelADF netlist, you must configure **Schematic Design Tools 386+** to use the OrCAD-supplied ADF.LIB library. You can only use the parts provided in the ADF library to create the schematic.

OrCAD's **Schematic Design Tools 386+** and the Intel software handle inputs and outputs differently. **Schematic Design Tools 386+** defines inputs and outputs with module ports and an input/output library object. Intel defines inputs and outputs with a library object which is then tagged with the appropriate pin number. In figure B-2, the CLOCK signal is an input and the STROBE signal is an output.

Also, library objects with unused pins default to pre-defined levels in the Intel software. Because **Schematic Design Tools 386+** does not default unconnected pins to any particular level, you must tie all unused pins to the appropriate level.

Example The netlist in figure B-25 was created—with no options selected—from the schematic in figure B-2.

```
ADF Example                          Revised:   June 30, 1994
OrCAD-02                             Revision: A
OrCAD
9300 SW Nimbus Avenue

TURBO = ON
5C031

OPTIONS:TURBO = ON
PART:5C031

INPUTS:
    ENABLE
    RESET
    COINDROP
    CUPFULL
    CLOCK

OUTPUTS:
    DROPCUP
    POURDRNK
    STROBE

NETWORK:
H.1=AND(F.1,E.1) % SYM 1 %
A.1=AND(B.1,C.1) % SYM 2 %
Q.1=AND(F.1,R.1) % SYM 3 %
G.1=AND(K.1,L.1,M.1) % SYM 4 %
P.1=AND(K.1,E.1,L.1) % SYM 5 %
STROBE=CONF(A.1,VCC) % SYM 6 %
M.1=INP(COINDROP) % SYM 7 %
D.1=INP(CLOCK) % SYM 8 %
N.1=INP(CUPFULL) % SYM 9 %
I.1=INP(RESET) % SYM 10 %
J.1=INP(ENABLE) % SYM 11 %
C.1=NOT(D.1) % SYM 12 %
L.1=NOT(N.1) % SYM 13 %
R.1=NOT(E.1) % SYM 14 %
K.1=NOT(F.1) % SYM 15 %
O.1=OR(P.1,Q.1) % SYM 16 %
DROPCUP,F.1=RORF(G.1,D.1,H.1,I.1,J.1) % SYM 17 %
POURDRNK,E.1=RORF(O.1,D.1,H.1,I.1,J.1) % SYM 18 %
B.1=XOR(E.1,F.1) % SYM 19 %

EQUATIONS:
G = (K & L & M);
H = (F & E);
O = (P # Q);
END$
```

Figure B-25. Example netlist in the IntelADF format.

Intergraph (FINTERGR.EXE)

The FINTERGR.EXE format file is used to produce netlists in the Intergraph format.

Intergraph netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers can have up to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a slash (/) and the sheet number to any labels.

Example

The netlist in figure B-26 was created—with no options selected—from the schematic in figure B-1.

```
%PART
14DIP300        U1
14DIP300        U2
%NET
N00001                  U1-8 U2-1
Q/1                     U1-6 U2-2 U1-1
N00003                  U1-5 U1-3
VCC                     U2-14 U1-14
A                       U1-9 U1-10
GND                     U2-7 U1-7
B                       U1-4
OUT                     U2-3
CLOCK                   U1-2
$
```

Figure B-26. Example netlist in the Intergraph format.

Mentor (FMENTOR.EXE)

The FMENTOR.EXE format file is used to produce netlists in the Mentor format.

Mentor netlists have the following characteristics:

- Part names, module names, and reference strings are limited to nineteen characters.
- Node names and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

File Options

In addition to the netlist file, IFORM also creates a component file when you select the Mentor format. You must enter a component filename in the **Destination 2** entry box.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to any labels.

Examples The netlist in figure B-27 and the corresponding component file in figure B-28 were created—with no options selected—from the schematic in figure B-1.

```
NET 'N00001' U1-8 U2-1
NET 'Q_1' U1-6 U2-2 U1-1
NET 'N00003' U1-5 U1-3
NET 'VCC' U2-14 U1-14
NET 'A' U1-9 U1-10
NET 'GND' U2-7 U1-7
NET 'B' U1-4
NET 'OUT' U2-3
NET 'CLOCK' U1-2
```

Figure B-27. Example netlist in the Mentor format.

```
# OrCAD Formatted Netlist for MENTOR Board Station V6
# Reference          .... Value Field          Module
Field
U1                          PART 74LS00          14DIP300
U2                          PART 74LS32          14DIP300
```

Figure B-28. Example component file in the Mentor format.

MultiWire (FMULTIWI.EXE)

The FMULTIWI.EXE format file is used to produce netlists in the MultiWire format.

MultiWire netlists have the following characteristics:

- Part names and module names are not checked for length.
- Reference strings and pin numbers together are limited to thirty-two characters.
- Node names are limited to sixteen characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example The netlist in figure B-29 was created—with no options selected—from the schematic in figure B-1.

```
N00001          U1              8
N00001          U2              1
Q-1             U1              6
Q-1             U2              2
Q-1             U1              1
N00003          U1              5
N00003          U1              3
VCC             U2             14
VCC             U1             14
A               U1              9
A               U1             10
GND             U2              7
GND             U1              7
B               U1              4
OUT             U2              3
CLOCK           U1              2
-1
```

Figure B-29. Example netlist in the MultiWire format.

OrCAD Hardware Description Language (FOHDLNET.EXE)

The FOHDLNET.EXE format file is used to produce netlists in the OrCAD hardware description language (OHDL) format.

OHDL netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore(_) and the sheet number to any labels.

Constraints

Make sure **Schematic Design Tools 386+** is configured to use the OrCAD-supplied PLDGATES.LIB or TTL.LIB part library. See the *Library Options* section in *Chapter 1: Configure Schematic Tools* for information on configuring libraries.

Example The netlist in figure B-30 was created—with no options selected—from the schematic in figure B-6.

```
|| CNTMUX                                                    Revised:   July  8, 1994
||                                                          Revision:
|| OrCAD
|| OrCAD
|| 9300 SW Nimbus Avenue
|| Beaverton, OR  97008
|| (503) 671-9500 Sales & Administration
|| (503) 671-9400 Technical Support }
| Type: "IFX780_132"
|
|
|  Netlist:
|  {
|   G04 (COUNT,N00020)  | U9
|   G04 (LOAD,N00016)  | U10
|   G169 (UP,CLK,I0,I1,I2,I3,GND,
|        -,N00016,N00020,Q3_1,Q2_1,Q1_1,Q0_1,N00012)  | U1
|   G169 -(UP,CLK,I4,I5,I6,I7,GND,
|        -,N00016,N00012,Q7_1,Q6_1,Q5_1,Q4_1,-)  | U2
|   G169 (UP,CLK,I8,I9,I10,I11,GND,
|        -,N00016,N00020,Q11_1,Q10_1,Q9_1,Q8_1,N00053)  | U3
|   G169 (UP,CLK,I12,I13,I14,I15,GND,
|        -,N00016,N00053,Q15_1,Q14_1,Q13_1,Q12_1,-)  | U4
|   G257 (SELECT,Q0_1,I0,O0,Q1_1,I1,O1,-,O2,I2,Q2_1,O3,I3,Q3_1,GND)  | U5
|   G257 (SELECT,Q4_1,I4,O4,Q5_1,I5,O5,-,O6,I6,Q6_1,O7,I7,Q7_1,GND)  | U6
|   G257 (SELECT,Q8_1,I8,O8,Q9_1,I9,O9,
|        -,O10,I10,Q10_1,O11,I11,Q11_1,GND)  | U7
|   G257 (SELECT,Q12_1,I12,O12,Q13_1,I13,O13,
|        -,O14,I14,Q14_1,O15,I15,Q15_1,GND)  | U8
|   PAD (I10,"IN")  | PAD1
|   PAD (I0,"IN")  | PAD2
|   PAD (I1,"IN")  | PAD3
|   PAD (I2,"IN")  | PAD4
|   PAD (I3,"IN")  | PAD5
|   PAD (I4,"IN")  | PAD6
|   PAD (I5,"IN")  | PAD7
|   PAD (I6,"IN")  | PAD8
|   PAD (I7,"IN")  | PAD9
|   PAD (I8,"IN")  | PAD10
|   PAD (I9,"IN")  | PAD11
|   PAD (I11,"IN")  | PAD12
|   PAD (I12,"IN")  | PAD13
|   PAD (I13,"IN")  | PAD14
|   PAD (I14,"IN")  | PAD15
|   PAD (I15,"IN")  | PAD16
```

Figure B-30. Example netlist in the OHDL format (continues).

```
|    PAD (LOAD,"IN")   | PAD17
|    PAD (CLK,"IN")    | PAD18
|    PAD (UP,"IN")     | PAD19
|    PAD (COUNT,"IN")   | PAD20
|    PAD (SELECT,"IN")   | PAD21
|    PAD (O0,"OUT")    | PAD38
|    PAD (O1,"OUT")    | PAD39
|    PAD (O2,"OUT")    | PAD40
|    PAD (O3,"OUT")    | PAD41
|    PAD (O4,"OUT")    | PAD42
|    PAD (O5,"OUT")    | PAD43
|    PAD (O6,"OUT")    | PAD44
|    PAD (O7,"OUT")    | PAD45
|    PAD (O8,"OUT")    | PAD46
|    PAD (O9,"OUT")    | PAD47
|    PAD (O10,"OUT")    | PAD48
|    PAD (O11,"OUT")    | PAD49
|    PAD (O12,"OUT")    | PAD50
|    PAD (O13,"OUT")    | PAD51
|    PAD (O14,"OUT")    | PAD52
|    PAD (O15,"OUT")    | PAD53
|   }
|
|Vectors:
|{ Display COUNT, LOAD, SELECT, CLK, \
|           (I[15..8])d, (O[15..8])d, \
|           (I[7..0])d,    (O[7..0])d
|
| Test LOAD=1; CLK
| Test LOAD=0; COUNT=1; UP=1; CLK=25(0,1)
| Set  I[15..8] = 10
| Set  I[7..0]  = 11
| Test SELECT=1,0
| Test LOAD=0; COUNT=1; UP=0; CLK=25(0,1)
| Test SELECT=1,0
| End }
```

Figure B-30. Example netlist in the OHDL format (continued).

OrCAD/PCB II (FPCBII.EXE)

The FPCBII.EXE format file is used to produce netlists in the OrCAD/PCB II format. Netlists in this format provide backward compatibility to OrCAD/PCB II. **PC Board Layout Tools**, Release IV, uses the connectivity database directly, so no formatting is required. For more information on transferring to **PC Board Layout Tools**, see *Chapter 34: To Layout*, and the *PC Board Layout Tools User's Guide.*

OrCAD/PCB II netlists have the following characteristics:

- Part names, module names, reference strings, and node names are limited to eight characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are limited to four characters.
- Pin numbers are not checked for length.
- All ASCII characters are legal except these:

  ( ) { }

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to any labels.

Example The netlist in figure B-31 was created—with no options selected—from the schematic in figure B-1.

```
(
 { OrCAD/PCB II Netlist Format
Generic Netlist Example  Revised:   June 30, 1994
OrCAD-01                 Revision: A
OrCAD
9300 SW Nimbus Avenue
Beaverton, OR  97008
(503) 671-9500 Sales & Administration
(503) 671-9400 Technical Support }

 ( 6CB84CBA 14DIP300 U1 74LS00
  ( 1 Q_1 )
  ( 2 CLOCK )
  ( 3 N00003 )
  ( 4 B )
  ( 5 N00003 )
  ( 6 Q_1 )
  ( 7 GND )
  ( 8 N00001 )
  ( 9 A )
  ( 10 A )
  ( 11 ?1 )
  ( 12 ?2 )
  ( 13 ?3 )
  ( 14 VCC )
 )
 ( 6E46169D 14DIP300 U2 74LS32
  ( 1 N00001 )
  ( 2 Q_1 )
  ( 3 OUT )
  ( 4 ?4 )
  ( 5 ?5 )
  ( 6 ?6 )
  ( 7 GND )
  ( 8 ?7 )
  ( 9 ?8 )
  ( 10 ?9 )
  ( 11 ?10 )
  ( 12 ?11 )
  ( 13 ?12 )
  ( 14 VCC )
 )
)
```

Figure B-31. Example netlist in the OrCAD/PCB II format.

OrCAD Programmable Logic Design Tools (FPLDNET.EXE)

The FPLDNET.EXE format file is used to produce netlists in the OrCAD Programmable Logic Design Tools format. This netlist format is used only when defining **Programmable Logic Design Tools** logic graphically. See the *Programmable Logic Design Tools User's Guide* and the *Programmable Logic Design Tools Reference Guide* for details.

OrCAD's **Programmable Logic Design Tools** netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

- ❑ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore(_) and the sheet number to any labels.

- ❑ `Create complete PLD source file`

  Tells IFORM to create a PLD source file in addition to a netlist. Any PLD vectors extracted from the schematic are placed in the PLD source file.

- ❑ `Only use 'pins' and 'nodes' keywords (do not use in, out, and i/o)`

  Tells IFORM to use only `pins` and `nodes` as keywords in the PLD source file. If you select this option, IFORM creates a complete PLD source file, as if you had also selected the **Create complete PLD source file** option.

Constraints When you create a PLD netlist, you must configure **Schematic Design Tools 386+** to use the OrCAD-supplied PLDGATES.LIB part library. You can only use the parts provided in PLDGATES.LIB to create the schematic.

Example The netlist in figure B-32 was created—with no options selected—from the schematic in figure B-5.

```
|| PLD Netlist Example     Revised:   June 30, 1994
|| OrCAD-05                Revision: A
|| OrCAD
|| 9300 SW Nimbus Avenue
|| Beaverton, OR  97008
|| (503) 671-9500 Sales & Administration
|| (503) 671-9400 Technical Support
||
| Netlist:  B0,A0,A1,B1
|           ->
|           Y0,Y1,Y2,Y3
|  {
|   G04 (A0,N00001)   | U1
|   G04 (A1,N00004)   | U2
|   G04 (B0,N00007)   | U3
|   G04 (B1,N00010)   | U4
|   G04 (A0,N00012)   | U5
|   G04 (B0,N00014)   | U6
|   G08 (B0,A0,Y0)   | U7
|   G11 (N00001,B0,-,-,-,-,-,-,-,-,-,N00002,A1) .| U8
|   G11 (B1,N00004,-,-,-,-,-,-,-,-,-,N00005,A0) .| U9
|   G11 (B1,N00007,-,-,-,-,-,-,-,-,-,N00008,A0) .| U10
|   G11 (N00010,B0,-,-,-,-,-,-,-,-,-,N00009,A1) .| U11
|   G11 (B1,A1,-,-,-,-,-,-,-,-,-,N00011,N00012) .| U12
|   G11 (B1,N00014,-,-,-,-,-,-,-,-,-,N00013,A1) .| U13
|   G21 (A0,A1,-,B0,B1,Y3)   | U14
|   G32 (N00011,N00013,Y2)   | U15
|   G32 (N00008,N00009,N00006)   | U16
|   G32 (N00002,N00005,N00003)   | U17
|   G32 (N00003,N00006,Y1)   | U18
|  }
```

Figure B-32. Example netlist in the PLD Netlist format.

OrCAD Digital Simulation Tools Model (FVSTMODE.EXE)

The FVSTMODE.EXE format file is used to produce netlists for OrCAD **Digital Simulation Tools** modeling. See the *Digital Simulation Tools User's Guide* for details.

OrCAD **Digital Simulation Tools** Model netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers and pin names are not used.
- All ASCII characters are legal.

Format Specific Options

☐ `Suppress comments`

If you select this option, IFORM does not append the primitive's reference designator to the end of the statement (see figure B-33).

Pipe commands

Lines of text may be placed in your schematic, to be included in the VSTModel netlist. Use **Draft's PLACE Text** command to place the text in your schematic, or use an editor to put the text in a text file, then use **Draft's BLOCK ASCII Import** command to import the contents of the file and place it on the worksheet.

Each line of text must start with the *pipe* character (|). The first line must be:

```
|VST_MODEL
```

This tells IFORM to extract the information in the following lines of text when generating a VSTModel netlist. The remaining lines can contain a header, comments, and directives compatible with OrCAD's **Digital Simulation Tools Add Device Model** device modeling language. For details on the **Add Device Model** Language, see the *Digital Simulation Tools User's Guide.*

Constraints When you create a VSTModel netlist, you must have **Schematic Design Tools 386+** configured to use the OrCAD-supplied VSTGATES.LIB, VSTRAM.LIB, VSTROM.LIB, and VSTOTHER.LIB part libraries. You can only use the parts provided in these libraries to create the schematic.

Example The netlist in figure B-33 was created—with no options selected—from the schematic in figure B-3.

```
;
;2 TO 1 MULTIPLEXOR
;
:21MUX          MY_LIB      4
LINV(P3;L1);M1
LAND(P2,P1;L2);M2
LNOR(L1,P1;L3);M3
SET(RHIGH)
OR(L2,L3;P4;10,50,15,50);M4
%
```

Figure B-33. Example netlist in the VSTModel format.

PADS 2000 (FPADS2K.EXE)

The FPADS2K.EXE format file is used to produce netlists in the PADS 2000 format.

PADS 2000 netlists have the following characteristics:

- Part names, module names, and pin numbers are not checked for length.
- Reference strings are limited to six characters.
- Node names are limited to forty-seven characters.
- Legal characters for reference strings and node names are limited to:

```
~   !   #   $   %   _   -   =
+   |   /   .   :   ;   <   >
A..Z   a..z   0..9
```

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except as noted for reference strings and node names.

Format Specific Options

❑ `Keep system generated netnames`

If you select this option, IFORM creates a persistent netname (.PNN) file. If the .PNN file already exists, IFORM compares any unnamed nets in the two versions. For each unnamed net in the new file, if more than half of the nodes in that net are in a single unnamed net in the old file, IFORM assigns that new net the same name as its counterpart in the old file. *Chapter 10: Creating a netlist* contains more information about persistent netnames and the effect of this option.

Without this option, IFORM does not create a .PNN file or check for an existing .PNN file. Instead, IFORM generates names for any unnamed nets as they are found.

- ❑ Do not append sheet number to labels

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example The netlist in figure B-34 was created—with **Keep system generated netnames** selected—from the schematic in figure B-1.

```
*PADS2000*
*PART*
U1      14DIP300
U2      14DIP300

*NET*
*SIGNAL* N00001
U1.8 U2.1
*SIGNAL* Q-1
U1.6 U1.1 U2.2
*SIGNAL* N00003
U1.5 U1.3
*SIGNAL* VCC
U2.14 U1.14
*SIGNAL* A
U1.9 U1.10
*SIGNAL* GND
U2.7 U1.7
*SIGNAL* B
U1.4
*SIGNAL* OUT
U2.3
*SIGNAL* CLOCK
U1.2
*END*
```

Figure B-34. Example connection list in the PADS 2000 format.

PADS ASCII (FPADSASC.EXE)

The FPADSASC.EXE format file is used to produce netlists in the PADS ASCII format. This is the older of two PADS ASCII formats. The next section describes the newer format.

PADS ASCII netlists have the following characteristics:

- Part names, module names, and pin numbers are not checked for length.
- Reference strings are limited to six characters from the following set:

```
A..Z   a..z   0..9
```

- Node names are limited to six characters from the following set:

```
~   !   #   $   %   _   -   =
+   |   /   .   :   ;   <   >
A..Z   a..z   0..9
```

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except as noted for reference strings and node names.

File Options

In addition to the connection list file, IFORM also creates a part list file when you select PADS ASCII format. You must enter a second filename in the **Destination 2** entry box on the **Configure Netlist Format** screen.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example The netlist file in figure B-35 and the corresponding part list in figure B-36 were created—with no options selected—from the schematic in figure B-1.

```
PADS-PCB
*Net
*Sig N1
U1.8 U2.1
*Sig Q-1
U1.6 U2.2 U1.1
*Sig N00003
U1.5 U1.3
*Sig VCC
U2.14 U1.14
*Sig A
U1.9 U1.10
*Sig GND
U2.7 U1.7
*Sig B
U1.4
*Sig OUT
U2.3
*Sig CLOCK
U1.2
*End
```

Figure B-35. Example connection list in the older PADS ASCII format (FPADSASC.EXE).

```
*PADS-PCB
*Part*
U1      14DIP300
U2      14DIP300
*End
```

Figure B-36. Example part list in the older PADS ASCII format (FPADSASC.EXE).

PADS ASCII (FPADSASØ.EXE)

The FPADSASØ.EXE format file is used to produce netlists in the PADS ASCII format. The resulting netlist is identical to the netlist IFORM creates using the FPADSASC.EXE format file. The characteristics and options described in the preceding section also apply to FPADSASØ.EXE, except that node names may have up to twelve characters instead of six.

PADS-PCB (FPADSPCB.EXE)

The FPADSPCB.EXE format file is used to produce netlists in the PADS-PCB format.

PADS-PCB netlists have the following characteristics:

- Part names, module names, and pin numbers are not checked for length.
- Reference strings are limited to six characters.
- Node names are limited to twelve characters.
- Legal characters for reference strings and node names are limited to:

```
~   !   #   $   %   _   -   =
+   |   /   .   :   ;   <   >
A..Z   a..z   0..9
```

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except as noted for reference strings and node names.

Format Specific Options

❑ `Keep system generated netnames`

If you select this option, IFORM creates a persistent netname (.PNN) file. If the .PNN file already exists, IFORM compares any unnamed nets in the two versions. For each unnamed net in the new file, if more than half of the nodes in that net are in a single unnamed net in the old file, IFORM assigns that new net the same name as its counterpart in the old file. *Chapter 10: Creating a netlist* contains more information about persistent netnames and the effect of this option.

Without this option, IFORM does not create a .PNN file or check for an existing .PNN file. Instead, IFORM generates names for any unnamed nets as they are found.

❑ Do not append sheet number to labels

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example The netlist in figure B-37 was created—with **Keep system generated netnames** selected—from the schematic in figure B-1.

```
*PADS-PCB*
*PART*
U1      14DIP300
U2      14DIP300

*NET*
*SIGNAL* N00001
U1.8 U2.1
*SIGNAL* Q-1
U1.6 U1.1 U2.2
*SIGNAL* N00003
U1.5 U1.3
*SIGNAL* VCC
U2.14 U1.14
*SIGNAL* A
U1.9 U1.10
*SIGNAL* GND
U2.7 U1.7
*SIGNAL* B
U1.4
*SIGNAL* OUT
U2.3
*SIGNAL* CLOCK
U1.2
*END*
```

Figure B-37. Example connection list in the PADS-PCB format.

PCAD (FPCAD.EXE)

The FPCAD.EXE format file is used to produce netlists in the PCAD format.

PCAD netlists have the following characteristics:

- Part names, module names, reference strings, and pin numbers are not checked for length.
- Node names are limited to eight characters.
- Node numbers are limited to five digits following the "NET" prefix.
- Pin names are not used.
- Characters are not checked for legality.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to any labels.

- ☐ `Include unconnected pins`

  If you select this option, IFORM assigns all unconnected pins a unique net. If you do not select this option, IFORM does not assign a net and reports a warning.

Example

The netlist in figure B-38 was created—with no options selected—from the schematic in figure B-1.

```
{COMPONENT ORCAD.PCB
 {ENVIRONMENT LAYS.PCB}
  {PDIFvrev 1.30}
  {DETAIL
   {SUBCOMP
{I 14DIP300.PRT U1
{CN
     1 Q_1
     2 CLOCK
     3 NET00003
     4 B
     5 NET00003
     6 Q_1
     7 GND
     8 NET00001
     9 A
    10 A
    11 ?
    12 ?
    13 ?
    14 VCC
}
}
{I 14DIP300.PRT U2
{CN
     1 NET00001
     2 Q_1
     3 OUT
     4 ?
     5 ?
     6 ?
     7 GND
     8 ?
     9 ?
    10 ?
    11 ?
    12 ?
    13 ?
    14 VCC
}
}
}
}
}
```

Figure B-38. Example netlist in the PCAD format.

PCADnlt (FPCADNLT.EXE)

The FPCADNLT.EXE format file is used to produce netlists in the PCADnlt format.

PCADnlt netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Legal characters for node names are limited to:

  ```
  $   -   +   _ (underscore)
  A..Z   a..z   0..9
  ```

- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except as noted for node names.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM adds an underscore(_) and the sheet number to any labels.

Example

The netlist in figure B-39 was created—with no options selected—from the schematic in figure B-1.

```
% Generic Netlist Example Revised:   June 30, 1994
% OrCAD-01                Revision: A
% OrCAD
% 9300 SW Nimbus Avenue
% Beaverton, OR  97008
% (503) 671-500 Sales & Administration
% (503) 671-9400 Technical Support
BOARD = ORCAD.PCB;

PARTS
14DIP300          = U1,    % 74LS00
                    U2;    % 74LS32

NETS

N00001      = U1/8 U2/1 ;
Q_1         = U1/6 U2/2 U1/1 ;
N00003      = U1/5 U1/3 ;
VCC         = U2/14 U1/14 ;
A           = U1/9 U1/10 ;
GND         = U2/7 U1/7 ;
B           = U1/4 ;
OUT         = U2/3 ;
CLOCK       = U1/2 ;
```

Figure B-39. Example netlist in the PCADnlt format.

PDUMP (FPDUMP.EXE)

The FPDUMP.EXE format file is used to produce a parts list containing all the information on the schematic sheets. No information is omitted or changed. You can use this netlist format when troubleshooting a design.

RacalRedac (FRACALRE.EXE)

The FRACALRE.EXE format file is used to produce netlists in the RacalRedac format.

RacalRedac netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

File Options

In addition to the netlist file, IFORM also creates a component file when you select the RacalRedac format. You must enter a component filename in the **Destination 2** entry box on the **Configure Netlist Format** screen.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM adds an underscore(_) and the sheet number to any labels.

Examples

The netlist in figure B-40 and the corresponding component file in figure B-41 were created—with no options selected—from the schematic in figure B-1.

```
.PCB
.REM Generic Netlist Example     Revised:   June 30, 1994
.REM OrCAD-01                    Revision: A
.REM OrCAD
.REM 9300 SW Nimbus Avenue
.REM Beaverton, OR  97008
.REM (503) 671-9500 Sales & Administration
.REM (503) 671-9400 Technical Support
.CON
.COD 2

.REM N00001
U1 8 U2 1
.REM Q_1 (local to sheet 1)
U1 6 U2 2 U1 1
.REM N00003
U1 5 U1 3
.REM VCC
U2 14 U1 14
.REM A
U1 9 U1 10
.REM GND
U2 7 U1 7
.REM B
U1 4
.REM OUT
U2 3
.REM CLOCK
U1 2
.EOD
```

Figure B-40. Example netlist in the RacalRedac format.

```
.PCB
.REM Generic Netlist Example     Revised:   June 30, 1994
.REM OrCAD-01                    Revision: A
.REM OrCAD
.REM 9300 SW Nimbus Avenue
.REM Beaverton, OR  97008
.REM (503) 671-9500 Sales & Administration
.REM (503) 671-9400 Technical Support
.COM
.REF

.REM 74LS00
U1   14DIP300
.REM 74LS32
U2   14DIP300
.EOD
```

Figure B-41. Example component list in the RacalRedac format.

Scicards (FSCICARD.EXE)

The FSCICARD.EXE format file is used to produce netlists in the Scicards format.

Scicards netlists have the following characteristics:

- Part names are limited to seventeen characters.
- Module names are limited to fifteen characters.
- Reference strings and pin numbers combined are limited to twelve characters.
- Pin numbers are limited to three characters.
- Node names are limited to eight characters.
- Node numbers are not checked for length.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM adds a dash (-) and the sheet number to any labels.

Example

The netlist in figure B-42 was created—with no options selected—from the schematic in figure B-1.

```
PARTS LIST

74LS00              14DIP300          U1
74LS32              14DIP300          U2
EOS
NET LIST

NODE      1                    $
    U1          8  U2            1
NODENAME Q-1                   $
    U1          6  U2            2  U1          1
NODE      3                    $
    U1          5  U1            3
NODENAME VCC                   $
    U2         14  U1           14
NODENAME A                     $
    U1          9  U1           10
NODENAME GND                   $
    U2          7  U1            7
NODENAME B                     $
    U1          4
NODENAME OUT                   $
    U2          3
NODENAME CLOCK                 $
    U1          2
EOS
```

Figure B-42. Example netlist in the Scicards format.

SPICE (FSPICE.EXE)

The FSPICE.EXE format file is used to produce flat netlists in the SPICE format. The hierarchical SPICE netlist format file is discussed later in this appendix.

SPICE netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits
- If the **Use node names** option is selected, legal characters for node names are limited to:

  `0..9  A..Z  a..z  $  _` (underscore)

- All ASCII characters are legal except as noted for node names.

File Options

In addition to the netlist file, IFORM also creates a map file when you select the SPICE format. The node numbers created by IFORM are placed in the .MAP file so you can cross reference the SPICE node numbers with the node names that you specified on your schematic. You must enter the map filename in the **Destination 2** entry box on the **Configure Netlist Format** screen.

△ *NOTE: If you select the **Use node names** option, the map file IFORM creates is invalid.*

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to any labels.

- `Use node names`

  Tells IFORM to use the node names you placed on the schematic (via labels and module ports) where available. Not all versions of SPICE support alphanumeric node names. Check your SPICE manual for details. If your version of SPICE does not allow alphanumeric node names, you can still give them numeric names such as "17." These numeric names do not interfere with the ones generated by IFORM, since IFORM-generated node numbers begin at 10000, except GND, which is always 0.

- `Include unconnected pins`

  Tells IFORM to assign node numbers to all unconnected pins. Node numbers for unconnected pins begin at 32,767 and decrease in value. If you do not select this option and there are unconnected pins on your schematic, they are assigned a space character and IFORM displays a warning.

Pipe commands

You can place lines of text in your schematic, to be included in the SPICE netlist. Use **Draft's PLACE Text** command to place the text in your schematic, or use an editor to put the text in an ASCII file, then use **Draft's BLOCK ASCII Import** command to import the contents of the file and place it on the worksheet.

Each line of text must start with the *pipe* character (|). The first line must be:

```
|SPICE
```

This tells IFORM to extract the information in the following lines of text when generating a SPICE netlist. The remaining lines can contain any information you want to include in the netlist. The lines following `|SPICE` are placed at the top of the netlist.

Constraints

Schematic Design Tools 386+ can create netlists larger than most PC-based SPICE programs accept. Consult your SPICE manual for the limits. If your PC meets SPICE's memory requirements, you can generate the largest netlist allowed.

The part value is used to pass modeling information to the netlist. For instance, resistor RS1 in figure B-4 has a value of 1K Ohms.

Use the special PSPICE.LIB or SPICE.LIB libraries supplied by OrCAD when generating a SPICE netlist. These libraries already have pin numbers on the parts and are compatible with most versions of SPICE. The PSPICE.LIB contains many specific part types, such as a 2N2222 NPN transistor, that are not provided in the generic SPICE.LIB.

To modify or create your own SPICE library, you must support the proper model pin numbers. To implement this, use the **Decompile Library** or **Edit Library** librarians and convert the desired OrCAD-supplied libraries into a library source file. Make the appropriate changes to the source file and recompile the modified library back to an object file using **Edit Library**.

All library part pin names should be changed to reflect the model node index. To find out the proper node ordering, see your SPICE manual.

As an example of what to change, the OrCAD-supplied NPN transistor has the pin names defined as base, emitter, and collector in the DEVICE.LIB library. For SPICE to understand the nodal information, the pin names must be changed from base, emitter, and collector to 2, 3, and 1 (as defined in the SPICE manual). Figure B-43 shows the library source file for an NPN transistor that is compatible with the SPICE pin-numbering convention.

```
'NPN'
REFERENCE   'Q'
{X Size =} 2  {Y Size =} 2 {Parts per Package =} 0
L1    SHORT    IN     '2'
B2    SHORT    IN     '3'
T2    SHORT    IN     '1'
{  0}.........##..........#
{  1}          ##           #
                .
                .
                .
```

Figure B-43. Library source file for the NPN transistor.

Examples

The netlist in figure B-44 and the corresponding map file in figure B-45 were created—with no options selected—from the schematic in figure B-4.

```
* SPICE Example              Revised:   June 30, 1994
* OrCAD-04                  Revision: A
* OrCAD
* 9300 SW Nimbus Avenue
* Beaverton, OR  97008
* (503) 671-9500 Sales & Administration
* (503) 671-9400 Technical Support
.OPTIONS ACCT LIST NODE OPTS NOPAGE RELTOL=.001
.WIDTH OUT=80
.DC VIN -0.25 0.25 0.005
.AC DEC 10 1 10GHZ
.TRAN/OP 5NS 500NS
.MODEL QNL NPN (BF=80 RB=100 CJS=2PF
+      TF=0.3NS TR=6NS VAF=50)
.PRINT DC V(4) V(5)
.PLOT DC IC(Q2)
.PROBE
VCC 10007 0 DC 12
VIN 10008 0 AC 1 SIN(0 0.1 5MEG)
VEE 10010 0 DC -12
RC1 10007 10001 10K
RC2 10007 10002 10K
RS1 10004 0 1K
RS2 10008 10003 1K
RBIAS1 10007 10006 20K
CLOAD1 10001 10002 5PF
Q1 10005 10006 10010 QNL
Q2 10001 10003 10005 QNL
Q3 10002 10004 10005 QNL
Q4 10006 10006 10010 QNL
.END
```

Figure B-44. Example netlist in the SPICE format.

```
10001 4
10002 5
10005 6
10006 7
10007 VCC DC 12
10008 VIN AC 1 SIN(0 0.1 5MEG)
    0 GND
10010 VEE DC -12
```

Figure B-45. Example map file in the SPICE format.

Tango (FTANGO.EXE)

The FTANGO.EXE format file is used to produce netlists in the Tango format.

Tango netlists have the following characteristics:

- Part names, module names, reference strings, and node names are limited to 16 characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin numbers are not checked for length.
- Pin names are not used.
- Reference strings and module names must be uppercase characters.
- All ASCII characters are legal except:

  ( ) [ ] - ,

  and except as noted for reference strings and module names.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example

The netlist in figure B-46 was created—with no options selected—from the schematic in figure B-1.

```
Generic Netlist Example  Revised:   June 30, 1994
OrCAD-01                 Revision: A
OrCAD
9300 SW Nimbus Avenue
Beaverton, OR  97008
(503) 671-9500 Sales & Administration
(503) 671-9400 Technical Support
```

Figure B-46. Example netlist in the Tango format (continues).

```
[
U1
14DIP300
74LS00
]
[
U2
14DIP300
74LS32
]
(
N00001
U1,8
U2,1
)
(
Q-1
U1,6
U2,2
U1,1
)
(
N00003
U1,5
U1,3
)
(
VCC
U2,14
U1,14
)
(
A
U1,9
U1,10
)
(
GND
U2,7
U1,7
)
(
B
U1,4
)
(
OUT
U2,3
)
(
CLOCK
U1,2
)
```

Figure B-46. Example netlist in the Tango format (continued).

Tango1 (FTANGO1.EXE)

The FTANGO1.EXE format file is used to produce netlists in the Tango1 format.

Tango1 netlists have the following characteristics:

- Part names, module names, reference strings, and node names are limited to 16 characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin numbers are not checked for length.
- Pin names are not used.
- Reference strings and module names must be uppercase characters.
- All ASCII characters are legal except:

  ( ) [ ] - ,

  and except as noted for reference strings and module names.

Format Specific Options

❑ Do not append sheet number to labels

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a dash (-) and the sheet number to any labels.

Example

The netlist in figure B-47 was created—with no options selected—from the schematic in figure B-1.

```
Generic Netlist Example  Revised:   June 30, 1994
OrCAD-01                 Revision: A
OrCAD
9300 SW Nimbus Avenue
Beaverton, OR  97008
(503) 671-9500 Sales & Administration
(503) 671-9400 Technical Support
```

Figure B-47. Example netlist in the Tango1 format (continues).

```
[
U1
14DIP300
74LS00
]
[
U2
14DIP300
74LS32
]
(
N00001
U1,8
U2,1
)
(
Q-1
U1,6
U1,1
U2,2
)
(
N00003
U1,5
U1,3
)
(
VCC
U2,14
U1,14
)
(
A
U1,9
U1,10
)
(
GND
U2,7
U1,7
)
(
B
U1,4
)
(
OUT
U2,3
)
(
CLOCK
U1,2
)
```

Figure B-47. Example netlist in the Tango1 format (continued).

Telesis (FTELESIS.EXE)

The FTELESIS.EXE format file is used to produce netlists in the Telesis format.

Telesis netlists have the following characteristics:

- Part names, module names, reference strings, node names, and pin numbers are not checked for length.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a slash (/) and the sheet number to any labels.

Example

The netlist in figure B-48 was created—with no options selected—from the schematic in figure B-1.

```
$PACKAGES
14DIP300! 74LS00; U1
14DIP300! 74LS32; U2
$NETS
N00001;  U1.8 U2.1
Q/1;  U1.6 U2.2 U1.1
N00003;  U1.5 U1.3
VCC;  U2.14 U1.14
A;  U1.9 U1.10
GND;  U2.7 U1.7
B;  U1.4
OUT;  U2.3
CLOCK;  U1.2
$END
```

Figure B-48. Example netlist in the Telesis format.

Vectron (FVECTRON.EXE)

The FVECTRON.EXE format file is used to produce netlists in the Vectron format.

Vectron netlists have the following characteristics:

- Part names, module names, and pin numbers are not checked for length.
- Reference strings are limited to eight characters.
- Node names are limited to twelve characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

File Options

In addition to the netlist file, IFORM also creates a part list file when you select the Vectron format. You must enter a part list filename in the **Destination 2** entry box on the **Configure Netlist Format** screen.

Format Specific Options

❑ `Do not append sheet number to labels`

Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends a period (.) and the sheet number to any labels.

Examples

The netlist in figure B-49 and the corresponding part file in figure B-50 were created—with no options selected—from the schematic in figure B-1.

```
*N00001         U1 8 U2 1
*Q.1            U1 6 U2 2 U1 1
*N00003         U1 5 U1 3
*VCC            U2 14 U1 14
*A              U1 9 U1 10
*GND            U2 7 U1 7
*B              U1 4
*OUT            U2 3
*CLOCK          U1 2
```

Figure B-49. Example netlist in the Vectron format.

```
U1          14DIP300
U2          14DIP300
```

Figure B-50. Example part list file in the Vectron format.

WireList (FWIRELIS.EXE)

The FWIRELIS.EXE format file is used to produce netlists in the WireList format.

WireList netlists have the following characteristics:

- Part names and node names are not checked for length.
- Module names are limited to twenty-nine characters.
- Reference strings are limited to nine characters.
- Node numbers are limited to five digits following the "N" prefix.
- Pin numbers are limited to seven characters.
- Pin names are limited to fifteen characters.
- Legal characters for node numbers are `0..9`.
- Legal characters for pin numbers are `0..9` unless the **Do not output numbers for Grid Array parts** option is selected. If it is selected, any ASCII character is legal.
- All ASCII characters are legal except as noted for node numbers and pin numbers.

Format Specific Options

- ☐ `Do not append sheet number to labels`

  Use this option with caution. If you select this option, IFORM merges labels with the same name on different sheets into one net. You typically select this option only when the design consists of a single schematic sheet. If you do not select this option, IFORM appends an underscore (_) and the sheet number to any labels.

- ☐ `Do not output pin numbers for Grid Array parts`

  If you select this option, IFORM skips non-numerical pin numbers, such as those on grid array parts.

- ☐ `Abbreviate label descriptions`

  If you select this option, IFORM shortens label descriptions.

Example

The netlist in figure B-51 was created—with no options selected—from the schematic in figure B-1.

```
Wire List

Generic Netlist Example          Revised:   June 30, 1994
OrCAD-01                         Revision: A
OrCAD
9300 SW Nimbus Avenue
Beaverton, OR  97008
(503) 671-9500 Sales & Administration
(503) 671-9400 Technical Support

<<< Component List >>>
74LS00                           U1          14DIP300
74LS32                           U2          14DIP300

<<< Wire List >>>

  NODE   REFERENCE   PIN #    PIN NAME        PIN TYPE     PART VALUE

[00001] N00001
        U1          8        O_C             Output       74LS00
        U2          1        I0_A            Input        74LS32

[00002] Q_1 (local to sheet 1)
        U1          6        O_B             Output       74LS00
        U2          2        I1_A            Input        74LS32
        U1          1        I0_A            Input        74LS00

[00003] N00003
        U1          5        I1_B            Input        74LS00
        U1          3        O_A             Output       74LS00

[00004] VCC
        U2          14       VCC             Power        74LS32
        U1          14       VCC             Power        74LS00

[00005] A
        U1          9        I0_C            Input        74LS00
        U1          10       I1_C            Input        74LS00

[00006] GND
        U2          7        GND             Power        74LS32
        U1          7        GND             Power        74LS00

[00007] B
        U1          4        I0_B            Input        74LS00

[00008] OUT
        U2          3        O_A             Output       74LS32

[00009] CLOCK
        U1          2        I1_A            Input        74LS00
```

Figure B-51. Example netlist in the WireList format.

Hierarchical netlists

In order to create a netlist in one of the hierarchical formats, your schematic must have been processed by INET (linking is not necessary). The necessary steps are explained in *Chapter 12: Create Hierarchical Netlist.*

Hierarchical netlist format files are distinguished by a filename beginning with "H." They cannot be used within the **Create Netlist** processor (the two formats are similar, but process incompatible data structures).

HFORM uses the hierarchical netlist format files to create formatted netlists that contain either simple or complex hierarchies.

Example Schematics

The two schematics shown in figures B-52 and B-53 are used by the hierarchical EDIF netlist. The two schematics shown in figures B-54 and B-55 are used by the hierarchical SPICE netlist.

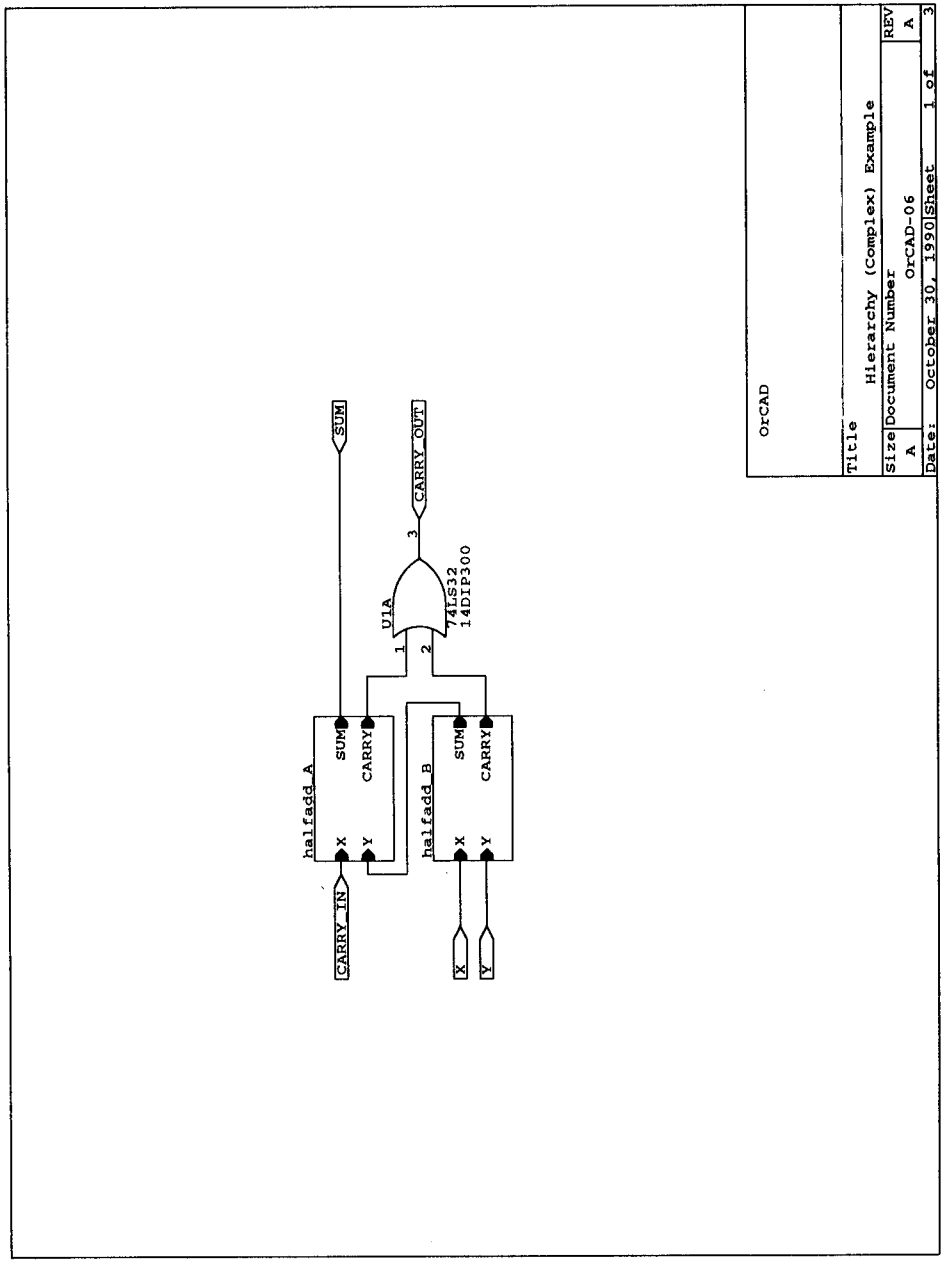


Figure B-52. Root sheet used by the hierarchical EDIF format.

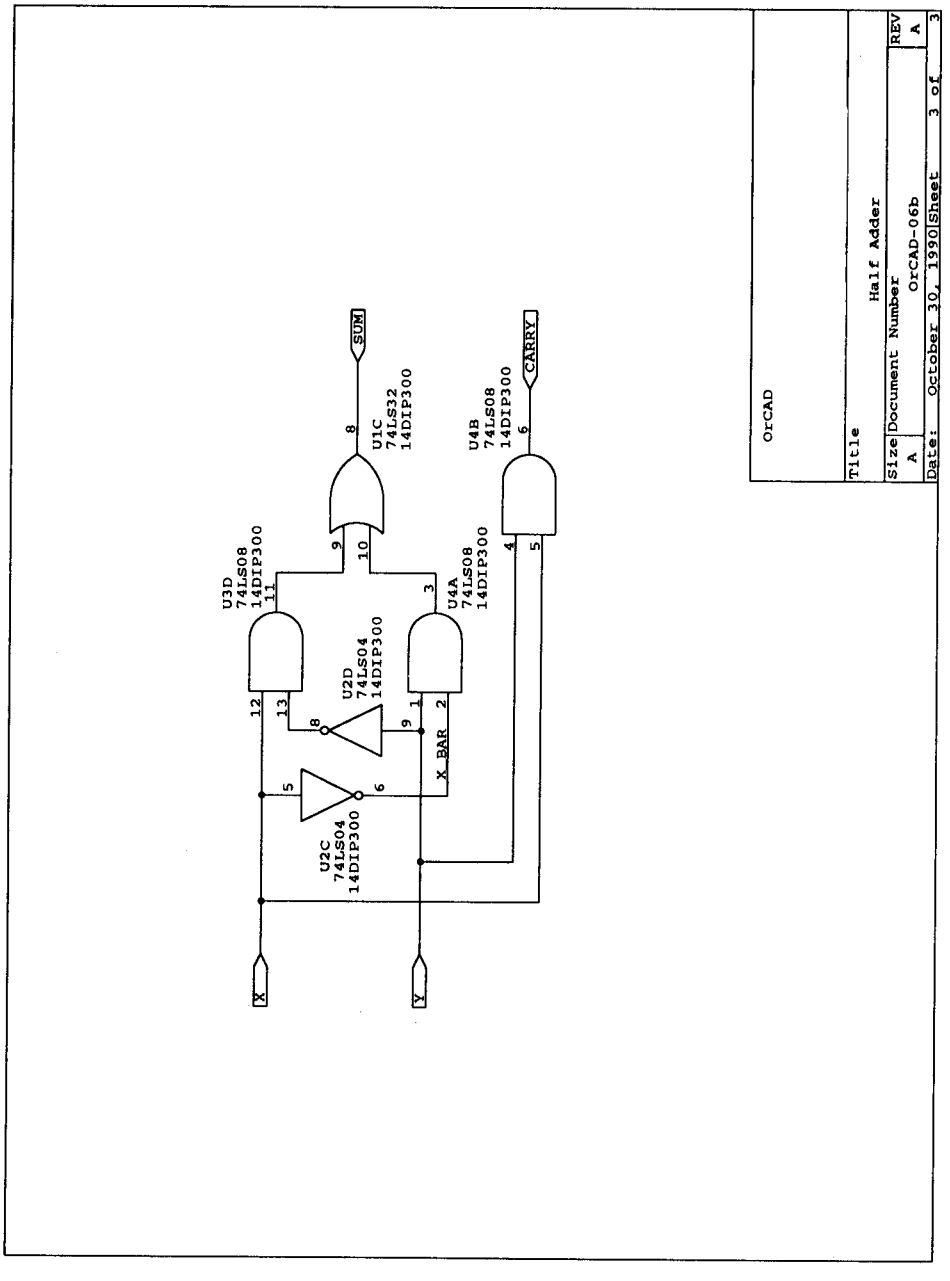


Figure B-53. Leaf sheet used by the hierarchical EDIF format.

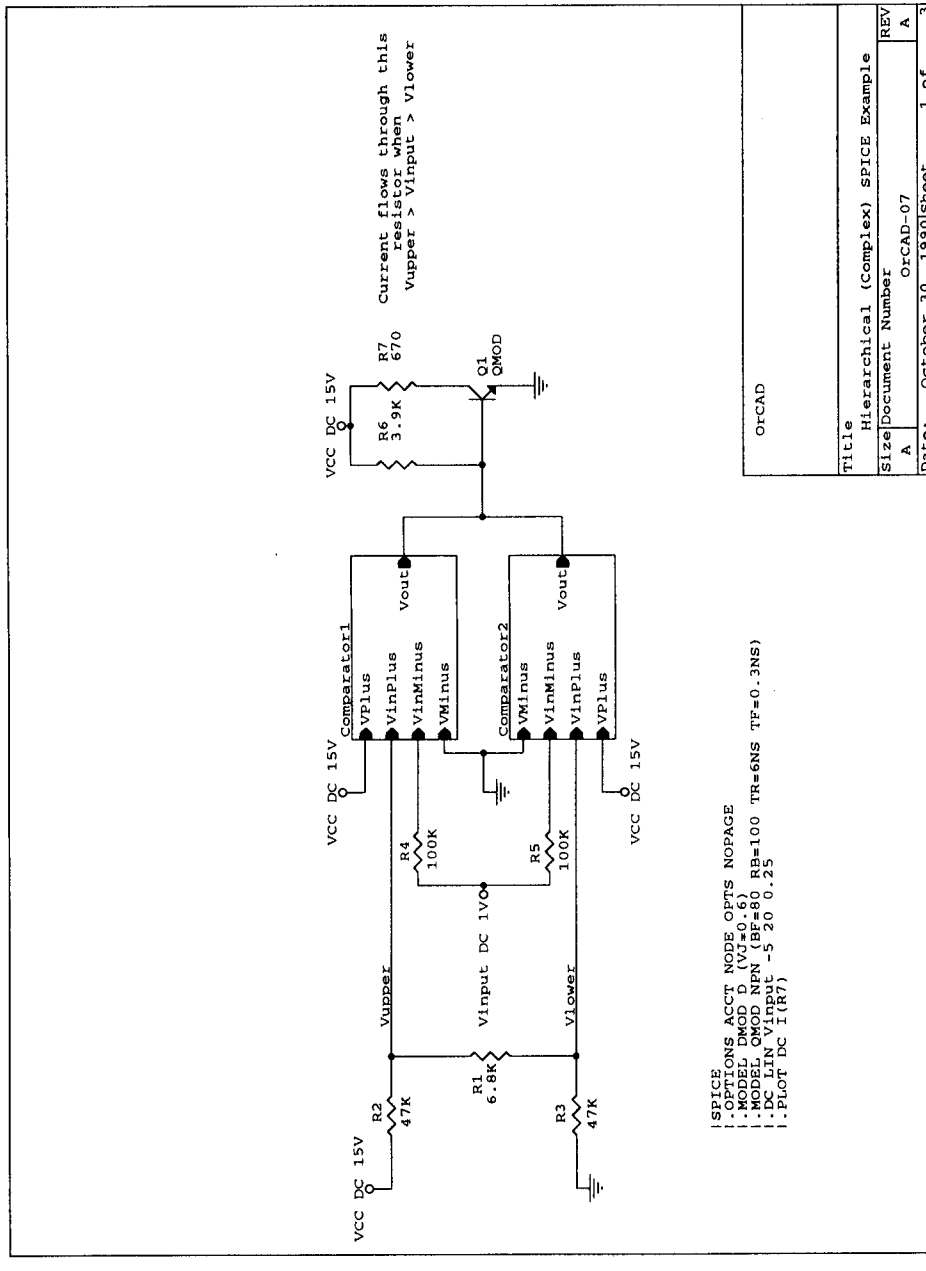


Figure B-54. Root sheet used by the hierarchical SPICE format.

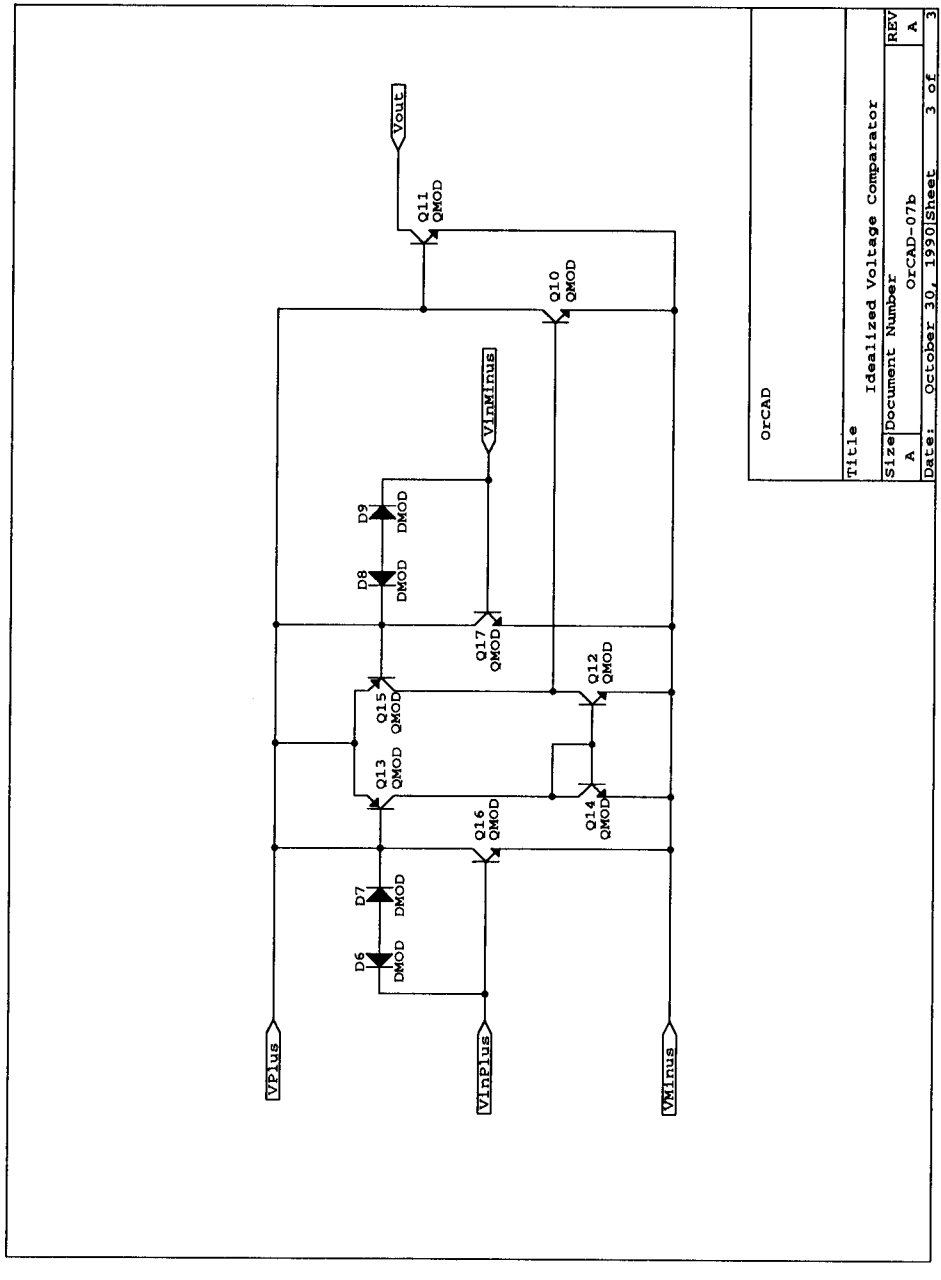


Figure B-55. Leaf sheet used by the hierarchical SPICE format.

EDIF (HEDIF.EXE)

This is one of two EDIF (Version 2 0 0) netlist format files supplied with OrCAD's **Schematic Design Tools 386+**. This netlist format file can only be used to create hierarchical EDIF netlists. The flat EDIF netlist format file is discussed earlier in this appendix.

Most of the options that apply to the flat EDIF format apply to the hierarchical version as well. The only difference is that HEDIF.EXE does not include an option for retaining system-generated netnames. See the section *EDIF (FEDIF.EXE)* for information about EDIF netlist characteristics and configuration options.

HFORM manages the hierarchy by turning sheets in the schematic into CELLs in the main LIBRARY. These cells can then be referred to by INSTANCE where needed. Because EDIF comprises a define-before-use philosophy, the hierarchy appears to be inverted in the netlist (the root sheet is the last CELL in the main LIBRARY).

Example

The netlist in figure B-56 was created—with no options selected—from the schematics shown in figures B-52 and B-53.

```
(edif &EX6
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap (keywordLevel 0))
 (status
  (written
   (timeStamp 0 0 0 0 0 0)
   (program "HFORM.EXE")
   (comment "Original data from OrCAD/SDT schematic"))
  (comment "Hierarchy (Complex) Example")
  (comment " June 30, 1994")
  (comment "OrCAD-06")
  (comment "A")
  (comment "OrCAD")
  (comment "9300 SW Nimbus Avenue")
  (comment "Beaverton, OR  97008")
  (comment "(503) 671-9500 Sales & Administration")
  (comment "(503) 671-9400 Technical Support"))
 (external OrCAD_LIB
  (edifLevel 0)
  (technology
   (numberDefinition
    (scale 1 1 (unit distance))))
  (cell &74LS04
   (cellType generic)
   (comment "From OrCAD library TTL.LIB")
   (view NetlistView
    (viewType netlist)
    (interface
     (port &I_A (direction INPUT))
     (port &I_B (direction INPUT))
     (port &I_C (direction INPUT))
     (port &I_D (direction INPUT))
     (port &I_E (direction INPUT))
     (port &I_F (direction INPUT))
     (port &O_A (direction OUTPUT))
     (port &O_B (direction OUTPUT))
     (port &O_C (direction OUTPUT))
     (port &O_D (direction OUTPUT))
     (port &O_E (direction OUTPUT))
     (port &O_F (direction OUTPUT))
     (port &VCC (direction INPUT))
     (port &GND (direction INPUT)))))
  (cell &74LS08
   (cellType generic)
   (comment "From OrCAD library TTL.LIB")
   (view NetlistView
    (viewType netlist)
    (interface
     (port &I0_A (direction INPUT))
```

Figure B-56. Example netlist in the hierarchical EDIF format (continues).

```
      (port &I0_B (direction INPUT))
      (port &I0_C (direction INPUT))
      (port &I0_D (direction INPUT))
      (port &I1_A (direction INPUT))
      (port &I1_B (direction INPUT))
      (port &I1_C (direction INPUT))
      (port &I1_D (direction INPUT))
      (port &O_A (direction OUTPUT))
      (port &O_B (direction OUTPUT))
      (port &O_C (direction OUTPUT))
      (port &O_D (direction OUTPUT))
      (port &VCC (direction INPUT))
      (port &GND (direction INPUT)))))
  (cell &74LS32
   (cellType generic)
   (comment "From OrCAD library TTL.LIB")
   (view NetlistView
    (viewType netlist)
    (interface
     (port &I0_A (direction INPUT))
     (port &I0_B (direction INPUT))
     (port &I0_C (direction INPUT))
     (port &I0_D (direction INPUT))
     (port &I1_A (direction INPUT))
     (port &I1_B (direction INPUT))
     (port &I1_C (direction INPUT))
     (port &I1_D (direction INPUT))
     (port &O_A (direction OUTPUT))
     (port &O_B (direction OUTPUT))
     (port &O_C (direction OUTPUT))
     (port &O_D (direction OUTPUT))
     (port &VCC (direction INPUT))
     (port &GND (direction INPUT))))))
 (library MAIN_LIB
  (edifLevel 0)
  (technology
   (numberDefinition
    (scale 1 1 (unit distance))))
  (cell &EX6B
   (cellType generic)
   (view NetlistView
    (viewType netlist)
    (interface
     (port &CARRY (direction OUTPUT))
     (port &SUM (direction OUTPUT))
     (port &X (direction INPUT))
     (port &Y (direction INPUT)))
    (contents
     (instance &U2
      (viewRef NetlistView
```

Figure B-56. Example netlist in the hierarchical EDIF format (continues).

```
      (cellRef &74LS04
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS04"))
     (property ModuleValue (string "14DIP300")))
    (instance &U3
     (viewRef NetlistView
      (cellRef &74LS08
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS08"))
     (property ModuleValue (string "14DIP300")))
    (instance &U4
     (viewRef NetlistView
      (cellRef &74LS08
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS08"))
     (property ModuleValue (string "14DIP300")))
    (instance &U1
     (viewRef NetlistView
      (cellRef &74LS32
       (libraryRef OrCAD_LIB)))
     (property PartValue (string "74LS32"))
     (property ModuleValue (string "14DIP300")))
    (net &VCC
     (joined
      (portRef &VCC (instanceRef &U4))
      (portRef &VCC (instanceRef &U1))
      (portRef &VCC (instanceRef &U2))
      (portRef &VCC (instanceRef &U3))))
    (net &X
     (joined
      (portRef &X)
      (portRef &I1B (instanceRef &U4))
      (portRef &IC (instanceRef &U2))
      (portRef &I0D (instanceRef &U3))))
    (net N00013
     (joined
      (portRef &OD (instanceRef &U3))
      (portRef &I0C (instanceRef &U1))))
    (net N00014
     (joined
      (portRef &OD (instanceRef &U2))
      (portRef &I1D (instanceRef &U3))))
    (net &GND
     (joined
      (portRef &GND (instanceRef &U4))
      (portRef &GND (instanceRef &U1))
      (portRef &GND (instanceRef &U3))
      (portRef &GND (instanceRef &U2))))
    (net &SUM
     (joined
```

Figure B-56. Example netlist in the hierarchical EDIF format (continues).

```
      (portRef &SUM)
      (portRef &OC (instanceRef &U1))))
    (net N00017
     (joined
      (portRef &I1C (instanceRef &U1))
      (portRef &OA (instanceRef &U4))))
    (net &X_BAR_3
     (joined
      (portRef &OC (instanceRef &U2))
      (portRef &I1A (instanceRef &U4))))
    (net &Y
     (joined
      (portRef &Y)
      (portRef &I0B (instanceRef &U4))
      (portRef &ID (instanceRef &U2))
      (portRef &I0A (instanceRef &U4))))
    (net &CARRY
     (joined
      (portRef &CARRY)
      (portRef &OB (instanceRef &U4)))))))
(cell &EX6
 (cellType generic)
 (view NetlistView
  (viewType netlist)
  (interface
   (port &CARRY_IN (direction INPUT))
   (port &CARRY_OUT (direction OUTPUT))
   (port &SUM (direction OUTPUT))
   (port &X (direction INPUT))
   (port &Y (direction INPUT)))
  (contents
   (instance &U1
    (viewRef NetlistView
     (cellRef &74LS32
      (libraryRef OrCAD_LIB)))
    (property PartValue (string "74LS32"))
    (property ModuleValue (string "14DIP300")))
   (instance &halfadd_A
    (viewRef NetlistView
     (cellRef &EX6B)))
   (instance &halfadd_B
    (viewRef NetlistView
     (cellRef &EX6B)))
   (net &CARRY_IN
    (joined
     (portRef &CARRY_IN)
     (portRef &X (instanceRef &halfadd_A))))
   (net &SUM
    (joined
     (portRef &SUM)
```

Figure B-56. Example netlist in the hierarchical EDIF format (continues).

```
        (portRef &SUM (instanceRef &halfadd_A))))
      (net N00023
       (joined
        (portRef &Y (instanceRef &halfadd_A))
        (portRef &SUM (instanceRef &halfadd_B))))
      (net N00024
       (joined
        (portRef &CARRY (instanceRef &halfadd_A))
        (portRef &I0A (instanceRef &U1))))
      (net &VCC
       (joined
        (portRef &VCC (instanceRef &U1))))
      (net &CARRY_OUT
       (joined
        (portRef &CARRY_OUT)
        (portRef &OA (instanceRef &U1))))
      (net N00027
       (joined
        (portRef &I1A (instanceRef &U1))
        (portRef &CARRY (instanceRef &halfadd_B))))
      (net &GND
       (joined
        (portRef &GND (instanceRef &U1))))
      (net &X
       (joined
        (portRef &X)
        (portRef &X (instanceRef &halfadd_B))))
      (net &Y
       (joined
        (portRef &Y)
        (portRef &Y (instanceRef &halfadd_B))))))))
  (design &EX6
   (cellRef &EX6
    (libraryRef MAIN_LIB))))
```

Figure B-56. Example netlist in the hierarchical EDIF format (continued).

HDUMP (HDUMP.EXE)

The HDUMP.EXE format file is used to produce a hierarchical netlist containing all the information on the schematic sheets. No information is omitted or changed. You can use this netlist format when troubleshooting a design.

SPICE (HSPICE.EXE)

The HSPICE.EXE format file is used to produce hierarchical netlists in the SPICE format. The flat SPICE netlist format file is discussed earlier in this appendix. All of the options that apply to the flat EDIF format apply to the hierarchical version as well. See the section *SPICE (FSPICE.EXE)* for information about SPICE netlist characteristics and configuration options.

HFORM uses HSPICE.EXE to produce hierarchical netlists with subcircuit (.SUBCKT) definitions for sheets in the hierarchy. These subcircuits are called by the X command. Since SPICE does not require the subcircuits to be defined before use, the hierarchy appears in normal form in the netlist with the root sheet at the top of the file.

Examples

The netlist in figure B-57 and the corresponding map file in figure B-58 were created—with no options selected—from the schematics shown in figures B-54 and B-55.

```
* Hierarchical (Complex) SPICE   Revised: June 30, 1994
* OrCAD-07                       Revision: A
* OrCAD
* 9300 SW Nimbus Avenue
* Beaverton, OR  97008
* (503) 671-9500 Sales & Administration
* (503) 671-9400 Technical Support
.OPTIONS ACCT NODE OPTS NOPAGE
.MODEL DMOD D (VJ=0.6)
.MODEL QMOD NPN (BF=80 RB=100 TR=6NS TF=0.3NS)
.DC LIN VINPUT -5 20 0.25
.PLOT DC I(R7)
VCC 10010 0 DC 15V
VINPUT 10013 0 DC 1V
R1 10011 10017 6.8K
R2 10010 10011 47K
R3 0 10017 47K
R4 10013 10014 100K
R5 10013 10018 100K
R6 10010 10012 3.9K
R7 10010 10015 670
Q1 10015 10012 0 QMOD
XCOMPARATOR1 10014 10011 0 10012 10010 EX7B
XCOMPARATOR2 10018 10017 0 10012 10010 EX7B
.SUBCKT EX7B 10023 10020 10027 10024 10019
D6 10021 10020 DMOD
D7 10021 10019 DMOD
D8 10022 10019 DMOD
D9 10022 10023 DMOD
Q13 10025 10019 10019 QMOD
Q15 10026 10019 10019 QMOD
Q11 10024 10019 10027 QMOD
Q10 10019 10026 10027 QMOD
Q12 10026 10025 10027 QMOD
Q14 10025 10025 10027 QMOD
Q16 10019 10020 10027 QMOD
Q17 10019 10023 10027 QMOD
.ENDS
.END
```

Figure B-57. Example netlist in the hierarchical SPICE format.

```
10010 VCC DC 15V (sheet EX7)
10011 VUPPER (sheet EX7)
10013 VINPUT DC 1V (sheet EX7)
    0 GND (sheet EX7)
10017 VLOWER (sheet EX7)
10019 VPLUS (sheet EX7B)
10020 VINPLUS (sheet EX7B)
10023 VINMINUS (sheet EX7B)
10024 VOUT (sheet EX7B)
10027 VMINUS (sheet EX7B)
```

Figure B-58. Example map file in the hierarchical SPICE format.



Interpreting connectivity databases

This appendix covers the subject of connectivity databases by:

- Introducing their form and function
- Describing in detail the format of incremental and linked connectivity databases that are used with OrCAD's tools
- Including a sample .INF file with comments

OrCAD's **Schematic Design Tools, Digital Simulation Tools,** and **PC Board Layout Tools** use a series of databases to organize design information.

One of the databases, the *incremental connectivity database,* organizes information about a design's connectivity. Basic information about this database is contained in *Chapter 10: Creating a netlist.*

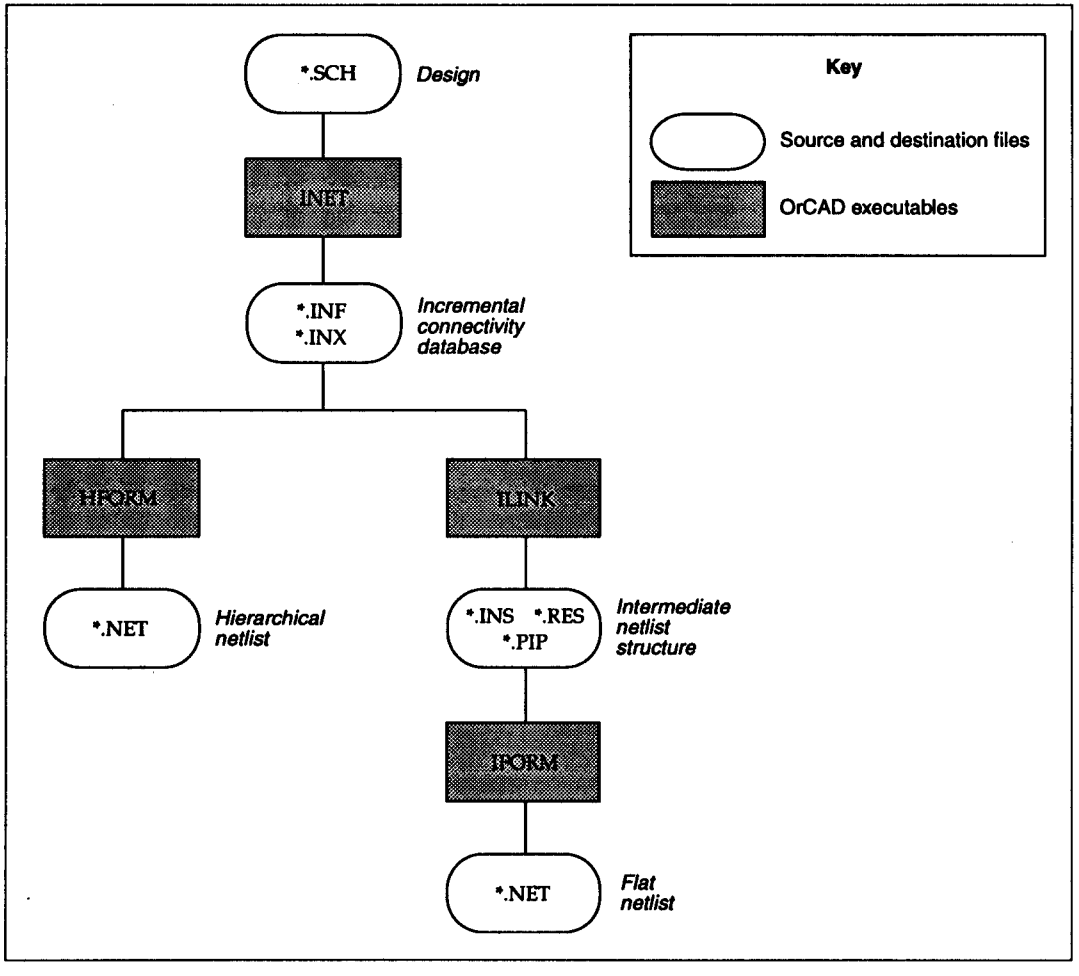
The information in this database is stored in .INF format, which is described in this appendix. With an understanding of this file format, you can develop programs that use connectivity databases as source files or that convert files in other formats to connectivity databases.

Overview of connectivity databases

A connectivity database is a group of one or more files that contain data extracted from schematic worksheets. The figure on the next page shows the relationship of INET, ILINK, IFORM, and HFORM, their source and destination files, and the name of the associated databases.

About the incremental connectivity database

INET gathers data from title blocks, parts, and connections between parts on schematics. INET can also gather specially-marked text and stimulus, trace, and vector data placed on schematics. INET creates one .INF file for each sheet in the design and one .INX file for the entire design.



The netlist process.

This appendix describes the format of .INF files; the .INX file is a text file containing a list of the sheets in the design. INET uses the list to track which sheets need updated .INF files. Together, the .INF files and .INX file are called the *incremental connectivity database.*

This database is the source for both HFORM and ILINK. OrCAD's **Digital Simulation Tools** uses the incremental connectivity database for simulation.

Typographical conventions

This appendix contains several typographical, or notation, conventions that are in addition to or used differently than the rest of this reference guide. The conventions are:

Italic text — This italic font shows a parameter representing specific data that you provide.

. . . — Ellipses shows the continuation of a series.

`monospace` — Monospace font shows characters which are, or might be, in the netlist.

"*parameter*" — Double quotes surrounding a parameter indicate the parameter is a quoted token. The double quotes are part of the syntax.

?parameter? — Question marks surrounding a parameter indicate the parameter can be either a number or a quoted token. The question marks are *not* part of the syntax.

parameter — Parameters without double quotes or questions marks are either numbers or strings. Parameter definitions specify the type of token.

Terminology

This section includes definitions for the various parts of .INF files. You need to know these terms to correctly interpret the syntax of the statements defined in this appendix. See the *Glossary* for other terms.

Token A token is a group of characters preceded and followed by white space. Files in the .INF format contain six kinds of tokens: quoted tokens, strings, delimiters, commands, numbers, and sub-part codes.

White space Files in the .INF format may contain white space between tokens. White space can be a space, a horizontal tab, a carriage return, a line feed, or any combination of these characters. Spaces are also allowed within quoted tokens.

Quoted token A quoted token is a token that begins and ends with double quotes. Quoted tokens may contain any ASCII character except the back quote (`) and may be empty. Two double quotes in a quoted token represent one double quote. This table shows some examples:

Quoted token	*Corresponds to*
"OUTPUT2"	OUTPUT2
"""SEC"" Carry"	"SEC" Carry
" "	*(empty field)*

Examples of quoted tokens.

String A string is an unquoted token that may contain only these characters:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
= - ! # $ % ' , . : ; = ? @ \ ^ _ * { } | ~ /
```

Strings may not contain spaces. For example:

```
Clock
42_22V10
@17#sec?
```

Delimiter A delimiter is an unquoted token comprised of a right or left parentheses. Delimiters occur in pairs to enclose groups of parameters. For example:

```
( R U1 11 ) ( R U2 9 )
```

Command A command is an unquoted token comprised of a back quote immediately followed by a single character from this list:

```
B E F H I J K L P S T V W |
```

The characters must be uppercase. For example:

```
`H
```

Character A character is an unquoted token comprised of a single character. For example:

```
C
```

Number A number is an unquoted token comprised of characters from this list:

```
0 1 2 3 4 5 6 7 8 9
A B C D E F a b c d e f
```

The format supports decimal, hexadecimal, and binary numbers and requires numbers to be in specific formats. For example, the parameter *absolute_identifier* must be an eight-digit hexadecimal number that does not include a suffix. For example:

```
023FB897
```

Sub-part code A sub-part code is an unquoted token that begins and ends with square brackets. The square brackets may enclose only zeros (0) and ones (1). The binary digit 1 indicates a sub-part is used; 0 indicates a sub-part is unused. The number of digits indicates the number of sub-parts. For example:

`[1010]`	Means four sub-parts, first and third used
`[1]`	Means a one-part package
`[0]`	Invalid sub-part code
`[]`	Means a zero-part package

Statement A statement is a group of tokens that begins with a command and ends with either the blank space before the next command or with the end-of-file character. The tokens following the command are parameters. Statements may extend over multiple lines in a file. For example:

```
`W S "INPUT_PORT" 1 ( 0:0 )
```

Parameter A parameter is a single token or a group of tokens delimited by parentheses. The .INF format requires a specific type of token, such as a number or quoted token, for each parameter. For example:

absolute_identifier
(S "*signal_name*" *sheet_number*)

.INF format specification

The following sections list the types of statements that appear in incremental and linked connectivity databases in the order they appear in the .INF files. Each discussion includes syntax and an example; notes contain software-related information.

For ILINK to read .INF files, the statements must appear in the order and frequency shown in the table below. The port and signal statements can be intermingled, as can the trace, vector, and stimulus statements.

Command	*Name*	*Frequency in .INF files*
`H or `F	Header	One
`B	Title block	One
`L	Link	Zero or more
`P	Port	Zero or more
`S	Signal	Zero or more
`E	External	Zero or more
`I	Instance	Zero or more
`J	Joined	Zero or more
`K	Layout	Zero or more
`T	Trace	Zero or more
`V	Vector	Zero or more
`W	Stimulus	Zero or more
`|	Pipe	Zero or more

Order and frequency of statements in .INF files.

H or F Header

The header specifies information about the .INF file and must be the first line in the file. The syntax is:

```
`file_type  format_version  file_name
```

file_type Tells whether the design is flat or hierarchical. *file_type is* one of two characters: `F` for flat or `H` for hierarchical.

△ *NOTE: ILINK accepts references to children in instance commands only from hierarchical .INF files.*

format_version Tells the version of the format. A string.

file_name Tells the name of the .INF file. A string.

Example

```
`F 1.00 NEWALARM
```

B Title block

The title block statement contains information from the title block. Each .INF file contains only one title block command. The title block statement immediately follows the header. The syntax is:

```
`B  "sheet_number"  "total_sheet_number"
"sheet_size"  "date"  "document_number"
"revision_code"  "title"  "organization_name"
"address_line_1"  "address_line_2"  "address_line_3"
"address_line_4"
```

`B Begins parameters describing title block information.

"*sheet_number*" Tells the sheet number field on the title block.

"*total_sheet_number*" Tells the total sheets field on the title block.

"sheet_size" Tells the sheet size field on the title block.

"date" Tells the date field on the title block.

"document_number" Tells the document number field on the title block.

"revision_code" Tells the revision code field on the title block.

"title" Tells the title field on the title block.

"organization_name" Tells the organization field on the title block.

"address_line" Tells one of four address line field on the title block.

Example

```
`B "1" "7" "A" "January 27, 1991" "1860107-001"
"A" "Demonstration Schematic" "OrCAD"
"9300 SW Nimbus Drive" "Beaverton, Oregon 97008"
"(503) 671-9500" " "
```

L Link

Link statements specify the files to include in an intermediate netlist structure or a linked connectivity database. The syntax is:

```
`L netlist_filename
```

`L Begins the link statement.

netlist_filename Tells the name of a file, with the extension omitted, to be linked. An .INF file contains one link statement for each linked file. A string.

△ ***NOTES:*** *ILINK assumes the extensions are .INF.*

INET and ILINK accept link commands only from a root schematic in a flat design. In addition, every schematic in a linked design must be a single sheet.

Example:

```
`L CPU
`L GRAPHICS
```

P Module port

Module port statements define module port types and names. The syntax is:

```
`P module_port_type "module_port_name"
```

`P Begins a module port definition.

module_port_type Tells a module port type from this table:

Character	*Module port type*
I	Input
O	Output
B	Bidirectional, I/O
U	Unspecified
S	Supply

Module port characters and types.

"*module_port_name*" Tells the name of the module port.

△ ***NOTE:*** *The supply port type is global, crossing all hierarchical boundaries.*

Example

```
`P I "OPEN CARRY"
```

S Signal

Signal statements define signal names and the worksheets containing them. The syntax is:

```
`S "signal_name" sheet_number
```

`S — Begins a signal definition.

"signal_name" — Tells the name of the signal.

sheet_number — Equals the number of the worksheet containing the signal. A whole decimal number.

Example

```
`S "TRYIN1" 2
```

E External

External statements specify configured libraries. The .INF file must contain an external statement for every library referenced on the schematic. The syntax is:

```
`E library_name
```

`E — Begins the external statement.

library_name — Tells the name of a library. A string.

Example

```
`E TTL.LIB
```

I Instance

Instance statements declare the instance of objects on a schematic. The *part instance* syntax is for parts and sheetpath parts; the *child instance* syntax is for sheets and sheet parts. This discussion has two sections; the first describes the part instance syntax, and the second describes the child instance syntax.

❖ Part instance

The part instance syntax is:

```
`I R "part_value" library "library_part_name"
absolute_identifier part_reference [sub_part_code]
"part_field_1" . . . "part_field_8" "module_field"
( "pin_name_1" ?pin_number_1? pin_type_1 )
( "pin_name_2" ?pin_number_2? pin_type_2 ) . . .
( "pin_name_n" ?pin_number_n? pin_type_n )
```

`I — Begins a declaration of an instance of a part or child.

R — Means the following parameters describe a part or a sheetpath part.

△ *NOTE: Sheetpath parts may appear in .INF files as either parts or children, depending on the local configuration settings for INET. If the option* ***Descend into sheetpath parts*** *is selected, INET classifies sheetpath parts as children; otherwise, INET classifies sheetpath parts as parts.*

"*part_value*" — Tells the part value field from the schematic.

library — Tells the name, excluding extension, of the library containing the part. A string.

"*library_part_name*" — Tells the name of the part in the library specified by the *library* parameter.

absolute_identifier Tells the eight-digit hexadecimal number based on the date and time the part was first placed on the schematic. The number remains unchanged and is unique.

△ *NOTE: When a part is one of several in a package, every part in the package has a unique absolute identifier. The absolute identifier in the .INF file is the latest one of all the parts in the package.*

part_reference Tells a unique reference designator for the part containing the pin. A string.

[*sub_part_code*] Tells the number of sub-parts in a part and which of those sub-parts is used. A sub-part code. See the section *Terminology* for examples.

"*part_field*" Tells user-supplied information from a part field. All eight part fields must be included, even if they are empty.

△ *NOTE: Parts that are packages may have only one set of part_field parameters. In cases where the sub-parts placed on a schematic have different information in the part fields, INET selects information for each field from the first sub-part it sees containing information in that field. INET looks at the sub-parts in absolute identifiers order, starting with the smallest absolute identifier.*

"*module_field*" Tells package description information consolidated from one or more of the part fields. The configuration of the **Module Value Combine** key field in **Schematic Design Tools 386+** determines the combination.

△ *NOTE: When sub-parts have different information in part fields used to create the module-field parameter, INET uses the information from the first sub-part in the package even if the part fields are empty.*

(Begins a pin description. The instance command must contain one delimited set of pin parameters for each pin.

"pin_name" Tells the name of the pin.

?pin_number? Tells the number of a pin on a part or a sheet net on a sheetpath part. *?pin_number?* is either a whole decimal number or a quoted token, depending on whether the characters are numeric or alphanumeric.

pin_type Tells one of the pin types listed in the table below.

Character	*Pin type*
I	Input
O	Output
B	Bidirectional, I/O
S	Supply, power
P	Passive
T	Three-state
C	Open collector
E	Open emitter

Characters for pin types.

) Ends a parameter describing a pin.

Example

```
`I R "74LS00" TTL "74LS00" 74A3F631 U1 [1000] ""
"" "14PDIP" "" "" "" "" "" "14PDIP" ( "I0_A" 1 I )
( "I0_B" 4 I ) ( "I0_C" 9 I ) ( "I0_D" 12 I )
( "I1_A" 2 I ) ( "I1_B" 5 I ) ( "I1_C" 10 I )
( "I1_D" 13 I ) ( "O_A" 3 O ) ( "O_B" 6 O )
( "O_C" 8 O ) ( "O_D" 11 O ) ( "GND" 7 S )
( "VCC" 14 S )
```

❖ **Child instance** The child instance syntax is:

```
`I C "sheet_file_name" absolute_identifier
"sheet_name" ( "sheet_net_name_1"
sheet_net_type_1 ) . . . ( "sheet_net_name_n"
sheet_net_type_n )
```

`I Begins a declaration of an instance of a part or child.

C Means the next parameters describe a sheet or a sheet part.

△ *NOTE: Sheetpath parts may appear in .INF files as either parts or children, depending on the configura-tion settings for INET. If the option* ***Descend into sheetpath parts*** *is selected, INET classifies sheet-path parts as children; otherwise, INET classifies sheetpath parts as parts.*

"*sheet_file_name*" Tells the complete filename of the worksheet containing the child's logic.

absolute_identifier Tells the absolute identifier, a unique eight-digit hexadecimal number based on the date and time the part was first placed on the schematic. This number remains unchanged.

"*sheet_name*" Tells the schematic name of a child. This name is unique for each instance of a child on a worksheet; if several children reference the same schematic, each instance has the same *sheet_file_name* but a unique *sheet_name.*

(Begins a parameter specifying information about a sheet net. The instance command must contain one delimited set of sheet net parameters for each sheet net.

"*sheet_net_name*" Tells the name of the sheet net.

sheet_net_type Tells one of the sheet net types listed in this table:

Character	*Sheet net type*
I	Input
O	Output
B	Bidirectional, I/O
S	Supply, power
P	Passive
T	Three-state
C	Open collector
E	Open emitter
U	Unspecified

Characters for sheet net types.

) Ends a sheet net description.

Example

```
`I C "SECOND.SCH" 54AF3C22 "SECOND_1" ( "A0" I )
( "A1" B ) ( "A2" U )
```

J Joined

Join statements specify entities made electrically common in a net. An entity is a signal, module port, pin, or sheet net. Join statements can contain any combination of one or more entities. Join statements containing only one entity indicate single-node nets. The syntax is:

`J (*entity_1*) (*entity_2*) . . . (*entity_n*)

`J Begins a list of electrically common entities.

entity Represents one of four types of parameters describing a signal, module port, pin, or sheet net. Each type has its own syntax as described on the next several pages.

Example

```
`J ( R U3 14 I ) ( R U2 "CENTER 1" P ) ( P S "VCC" )
( S "X0" 3 ) ( C "TRYAGAIN" "X" U )
```

❖ **Signal** The syntax of a signal entity is:

```
( S "signal_name" sheet_number )
```

(S Begins a signal description.

"*signal_name*" Tells the name of a signal.

sheet_number Tells the number of the worksheet containing a signal. A whole decimal number.

) Ends a set of parameters describing a signal.

Example

```
( S "TRYIN1" 2 )
```

❖ **Module port** The syntax of a module port entity is:

```
( P module_port_type "module_port_name" )
```

(P Begins a module port description.

module_port_type Tells the type of module port as listed in this table:

Character	*Module port types*
I	Input
O	Output
B	Bidirectional
U	Unspecified
S	Supply, power

Characters for module port types.

"*module_port_name*" Tells the name of the module port.

) Ends a module port description.

Example

```
( P S "VCC" )
```

❖ **Pin** The syntax of a pin entity is:

(R *part_reference ?pin_number? pin_type*)

(R Begins a pin description.

part_reference Tells the unique reference designator for the part containing the pin. A string.

?pin_number? Tells the pin number. *?pin_number?* is either a whole decimal number or a quoted token. When the pin is a zero-part-per-package pin, *pin_number* is the pin name.

pin_type Tells one of the pin types listed in this table:

Character	*Pin type*
I	Input
O	Output
B	Bidirectional, I/O
S	Supply, power
P	Passive
T	Three-state
C	Open collector
E	Open emitter

Characters for pin types.

) Ends a pin description.

Example

```
( R COAX1 "CENTER 1" O ) ( R U2 9 B )
```

❖ **Sheet net** The syntax of a sheet net entity is:

```
( C "sheet_name" "sheet_net_name" sheet_net_type )
```

(C Begins a sheet net description.

"*sheet_name*" Tells the part value of a sheet part or sheetpath part, or the name of a sheet.

"*sheet_net_name*" Tells the name of the sheet net.

sheet_net_type Tells one of the sheet net types listed in the table at the top of the next page.

Character	*Sheet net types*
I	Input
O	Output
B	Bidirectional
S	Supply, power
P	Passive
T	Three-state
C	Open collector
E	Open emitter
U	Unspecified

Characters for sheet net types.

) Ends a sheet net description.

Example

```
( C "TRY" "A0" I )
```

K Layout

Layout statements specify layout directives for OrCAD's **PC Board Tools**. Four types of layout statements are possible: signal, pin, sheet net, and bus. Each is described separately.

❖ Signal

The syntax of a signal layout statement is:

```
`K S "signal_name" sheet_number "directive"
```

`K Begins layout information.

S Begins a signal definition.

"*signal_name*" Tells the name of the signal.

sheet_number Tells the number of the worksheet containing the signal. A whole decimal number.

"directive" Tells the directive for the layout software.

Example

```
`K S "OUTPUT" 4 "WIDTH(.010)"
```

◆ **Pin**

The syntax of a pin layout statement is:

`K R *part_reference ?pin_number? "directive"*

`K Begins parameters describing layout information.

R Means the following three parameters describe a pin or a sheet net. The next section describes the parameters for sheet nets.

part_reference Tells the unique reference designator for the part containing the pin or sheet net. A string.

?pin_number? Tells the pin number. Either a whole decimal number or quoted token.

"directive" Tells the directive for the layout software.

Example

```
`K R U1 11 "STRATEGY(EXTENSIVE)"
```

◆ **Sheet net**

The syntax of a sheet net layout statement is:

`K R *sheet_name "sheet_net_name" "directive"*

`K Begins layout information.

R Means the following parameters describe a sheet net or pin. The previous section describes the parameters for pins.

sheet_name Tells the name of the sheet. A string.

"*sheet_net_name*" Tells the sheet net name.

"*directive*" Tells the directive for the layout software.

Example

```
`K R U1 "ALTERNATE" "STRATEGY(EXTENSIVE)"
```

❖ **Bus** The syntax of a bus layout statement is:

`K B "*bus_name*[*range*]" *sheet_number* "*directive*"

`K Begins layout information.

B Means the following three parameters describe a bus.

"*bus_name*[*range*]" Tells the name of the bus immediately followed by a range index. If the range is a single value, the index is a decimal number. If the range is more than one value, square brackets enclose the index that is comprised of two whole decimal numbers separated by two periods. The smallest value must be listed first. For example:

```
"COUNTER2"
"COUNTER[0..1]"
```

sheet_number Tells the number of the worksheet containing the bus. A whole decimal number.

"*directive*" Tells the directive for the layout software.

Example

```
`K B "AD[0..7]" 4 "PATTERN(TREE)"
```

T Trace

Trace statements specify trace information for OrCAD's **Digital Simulation Tools**. Four types of trace statements are possible: signal, pin, sheet net, and bus. Each is described separately.

❖ Signal

The syntax of a signal trace is:

```
`T S "signal_name" sheet_number
"signal_display_name"
```

`T — Begins trace information.

S — Begins a signal definition.

"*signal_name*" — Tells the name of the signal.

sheet_number — Tells the number of the worksheet containing the signal. A whole decimal number.

"*signal_display_name*" — Tells the trace name to be displayed during simulation.

Example

```
`T S "OUTPUT" 4 "Y Output"
```

❖ Pin

The syntax of a pin trace is:

```
`T R part_reference ?pin_number?
"part_display_name"
```

`T — Begins parameters describing trace information.

R — Means the following three parameters describe a pin or a sheet net. The next section describes the parameters for sheet nets.

part_reference Tells the unique reference designator for the part containing the pin or sheet net. A string.

?pin_number? Tells the pin number. Either a whole decimal number or quoted token.

"*part_display_name*" Tells the trace name to be displayed during simulation.

Example

```
`T R U1 11 "ALTERNATE"
```

❖ **Sheet net** The syntax of a sheet net trace is:

```
`T R sheet_name "sheet_net_name"
"sheet_display_name"
```

`T Begins trace information.

R Means the following parameters describe a sheet net or pin. The previous section describes the parameters for pins.

sheet_name Tells the name of the sheet. A string.

"*sheet_net_name*" Tells the sheet net name.

"*sheet_display_name*" Tells the trace name to be displayed during simulation.

Example

```
`T R U1 "ALTERNATE" "ALTERNATE"
```

❖ **Bus**

The syntax of a bus trace is:

```
`T B "bus_name[range]" sheet_number display_type
"bus_display_name"
```

`T — Begins trace information.

B — Means the following three parameters describe a bus.

"bus_name[range]" — Tells the name of the bus immediately followed by a range index. If the range is a single value, the index is a decimal number. If the range is more than one value, square brackets enclose the index that is comprised of two whole decimal numbers separated by two periods. The smallest value must be listed first. For example:

```
"COUNTER2"
"COUNTER[0..1]"
```

sheet_number — Tells the number of the worksheet containing the bus. A whole decimal number.

display_type — Tells one of the display types listed in the table on the next page.

Character	*Display type*
B	Binary bus
D	Decimal bus
H	Hexadecimal bus
O	Octal bus

Characters for bus types.

△ *NOTE: If no value for display_type is provided, INET assigns hexadecimal.*

"bus_display_name" — Tells the name of the bus to display during simulation.

Example

```
`T B "AD[0..7]" 4 H "ADDRESS"
```

V Vector

Vector statements specify vector information for OrCAD's **Digital Simulation Tools**. Three types of vector specifications are possible: signal, pin, and bus.

◆ Signal

The syntax of a signal vector is:

```
`V  S  "signal_name"  sheet_number  column_number
```

`V — Begins parameters describing vector information.

S — Means the following three parameters describe signals.

"signal_name" — Tells the name of the signal.

sheet_number — Tells the number of the worksheet containing the signal. A whole decimal number.

column_number — Tells the column number in the test vector. A whole decimal number.

Example

```
`V S "OUTPUT" 4 13
```

◆ Pin

The syntax of a pin vector is:

```
`V  R  part_reference  ?pin_number?  column_number
```

`V — Begins parameters describing vector information.

R — Means the following three parameters describe a pin.

part_reference — Tells the unique reference designator for the part containing the pin. A string.

?pin_number? Tells the pin number. Either a quoted token or a whole decimal number

column_number Tells the column number in the text vector. A whole decimal number.

Example

```
`V R U1 3 8
```

❖ **Bus** The syntax of a bus vector is:

`V B "*bus_name*[*range*]" *sheet_number* *first_column_number*

`V Begins vector information.

B Means the following three parameters describe a bus.

"*bus_name*[*range*]" Tells the name of the bus immediately followed by a range index. If the range is a single value, the index is a decimal number. If the range is more than one value, square brackets enclose the index that is comprised of two whole decimal numbers separated by two periods. The smallest value must be listed first.

sheet_number Tells the number of the worksheet containing the bus. A whole decimal number.

first_column_number Tells the starting column number in the test vector. A whole decimal number.

Example

```
`V B "AD[0..7]" 4 14
```

W Stimulus

Stimulus statements specify stimulus information for OrCAD's **Digital Simulation Tools**. Three types of stimulus specifications are possible: signal, pin, and bus. Stimulus statements for signals and pins can contain set and branch parameters, which are described first.

❖ Set parameter

The syntax of a set parameter is:

time : *function*

time Tells the time at which a state function will occur. An unsigned whole number.

function Tells a state function as shown in the table below.

Character	*State function*
0	Set to value 0
1	Set to value 1
U	Set to undefined state
Z	Set to high impedance state
T	Toggle the signal state

Characters for state functions.

Example

```
100:Z
```

❖ Branch parameter

The syntax of a branch parameter is:

time1 : G : *time2*

time1 Tells the start time. An unsigned whole number.

G Represents the GOTO function.

time2 Tells the target time. An unsigned whole number.

△ ***NOTE: Only one branch statement per stimulus is allowed.***

Example

```
0:G:200
```

❖ **Signal** The syntax of a signal stimulus is:

```
`W S "signal_name" sheet_number ( stimuli )
```

`W Begins parameters describing stimulus information.

S Means the following parameters are for signals.

"signal_name" Tells the name of the signal.

△ ***NOTE: Digital Simulation Tools** does not recognize signal names containing periods (.) anywhere in the signal name or colons (:) at the end of the signal name.*

sheet_number Tells the number of the worksheet containing the signal. A whole decimal number.

(Begins a list of stimuli.

stimuli Lists one or more set or branch parameters.

) Ends a list of stimuli.

Example

```
`W S "INPUT" 3 ( 0:0 50:1 100:T 200:G:50 )
```

- **Pin**

  The syntax of a pin stimulus is:

  ```
  `W  R  part_reference  ?pin_number?  (stimuli )
  ```

  `W — Begins a stimulus description.

  R — Means the following parameters describe a pin.

  part_reference — Tells the unique reference designator for the part containing the pin. A string.

  ?pin_number? — Tells the pin number. Either a whole decimal number or quoted token.

  (— Begins a list of stimuli.

  stimuli — Lists one or more set or branch parameters.

) — Ends a list of stimuli.

  Example

  ```
  `W R U1 3 ( 0:1 50:T )
  ```

- **Bus**

  The syntax of a bus stimulus is:

  ```
  `W  B  "bus_name[range]"  sheet_number  ( stimuli )
  ```

  `W — Begins parameters describing stimulus information.

  B — Means the following three parameters describe a bus.

"*bus_name*[*range*]" Tells the name of the bus immediately followed by a range index. If the range is a single value, the index is a decimal number. If the range is more than one value, square brackets enclose the index that is comprised of two whole decimal numbers separated by two periods. The smallest value must be listed first.

△ ***NOTE: Digital Simulation Tools** does not recognize bus names containing periods (.) anywhere in the bus name or colons (:) at the end of the bus name. You can include periods in the range index.*

sheet_number Tells the number of the worksheet containing the bus. A whole decimal number.

(Begins a list of stimuli.

stimuli Lists stimuli information. Version 1.00 of the .INF format does not define a syntax for bus stimuli, so the stimuli can be any valid string.

△ ***NOTE: Digital Simulation Tools** simulates only signals and parts.*

) Ends a list of stimuli.

Example

```
`W B INPUTBUS[1..4] 3 ( 0:1 50:T )
```

Pipe statements

Pipe statements contain text extracted from the schematic. For the text to be extracted, each line must begin with a pipe (|) symbol. Pipe text may contain any type of information; each program that uses the .INF file scans the pipe statements for keywords and uses those it recognizes. OrCAD tools recognize the keywords PLD, VSTModel, SPICE, and others. An .INF file may contain more than one section of pipe statements. Pipe statements appear at the end of the file. The syntax is:

```
`|
"|keyword"
"|information_line_1"
"|information_line_2"
```

`|` Begins a section of pipe text.

"|*keyword*" Lists a command word or string for a specific program or any text.

"|*information_line*" Lists text.

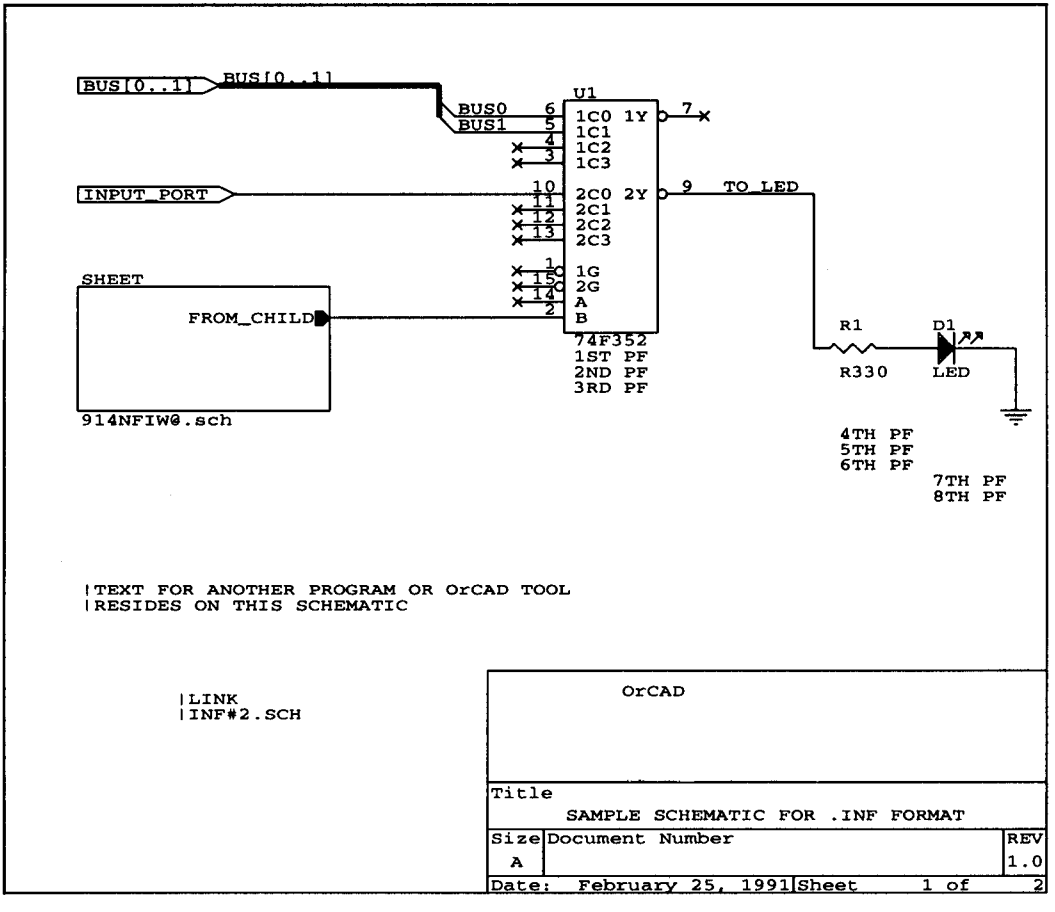
△ ***NOTE:*** *Each section of pipe statements on a schematic created with OrCAD's* ***Schematic Design Tools 386+*** *should begin in a different column.*

Example

```
`|
"|PLDX1 in:(A1, A2, B1, B2),"
"| io:(Y1, Y2)"
"| Y1 = A1 & B1"
"| Y2 = A2 # B2"
```

Sample .INF file

The sample .INF file in this section was created from the schematic drawing below. The schematic shows a variety of objects and how they appear in an .INF file, but is not a working circuit.



The schematic TEST_INF.SCH.

Comments	Statements
TEST_INF.SCH is a hierarchy Title block information	`H 1.00 TEST_INF `B "1" "2" "A" " February 19, 1991" "" "1.0" "SAMPLE SCHEMATIC FOR .inf FORMAT" "OrCAD" "9300 SW NIMBUS DRIVE" "BEAVERTON, OREGON 97008" "U.S.A." "(503)671-9500"
Link statement, for example Module ports on the parent	`L inf#2 `P S "VCC" `P I "INPUT_PORT" `P S "GND" `P I "BUS0" `P I "BUS1"
Signal on the parent Configured libraries	`S "TO_LED" 1 `E TTL.LIB `E DEVICE.LIB
Instance of an LED from DEVICE.LIB	`I R "LED" DEVICE.LIB "LED" 7B0738E9 D1 [] "" "" "" "" "" "" "7TH PF" "8TH PF" "" "ANODE" ("ANODE" P) ("CATHODE" "CATHODE" P)
Instance of a resistor from DEVICE.LIB	`I R "R330" DEVICE.LIB "R" 7B0738E8 R1 [] "" "" "" "4TH PF" "5TH PF" "6TH PF" "" "" "" ("1" "1" P) ("2" "2" P)
Instance of a 4-to-1 multiplexer from TTL.LIB	`I R "74F352" TTL.LIB "74F352" 7B07BEA3 U1 [1] "1ST PF" "2ND PF" "3RD PF" "" "" "" "" "" "" ("1C0" 6 I) ("1C1" 5 I) ("1C2" 4 I) ("1C3" 3 I) ("2C0" 10 I) ("2C1" 11 I) ("2C2" 12 I) ("2C3" 13 I) ("1G" 1 I) ("2G" 15 I) ("A" 14 I) ("B" 2 I) ("1Y" 7 O) ("2Y" 9 O) ("VCC" 16 S) ("GND" 8 S)
Instance of a child—a sheet Join statements listing the contents of each net on the parent	`I C "914NFIW@.sch" 7B5427A5 "SHEET" ("FROM_CHILD" O ) `J (P S "VCC") (R U1 16 S) `J ( P I "BUS0" ) ( R U1 6 I ) `J (P I "BUS1") (R U1 5 I) `J ( P I "INPUT_PORT" ) ( R U1 10 I ) `J (R U1 9 O) (S "TO_LED" 1) (R R1 "1" P) `J ( R U1 1 I ) `J (C "SHEET" "FROM_CHILD" O) (R U1 2 I) `J ( R D1 "CATHODE" P ) ( P S "GND" ) ( R U1 8 S ) `J (R R1 "2" P) (R D1 "ANODE" P)
Trace, vector, layout, and stimulus information from a bus	`T B "BUS[0..1]" 1 H "TRACENAME" `V B "BUS[0..1]" 1 5 `K B "BUS[0..1]" 1 "PATTERN(TREE)" `W B "BUS[0..1]" 1 (2:200)
Stimulus, vector, trace, and layout information from a signal	`W S "INPUT_PORT" 1 ( 0:0 0:G:0 0:G:10 ) `V S "INPUT_PORT" 1 10 `T S "INPUT_PORT" 1 "INPUT_PORT" `K S "INPUT_PORT" 1 "WIDTH(.010)"

Sample .INF file (continues).

Comments	Statements
Vector, stimulus, trace, and layout information from a part	`V R U1 2 6 `W R U1 2 (STIM_CHILD) `T R U1 2 "TRACE_CHILD" `K R U1 2 "STRATEGY(EXTENSIVE)"
Pipe text from the worksheet	`| "|TEXT FOR ANOTHER PROGRAM OR OrCAD TOOL" "|RESIDES ON THIS SCHEMATIC"

Sample .INF file (continued).

Creating a custom netlist format

This appendix describes the process of creating a custom netlist format to use with OrCAD's **Schematic Design Tools 386+**. You specify a netlist format for flat netlists in a netlist format file or for hierarchical netlists in a hierarchical netlist format file.

IFORM uses the netlist format file and an intermediate netlist structure created by ILINK to create a flat netlist in the format you define.

HFORM uses the hierarchical netlist format file and an incremental connectivity database created by INET to create a hierarchical netlist in the format you define. Figure D-1 shows both processes.

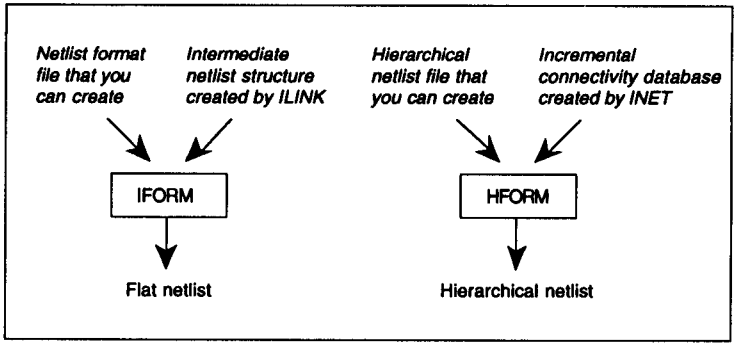


Figure D-1. Source and destination files for IFORM and HFORM.

To create a custom netlist format file, you need Microsoft C version 7.x or Borland C version 3.x.

About netlist formats

At the end of OrCAD's netlist process, IFORM and HFORM use netlist format files and OrCAD databases to create flat and hierarchical netlists in formats you select from local configuration screens. OrCAD supplies over thirty format files for the most common file formats; you can create your own format files for uncommon or situation-specific formats. For more information about the netlist process, see chapters 10, 11, and 12 in this guide.

The two types of netlist formats are:

- A flat format where all ports and signals are resolved across the entire design.
- A hierarchical format where all instances of sheets and subsheets, called children, remain intact and are used to reference subnets.

Either type of netlist format can be part- or net-oriented.

Flat formats

The flat format gives all parts unique references and all nodes unique names. Although this view accurately reflects the design, it removes evidence of the structure of the design.

Hierarchical formats

The hierarchical format retains the hierarchical format of the original design, complete with all subsheets and non-unique references and nodes.

Part and net orientations

Both flat and hierarchical netlist formats can focus on parts or on nets. IFORM and HFORM manage separate data structures for parts, nets, and children; you use different functions to access the data structures. A subset of the pre-defined functions work only when creating part-oriented netlists; the remaining functions work for both net- and part-oriented netlists. For specifics, see the section *Data structures.*

OrCAD-supplied formats

OrCAD supplies format files for over thirty formats with **Schematic Design Tools 386+**. The formats and sample netlists are in *Appendix B: Netlist formats.*

Customer-contributed formats

In addition to OrCAD-supplied netlist formats, customers who create netlist format files can post them on OrCAD's bulletin board so other customers can use them.

Directory structure

Table D-1 shows the **Schematic Design Tools 386+** directory structure that is required to compile a netlist.

Path	*File*	*Description*
\ORCADESP\SDT\NETFORMS	F*fmtName*.EXE, H*fmtName*.EXE	Compiled netlist format files
\ORCADESP\SDT\NETFORMS\BRLNDOBJ	*.OBJ	Object files for use with Borland C compiler
\ORCADESP\SDT\NETFORMS\OBJ	*.OBJ	Object files for use with Microsoft C compiler
\ORCADESP\SDT\NETFORMS\NETSRCF (flat source files)	*fmtName*.C	Flat netlist format source files
	FORMAT.H	Header file called from source files
\ORCADESP\SDT\NETFORMS\NETSRCH (hierarchical source files)	*fmtName*.C	Hierarchical netlist format source files
	FORMAT.H	Header file called from source files
\ORCADEXE	MKNETFMT.BAT	Batch file that calls the compiler and linker

Table D-1. Directory structure for netlist format files.

Compiled netlist format files

The directory \ORCADESP\SDT\NETFORMS contains the compiled netlist format files that are used with IFORM and HFORM. Flat format source filenames begin with "F," while hierarchical format source files begin with "H." The prefix is followed by the name of the format and the extension .EXE (for example, FWIRELIS.EXE).

Object files

OrCAD supplies five object files for use with Microsoft C V7.x, placed in the \ORCADESP\SDT\NETFORMS\OBJ directory, that are necessary to all netlist formats. They must be linked when you compile and link your format file. These files are:

- CONTEXT.OBJ
- RESIDENT.OBJ
- STUBS.OBJ
- TSR.OBJ (for flat formats) or HTSR.OBJ (for hierarchical formats)

Borland C V3.x uses a different set of the same five object files. The Borland versions are placed in the directory \ORCADESP\SDT\NETFORMS\BRLNDOBJ.

△ *NOTE: The MKNETFMT batch file automatically uses the appropriate set of object files for the compiler you specify.*

Flat source files

The \ORCADESP\SDT\NETFORMS\NETSRCF directory contains the source files for flat netlist formats. Each netlist format has a separate *source file,* written in the C programming language. This file has the name of the format and a .C extension. WIRELIST.C is an example.

This directory also contains a *header file* called FORMAT.H. This file is called from each source file using an `include` statement.

Hierarchical source files

The \ORCADESP\SDT\NETFORMS\NETSRCH directory contains the source files for hierarchical netlist formats. Each netlist format has a separate *source file,* written in the C programming language. This file has the name of the format and a .C extension.

This directory also contains a *header file* called FORMAT.H, which is called from each source file using an `include` statement.

How to create a new format

This section describes the process of creating a new netlist format.

1. **Know what you want the destination file to look like.**

   You should have sample netlists in the desired format, and, if possible, a specification for the format.

2. **Check for one or more formats that are similar to what you want.**

   Look through *Appendix B: Netlist formats* and find one or two netlist formats that are similar to the one you want. If your netlist format is in two parts like the Spice or Vectron formats, find the one that is closest to your format. In the following steps, you will use the corresponding format files as templates for the new format.

3. **Examine the .C files and compare them to the netlists they create.**

   Examine the .C files for the formats you identified as being similar to yours. Determine how IFORM or HFORM creates the netlist, looking at how nets and pins are written.

4. **Identify what changes you need to make.**

   Identify the changes you need to make in the .C file. When necessary, refer to the syntax and reference information later in this appendix.

5. **Make the changes.**

   Edit a copy of the .C file that is nearest to what you want using **Edit File** or another text editor. Be sure to name it something different.

6. **Compile and link the format file.**

   To compile and link a flat or hierarchical netlist format file, use OrCAD's DOS batch program called MKNETFMT.BAT. A full description of the program appears in this appendix in *Compiling and linking a format file.*

7. **Test the format file.**

   Create a netlist for a sample schematic using the **Create Netlist** or **Create Hierarchical Netlist** tool on the **Schematic Design Tools** work screen.

   Chances are, IFORM or HFORM will report error messages. A list of the messages and explanations is at the end of this appendix.

8. **Fine tune the format file.**

   Go back to step 3 and identify what worked and what didn't, then redo steps 4 through 7 until IFORM or HFORM creates a netlist that meets your specification.

Compiling and linking a format file

The batch program MKNETFMT.BAT is used to compile and link netlist format files.

The MKNETFMT.BAT batch program assumes that your system has the OrCAD-recommended directory structure, but it can accept nonstandard directory structures if you specify them on the command line.

MKNETFMT.BAT command syntax

Enter **MKNETFMT** or **MKNETFMT ?** to display information about how the batch program works. This information also displays if you run MKNETFMT with invalid parameters.

MKNETFMT *compiler type fmtName* [*srcPath* [*objPath* [*cmpPath* [*libPath*]]]]

compiler — *compiler* must be M or B to specify the Microsoft C V7.x compiler or the Borland C V3.x compiler. The M or B can be either uppercase or lowercase.

type — *type* must be either F or H, to specify whether the netlist format is flat or hierarchical. The F or H can be either uppercase or lowercase.

fmtName — *fmtName* is the name of the netlist format source file, excluding the dot and extension (.C). The netlist format source file must be named *fmtName*.C for MKNETFMT to work.

srcPath *srcPath* is the path to the directory containing the netlist format source file specified in *fmtName*. The path can be relative or absolute. *srcPath* is optional.

If you don't specify it, MKNETFMT looks for *fmtName* in C:\ORCADESP\SDT\NETFORMS\NETSRCF if the format is flat (*type* is F) or in C:\ORCADESP\SDT\NETFORMS\NETSRCH if the format is hierarchical (*type* is H).

objPath *objPath* is the path to the directory containing the required object files. The path can be relative or absolute. *objPath* is optional.

If you don't specify it, MKNETFMT looks for the object files in C:\ORCADESP\SDT\NETFORMS\OBJ for the Microsoft C compiler (*compiler* is M) or in C:\ORCADESP\SDT\NETFORMS\BRLNDOBJ for the Borland C compiler (*compiler* is B).

If you specify *objPath*, you must also specify *srcPath*.

cmpPath *cmpPath* is the path to the directory where the compiled netlist format file is to be stored. The path can be relative or absolute. *cmpPath* is optional.

If you don't specify it, MKNETFMT places the compiled netlist format file in C:\ORCADESP\SDT\NETFORMS.

If you specify *cmpPath*, you must also specify *srcPath* and *objPath*.

libPath *libPath* is the path to the directory containing the Borland library files. The path can be relative or absolute. This parameter is checked only for the Borland C compiler (*compiler* is B). *libPath* is optional.

If you do not specify it, MKNETFMT looks for the Borland library files in \BORLAND\LIB.

If you specify *libPath*, you must also specify *srcPath*, *objPath*, and *cmpPath*.

More about MKNETFMT.BAT

The MKNETFMT.BAT batch program assumes that your system has the OrCAD-recommended directory structure, but it can accept nonstandard pathnames if you specify them on the command line.

When specifying pathnames, do not include a backslash (\) at the end of the path.

Compiler and linker messages are written to the file MKNETFMT.OUT.

Requirements

For the Microsoft C V7.x compiler, MKNETFMT verifies that the compiler is available by testing the presence of certain environment variables added to your AUTOEXEC.BAT file when you installed the Microsoft C V7.x compiler (see step 4 in the next section).

For the Microsoft C V7.x compiler, MKNETFMT also checks that you have set an additional environment variable, BINB, pointing to the directory that contains the Microsoft C V7.x linker, typically C:\C700\BINB (see step 4 in the next section).

How MKNETFMT.BAT works

If you run it without parameters or with missing or incorrect parameters, MKNETFMT displays a usage summary. If **?** is the first parameter, MKNETFMT displays the information in this appendix.

When you run MKNETFMT with the correct parameters, these are the steps it performs:

1. MKNETFMT checks that the *compiler* parameter is "M" or "B" (lowercase or uppercase). If it is not, MKNETFMT displays a message and the usage summary.
2. MKNETFMT checks that the *type* parameter is "F" or "H" (lowercase or uppercase). If it is not, MKNETFMT displays a message and the usage summary.
3. MKNETFMT checks that you specified a netlist format (*fmtName* is not empty). If you did not, MKNETFMT displays a message and the usage summary.

4. If *compiler* is "M," MKNETFMT verifies that the Microsoft C v7.0 environment variables—LIB, INIT, INCLUDE and BINB—are set. If any is not set, MKNETFMT displays an error message and explanation.

5. If you specify an alternate pathname in *srcPath*, MKNETFMT looks there for *fmtName*.C. Note that MKNETFMT also places the object file that it creates in the directory specified in *srcPath*.

   If you don't specify *srcPath*, MKNETFMT looks for *fmtName*.C in one of these directories:

   - If *type* is "F," MKNETFMT looks in \ORCADESP\SDT\NETFORMS\NETSRCF.
   - If *type* is "H," MKNETFMT looks in \ORCADESP\SDT\NETFORMS\NETSRCH.

   If MKNETFMT doesn't find *fmtName*.C, it displays an error message and the usage summary.

6. If you specify an alternate pathname in *objPath*, MKNETFMT looks there for these four object files:

   - TSR.OBJ (if *type* is "F") or HTSR.OBJ (if *type* is "H")
   - CONTEXT.OBJ
   - RESIDENT.OBJ
   - STUBS.OBJ

   If you don't specify *objPath*, MKNETFMT looks for these four files in one of these directories:

   - If *compiler* is "M," MKNETFMT looks in \ORCADESP\SDT\NETFORMS\OBJ.
   - If *compiler* is "B," MKNETFMT looks in \ORCADESP\SDT\NETFORMS\BRLNDOBJ.

   If MKNETFMT doesn't find all four of these files, it displays an error message and the usage summary.

7. If you specify an alternate pathname in *cmpPath*, MKNETFMT stores the executable (.EXE) and map (.MAP) files it creates there.

   If you don't specify *cmpPath*, MKNETFMT stores these files in \ORCADESP\SDT\NETFORMS.

8. If *compiler* is "B":

   If you specify an alternate pathname in *libPath*, MKNETFMT looks there for the Borland small-memory-model object file, C0S.OBJ.

   If you don't specify *libPath*, MKNETFMT looks for C0S.OBJ in \BORLANDC\LIB.

   If it does not find C0S.OBJ, MKNETFMT displays an error message and the usage summary.

9. MKNETFMT deletes any previous version of the file MKNETFMT.OUT that it finds in the current directory (the directory from which you invoke MKNETFMT).

10. MKNETFMT deletes any previous version of the file *fmtName*.OBJ that it finds in the current directory.

11. MKNETFMT invokes the specified compiler with the appropriate command:

    - If *compiler* is "M":

      `cl /c /Zp1 /W3 /DLINT_ARGS` *srcPath*`\`*fmtName*`.C`

    - If *compiler* is "B":

      `bcc -c` *srcPath*`\`*fmtName*`.C`

    MKNETFMT redirects any compiler messages to the file MKNETFMT.OUT, which it creates in the current directory.

    If the compiler returns a 0, indicating successful completion, MKNETFMT copies the resulting object file to the source directory.

    If the compiler returns any other value, indicating a problem, MKNETFMT displays an error message and terminates (see step 17).

12. MKNETFMT deletes any previous version of the file TMP.LNK that it finds in the current directory.

13. MKNETFMT creates the appropriate linker response file, TMP.LNK, for the format type (flat or hierarchical) you specified in *type* and for the compiler (Microsoft or Borland) you specified in *compiler*. TMP.LNK contains the path (relative or absolute) and filenames of the netlist format source file, the required object files, and the executable and map files.

    If the compiler parameter is "B," TMP.LNK also contains the names of Borland library files and an additional command expected by the Borland linker.

    If there is a problem creating TMP.LNK, MKNETFMT displays an error message and terminates (see step 17).

14. MKNETFMT deletes any previous versions of the files *fmtName*.MAP and *fmtName*.EXE that it finds in the current directory.

15. MKNETFMT invokes the specified linker with the appropriate command:

    - If *compiler* is "M":

      valueOfBINB`\link /st:4096 /MA /CODEVIEW @TMP.LNK`

    - If *compiler* is "B":

      `tlink -L`*libPath* `@TMP.LNK`

    MKNETFMT redirects any linker messages to the file MKNETFMT.OUT.

    If the linker returns a 0, indicating successful completion, MKNETFMT copies the resulting compiled netlist format file and map file to the appropriate directory (\ORCADESP\SDT\NETFORMS, unless you specify an alternate directory in *cmpPath*).

    If the linker returns any other value, indicating a problem, MKNETFMT displays an error message and terminates (see step 17).

16. If it hasn't detected any problems, MKNETFMT displays a message and terminates (see step 17).

17. Before it terminates for any reason, MKNETFMT deletes the temporary files (TMP.*) it creates in the current directory and clears any variables it has set (except BINB).

If you have problems . . .

If you're unable to successfully compile a netlist format file, check the things listed here.

Environment variables

Make sure your environment variables match those below. Microsoft C version 7.x sets these variables by default.

```
SET LIB=C:\c700\LIB
SET INCLUDE=C:\c700\INCLUDE
SET HELPFILES=C:\c700\HELP\*.HLP
SET INIT=C:\c700\INIT
```

Compiler and linker messages

MKNETFMT.BAT places messages from the compiler and linker in a file called MKNETFMT.OUT. Check MKNETFMT.OUT for messages after your run MKNETFMT.

Combined library file

Check that the combined library SLIBCE.LIB exists in the directory defined by the LIB environment variable. This library is created when you select coprocessor emulation and the small memory model during the Microsoft C V7.x installation process.

About format files

Netlist format files must contain specific constructs that IFORM and HFORM can use. This section describes language guidelines, functions, and symbols. Following sections, including the reference portion of this appendix, describe individual elements of format files.

File names

Netlist format files must have .EXE as an extension. So the ESP design environment can group and display format files correctly in list boxes, it is recommended that flat netlist format files have names beginning with the letter "F" and hierarchical netlist format files have names that begin with the letter "H." MKNETFMT creates names that conform to these guidelines automatically.

Language

The format files are written in Microsoft C V7.x or Borland C V3.x and support all of the standard features of the C programming language.

Functions

IFORM and HFORM recognize both C-language functions and OrCAD-defined functions; you can define others. The reference part of this appendix describes both types of functions. For information about defining your own functions or symbols, see a C-language reference manual such as *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie.

Standard symbols

IFORM and HFORM recognize a number of predefined standard symbols. For a list of the standard symbols, see the section *Symbol reference.*

User-defined symbols

IFORM and HFORM recognize up to thirty-six user-defined symbols. You refer to each symbol by the name you specify in its definition. Symbol names may be up to thirty-two characters long and may contain any ASCII character except the double quote ("). User-defined symbols can be set, reset, deleted, and printed.

Two user symbols are already defined for you. See the discussions for **ExitType** and **LocalSignal** for more information about these symbols.

Flat format

This section includes pseudocode of the loop that IFORM executes to perform netlist formatting. Pseudocode is an English-like description of what happens in a program. For information about required functions in .C files, see the section *Required functions*.

Following the pseudocode is a "Hello World" example that, like many first examples of computer languages, simply writes the words "Hello World" to a file.

```
FormatNetlist()  /* Pseudocode for creating a flat netlist */
{
   InitializeFormatModule();

   /* either output a header directly for a net based netlist
    * or create the part data base for a part based netlist    */
   WriteHeader();

   NetNumber = 0;
   read the TypeCode from the Resolved File

   while not at the end of the Resolved File
   {
       increment NetNumber

       NotConnected = FALSE;

       switch (TypeCode)
       {
           'L' :      read the SignalType
                      read the SignalNameString from resolved file
           | 'P' :    SignalType = Byte('u');
                      read the SignalNameString from resolved file
           | 'U' :    NotConnected = TRUE;
           | 'N' :    /* do nothing */
       }

       HandleNodeName(); /* part- or net-oriented */

       FirstTime = TRUE;
       NetCount = 0;
```

Pseudocode of the format loop IFORM executes (continues).

```
        read the ReferenceString
        while ReferenceString is not empty
        {
            read PinNumberString from resolved file
            read PinNameString from resolved file
            read PinIndex from resolved file
            read PartName from resolved file
            read ModuleName from resolved file
            read PinType from resolved file

            /* Either output to a file directly for net-oriented netlists
               or add the net to an internal data structure for part-oriented
               netlists                */
            WriteNet();

            FirstTime = FALSE;
            read the ReferenceString
        }
        WriteNetEnding(); /* can be empty if part-oriented */
        read the TypeCode from the Resolved File
    }

        ProcessFieldStrings();

    /* output the trailer for net-oriented netlists or output the netlist
       itself for a part-oriented netlist     */
    WriteNetListEnd();

}   /* end of FormatNetlist */
```

Pseudocode of the format loop IFORM executes (continued).

```
/* Hello World example for a flat netlist */

Initialize()
{
/* add a symbol called Formatter */
AddSymbol( "Formatter" );

/* set formatter to a value */
SetSymbol( Formatter, "Hello World" );
}

WriteHeader()
{
/* now output the symbol to the first file */
WriteSymbol( 1, Formatter );

/* write a newline to the first file */
WriteCrLf( 1 );
}

/* the following required functions are not used but are still required
   in the file, so leave them empty */
WriteNet()
{
}

HandleNodeName()
{
}
WriteNetEnding()
{
}
ProcessFieldStrings()
{
}
WriteNetListEnd()
{
}
```

"Hello World" example for a flat netlist.

Hierarchical format

This section includes pseudocode of the loop that HFORM executes to perform netlist formatting. For information about required functions in .C files, see the section *Required functions.*

Following the pseudocode is a "Hello World" example.

```
FormatNetList()  /* Pseudocode for creating a hierarchical netlist */
{
   InitializeFormatModule();
   InitializeParser();

   /* call the user initialization function  */
   Initialize();

   NetNumber := 0;
   depth := 0;

   scan for all parts in the design and build a small version
   of the libraries containing only the parts used in the design

   add all files found in the design to the file stack

   set the worksheet number to zero
   WHILE not empty file stack
   {
       increment the worksheet number

       open the top file on the stack
       pop the file stack

       PreFile;

       parse the current .inf file

       PostFile();
   }

   /* initialize the pipe file and call the user post processing function,
      finally, cleanup and leave    */
   InitializePipeFile();
   PostProcess();
   CleanupFormatModule;

   exit with the indicated code

}  /* end of FormatNetlist */
```

Pseudocode of the format loop HFORM executes.

```
/* Hello World example for a hierarchical netlist */

Initialize()
{
/* add a symbol called Formatter */
AddSymbol( "Formatter" );

/* set formatter to a value */
SetSymbol( Formatter, "Hello World" );

/* now output the symbol to the first file */
WriteSymbol( 1, Formatter );

/* write a newline to the first file */
WriteCrLf( 1 );
}

/* the following required functions are not used but are still required
 * in the file, so leave them empty
 */
PreFile()
{
}
PostFile()
{
}
PostProcess()
{
}
```

"Hello World" example for a hierarchical netlist.

Required functions

IFORM and HFORM require netlist format files to include the following functions in any order:

Required functions for IFORM

HandleNodeName();
Initialize();
ProcessFieldStrings();
WriteHeader();
WriteNet();
WriteNetEnding();
WriteNetListEnd();

Required functions for HFORM

Initialize();
PreFile();
PostFile();
PostProcess();

Data functions

IFORM and HFORM access data from data structures and from instance files. This section describes data structures and instance files.

Data structures

IFORM and HFORM use four data structures to manage data extracted from the intermediate netlist structure and the incremental connectivity database, respectively. The data structures contain net, part, and child data.

IFORM accesses the part, part-oriented, and net-oriented data structures. HFORM accesses the part, child, and net-oriented data structures for each file on the file stack.

The following sections describe four types of data structures: part, child, net-oriented, and part-oriented, and lists the functions you use with each type.

Part data structure

The part data structure contains information about all parts; it contains the symbols **ReferenceString, PartName, ModuleName, TimeStamp, PinNameString, PinNumberString,** and **PinType.**

HFORM accesses and traverses the part data structure when you use these functions in .C files :

```
FirstPin();
int AccessPart( user_symbol );
int FirstPart();
int NextPart();
int NextPin();
int PinCount();
int SetPartIndex( integer_constant );
int SetPartIndex( variable );
int SetSignal();
```

IFORM, however, accesses part data information from part-oriented data structures as described on the next page.

Child data structure

The child data structure, accessed only by HFORM, contains information about the children for the current worksheet in a hierarchical design. The child data structure is comprised of children and the sheet nets on each child; each child and sheet net is represented by standard symbols. The standard symbols are **PinNameString, PinNumberString, PinType, ModuleName, PartName, ReferenceString,** and **TimeStamp.**

The functions that access child data structures are:

```
FirstChildPin();
int AccessChild( user_symbol );
int ChildPinCount();
int FirstChild();
int NextChild();
int NextChildPin();
```

Net-oriented data structure

Net-oriented databases contain all the net information in flat designs and all the net information for the current worksheet in hierarchical designs. The data is arranged by nets, with information about nodes and parts subordinate to the net information. Both IFORM and HFORM can access the net-oriented database.

HFORM can also access data in a net-oriented data structure from a part perspective.

The net-oriented data structure is the default data structure. Until you load another data structure, a net from the net-oriented data structure is current. Net-oriented data structures are comprised of nets and nodes comprised of standard symbols.

IFORM and HFORM access net-oriented data structures differently. IFORM scrolls through data structure automatically and you have no control over what is current unless you traverse the part-oriented data structure and use the function **SetSignal()** to synchronize the two data structures.

HFORM, however, accesses information from net data structures in a variety of ways depending on which functions you use to traverse the data structure. When you use such a function to change the current net, the first node on that net also becomes current.

The functions for net-oriented data structures you use in .C format files for HFORM are:

```
FirstNet();
FirstNode();
int NextNet();
int NextNode();
int PreviousNode();
```

Part-oriented data structure

Part-oriented data structures contain net information about flat designs organized so IFORM can access data from a part perspective.

You can include functions that direct IFORM to create and use a part-oriented data structure. When creating a part-oriented data structure, include these functions in the format file:

- Include the function **RecordNode()** in the **WriteNet()** function.
- Include the function **AddSignalName()** in the **HandleNodeName()** function.
- Include the function **CreatePartDatabase()** in the **WriteHeader()** function.

These functions access part-oriented data structures:

```
int EndNode();
int GetIndex( part_symbol );
SetFirst( part_symbol );
int SetIndexByRef( user_symbol );
int SetNext( part_symbol );
int SetPrevious( part_symbol );
int SetToIndex(part_symbol, variable );
```

With one exception, you can use all other functions with part-oriented netlists: using **SortByNumber** scrambles the data, making the part-oriented data structure invalid.

Instance files

The instance file (*.INS), part of the intermediate netlist structure, contains a list of all parts and instances in a design. Each instance file is comprised of instances that, in turn, are comprised of standard symbols. At any given time, one instance is current.

IFORM accesses the instance file automatically; HFORM can create an instance file and access it if you use the function **MakeInstanceFile()**. Once HFORM creates an instance file, it can use all the instance file functions.

These functions access instance files:

LoadFieldString(string_symbol);
int LoadFirstPin();
int LoadInstance();
int LoadPin();
MakeInstanceFile();
int NextAccessType();
int NextInstance();
RewindInstanceFile();
SetAccessType(string_symbol);
SortByNumber();

Traversal functions

Traversal functions allow you to control how HFORM traverses a design. HFORM recognizes two variations of a traversal function: **SetTraversal(string_constant)** and **SetTraversal(user_symbol)**.

Pipe file functions

IFORM can access the pipe file (*.PIP) of the intermediate netlist structure for pipe information, and HFORM can access the incremental connectivity database for pipe information.

When using HFORM, the .PIP file can only be accessed during the **PostProcess()** function.

The pipe file functions are:

int AccessKeyWord(string_constant);
int AccessKeyWord(standard_symbol);
int AccessKeyWord(user_symbol);
int FirstPipe();
int IsKeyWord();
int NextKeyWord();
int NextPipe();

For more about these functions, see the section *Function reference.* See also the function **PostProcess()**.

General functions

General functions are OrCAD-defined functions. Some access data structures and some do not. The general functions are:

AddSymbol(string_constant);
ClearSymbolicStrings();
int CompareSymbol(symbol, symbol);
ConcatFile (file_index, file_index);
CopySymbol(symbol, user_symbol);
ExceptionsFor(string_constant, user_symbol);
int FindSymbolChar(variable, user_symbol);
int GetStandardSymbol(standard_symbol);
int GetSymbolChar(variable, user_symbol);
MakeLocalSignal(string_constant);
int PackString(integer_constant, integer_constant, user_symbol, user_symbol);
int PackString(variable, variable, user_symbol, user_symbol);
PadSpaces(user_symbol, integer_constant);
PadSpaces(user_symbol, variable);
PutSymbolChar(variable, integer_constant, user_symbol);
PutSymbolChar(variable, variable, user_symbol);
SetAccessType(user_symbol);
SetCharSet(string_constant);
SetNumberWidth(integer_constant);
SetNumberWidth(variable);
SetPinMap(integer_constant, string_constant);
SetPinMap(variable, string_constant);
SetSymbol(user_symbol, string_constant);
StripPath(string_symbol);
int SwitchIsSet(string_constant);
int SymbolInCharSet(user_symbol);
int SymbolLength(symbol);
ToUpper(user_symbol);
WriteCrLf(file_index);
WriteInteger(file_index, integer_constant);
WriteInteger(file_indes, variable)
WriteMap(file_index, integer_constant);
WriteMap(file_index, variable);
WriteString(file_index, string_constant);
WriteSymbol(file_index, user_symbol);
WriteStdSymbol(file_index, standard_symbol);

For more about these functions and symbols, see the reference portion of this appendix.

Switches

You can define switches, called "options" in the ESP design environment, that provide options for creating custom netlists. Then when you configure IFORM and HFORM, you select a file format and select options for that format. The first comment in the .C file names switches and defines descriptions that display on the configuration screen. Then you use **SwitchIsSet()** to find out if IFORM or HFORM was called with the switch set.

The switch names can be uppercase or lowercase. However, IFORM and HFORM convert all lowercase switch names to uppercase.

△ *NOTE: Do not use these switches that IFORM or HFORM already use: /B, /Q, and /Z.*

Standard symbol reference

Symbols are names associated with an array of characters that is accessed via the symbol name. Functions use symbolic information as the main method for accessing data. Standard symbols may be accessed and printed but may not be changed.

This section lists standard symbols of four classifications:

- **Symbols for HFORM** that you use only for hierarchical format files.
- **Title block symbols** that contain information from worksheet title blocks.
- **User symbols** that act as standard symbols.
- **Standard symbols** that contain information about parts, pins, nets, and children, text from the pipe file, and indexes.

AddressLine1–AddressLine4	*title block symbol*	String. The address lines from the worksheet.
DateString	*title block symbol*	String. The date string from the worksheet.
DocumentNumber	*title block symbol*	String. The document number from the worksheet.
ExitType	*user symbol*	String. This is a reserved symbol in the user symbol space since this symbol may be set. The symbol is read after the formatting loop is complete. ❖ If the symbol contains "W," then there were warnings found during the format process. ❖ If the symbol contains "E," then errors were found during the format process. IFORM and HFORM each have the **Ignore warnings** option on the **Local Configuration** screen. The table on the next page shows the results of combinations of **ExitType** values and option settings.

ExitType contains	*Ignore warnings option*	*Result*
W	ON	Exits
W	OFF	Exits reporting a warning
E	ON or OFF	Exits reporting an error
nothing	ON or OFF	Exits

Results of combinations of ExitType and Ignore warnings options.

Use this symbol when you define error or warning messages for a format file. You do not need to use the **AddSymbol()** function for **ExitType**; it is already defined.

The function **ClearSymbolicStrings()** does not clear **ExitType**.

FieldString1–FieldString8 *standard symbol* String. Field strings for an instance of a part. A **LoadInstance()** call must precede use of this symbol.

FileName *standard symbol* String. The file that is being processed.

KeyWord *standard symbol* String. Contains the current keyword from the pipe file.

LibraryNameString *standard symbol* String. Contains the library name for the current part.

LocalSignal *user symbol* A user symbol containing a local signal name constructed from a specified **string_constant** and the standard symbols **SignalNameString** and **SheetNumber**. The function that creates this user symbol is **MakeLocalSignal()**. You do not need to use **AddSymbol()** to define this symbol.

LookupNameString	*standard symbol*	String. Contains the name used to look up the current part in the part library indicated by **LibraryNameString**.
ModuleName	*standard symbol*	String. This is the module name obtained from the module value field.
NetCode	*symbol for HFORM*	Character. Contains the current net node type. It may be one of: L Label node—a node labeled on a worksheet P Port node—a node connected to a module port on a worksheet S Power node—a node connected to power or ground N Node—an unlabeled node connected to something other than a module port, power, or ground U Unconnected node—a node unconnected to anything on a worksheet This is only loaded when a net is made current in HFORM. It is the **NetType** obtained from scanning the nodes on the net as though a "link" had been run for the net.
NetNameString	*symbol for HFORM*	String. The name of the current net. Since a net for HFORM is composed of all the nodes, including multiple ports or labels if present, the **NetNameString** is built as it would have been if the net had been "linked". This is a unique name and is loaded if the **NetType** is "L," "P," or "S."
NetNumber	*standard symbol*	Unsigned integer. The current net number. IFORM and HFORM assign unique net numbers to all nodes in a net.

NetType	*symbol for HFORM*	Character. Contains the signal type. This may be one of: 0 Unspecified 1 Output 2 Input 3 Bidirectional 255 Not Used (0FFH) HFORM loads **NetType** only when a net is made current.
Organization	*title block symbol*	String. The organization name string from the worksheet.
PartIndex	*standard symbol*	Unsigned integer. The index corresponds to the parts-per-package number of the current part. This value represents the annotated suffix, with 0 = "A", 1 = "B", and so on. The value is loaded when a signal is made current. For a part-oriented netlist, **PartIndex** is loaded with the index into the part list when a part is made current. Since parts are assumed to have unique references, the **PartIndex** is unique for each part in the package.
PartName	*standard symbol*	String. This is the part name obtained from the part value field.
PinIndex	*standard symbol*	Unsigned integer. If **PinNameString** is "OUT" or "FBK", then the index is set to 0FFFFH to accommodate Altera's ADF netlist format. Otherwise, the integer is encoded with the most significant four bits containing which device in the package this pin belongs to and the lower twelve bits containing the pin definition offset. The value of interest is 0FFFFH or not 0FFFFH.

PinNameString *standard symbol*
String. Contains the pin name string for the current instance of a node.

PinNumberString *standard symbol*
String. Contains the string representation of the pin number for the current instance of a node.

PinType *standard symbol*
Character. Contains one of these types for pins, ports and signals, or sheets:

Pin
0 Input
1 BiDirectional
2 Output
3 Open Collector
4 Passive
5 Hi-Z
6 Open Emitter
7 Power

Port & signal
0 Unspecified
1 Output
2 Input
3 BiDirectional

Sheet
0 Unspecified
1 Output
2 Input
3 BiDirectional
4 Open collector
5 Passive
6 Hi-Z
7 Open emitter
8 Power

PipeLine *standard symbol*
String. This contains the current line in the pipe file.

ReferenceString *standard symbol*
String. Contains the reference designator for either the instance of a part or of a node.

Revision	*title block symbol*	String. The revision number from the worksheet.
SheetNumber	*title block symbol*	Unsigned integer. Contains the current worksheet number. In IFORM, if the current **TypeCode** is "L", **SheetNumber** reflects the number of the sheet containing the signal. If **TypeCode** is any other value, **SheetNumber** may not be accurate because it is updated only if the current signal net has a label. In HFORM, **SheetNumber** is always correct.
SheetSize	*title block symbol*	Character. Contains the size of the worksheet.
SignalNameString	*standard symbol*	String. May contain a signal name. If **TypeCode** is "L", "P", or "S", then **SignalNameString** contains the signal name. The **SignalNameString** is empty otherwise.
SignalType	*standard symbol*	Character. Contains the signal type. The signal types are: 0 Unspecified 1 Output 2 Input 3 Bidirectional
TimeStamp	*standard symbol*	String. Contains the hexadecimal time stamp for the part. The hexadecimal time stamp has no suffix to indicate the number is hexadecimal.
TitleString	*title block symbol*	String. Contains the title string from the worksheet.

TotalSheets	*title block symbol*	Unsigned integer. Contains the total number of sheets as shown on the worksheet.
TypeCode	*standard symbol*	Character. Contains the current net node type. The net node types are: L Label node S Power node P Port node N Node containing no symbolic name U Unconnected node

Type definition reference

Type definitions are constants or variables that you supply for functions in the format file. This section lists the type definitions you use in format files.

access_constant	*type definition for general functions*	A subset of **string_constant**, this must be one of these four access fields:
		`"PARTVALUE"` `"LOOKUP"`
		`"LIBRARY"` `"REFERENCE"`
		See **SetAccessType()** in the section *Function reference.*

file_index	*type definition for general functions*	An integer constant that indicates to which output file IFORM or HFORM writes. Index 0 is reserved for output to the screen while index 1 and index 2 refer to the files specified on the **Local Configuration** screen. Index 3 refers a scratch file that is deleted after IFORM or HFORM are finished. The default is 0. A file index may also be a variable.

format_constant	*type definition*	One of these three values: ADF A standard known as both the Altera Design Format and Intel's Advanced Design File. MOD OrCAD's **Programmable Logic Device Modeling Tools.** EDIF Electronic Design Interchange Format See **ExceptionsFor()** in the section *Function reference.*
integer_constant	*type definition for general functions*	A whole number in the range –32,678 to 32,767.

part_symbol *type definition for part-oriented functions*

A subset of **user_symbol** for use with functions for part-oriented netlists. The following tables are the source of all the standard symbols when writing format files for creating part-oriented netlists.

SIGNALS	The table of all signal names and types. The net contains one signal name and type for each connected node.
PARTS	The table of all parts found in the instance file. Each part entry is comprised of the standard symbols **ReferenceString**, **PartName**, and **ModuleName**. A one-to-one correspondence exists between entries in the PARTS and NETS tables.
NETS	The table of all nets. The table entry points to all the pins, or nodes, on a part that are electrically connected to another node.
NODES	The list of all nodes on a given net. Each pin on a part may occur in a net. The electrical node is identified by the SIGNAL name.
ALL	The list of all signals, parts, nets, and nodes.

This symbol is one of the thirty-six you can define. However, to access a part-oriented database, **part_symbol** must contain one of the five table names listed above. To have access to these symbols, you must use AddSymbol.

standard_symbol *type definition for general functions*

One of the standard symbols listed in the section *Symbol reference.*

string_constant	*type definition for general functions*	A string enclosed in quotation marks.
symbol	*type definition for general functions*	Either a **standard_symbol** or a **user_symbol**.
traversal_constant	*type definition for general functions*	A subset of **string_constant**, this must be either `"ROOT"` or `"LEAF"`. See **SetTraversal()** in the section *Function reference.*
user_symbol	*type definition for general functions*	A user-defined symbol.
variable	*type definition for general functions*	An integer defined with an **int** statement in the range –32,768 to 32,767.

Function reference

This portion of the appendix lists all the standard functions that IFORM and HFORM recognize in format files. The functions appear in alphabetical order, disregarding any **int** prefixes. The descriptions include a function classification and definition.

int AccessChild (user_symbol);

child data structure function

Uses the input **user_symbol** containing a valid **PartName** to access the child that matches the name. The function returns 1 on success or 0 otherwise.

int AccessKeyWord (string_constant);

int AccessKeyWord (standard_symbol);

int AccessKeyWord (user_symbol);

pipe file functions

Uses the input **string_constant** or **symbol** containing a pipe symbol keyword to make the next line in the file containing that keyword current. **KeyWord** and **PipeLine** are both loaded. The function returns 1 on success or 0 otherwise. If the function returns 0, then no action has been taken—the position in the file is the same as before the call and **KeyWord** and **PipeLine** are the same as before the call.

int AccessPart (user_symbol);

part data structure function

Makes the part that matches the input user symbol—assumed to contain a valid **ReferenceString**—current. The function loads the associated standard symbols and returns 1 on success or 0 if the input reference was not found. Use this function only when creating format files for hierarchical netlists.

AddSignalName();

part-oriented netlist function

Adds the current **SignalNameString** and **SignalType** to the signal table. If the **SignalNameString** is empty, the **NetNumber** is used instead. This function is required in formats for part-oriented netlists and should be used within the **HandleNodeName()** required function.

AddSymbol (string_constant);

general function

Adds the symbol name indicated by **string_constant** to the list of available symbols. You must use this function to create new user symbols before you use the symbols.

int ChildPinCount();	*child data structure function*	Returns the total number of sheet pins on the current child.
ClearSymbolic Strings();	*general function*	Clears the list of available user symbols. All user symbols and their contents are erased. See also **AddSymbol()** in the section *Function reference.*
int CompareSymbol (symbol, symbol);	*general function*	Returns 0 if the values represented by the two input **symbols** are the same and an non-zero value if they either are not the same or are not both strings or both numbers.
ConcatFile (file_index_1, file_index_2);	*general function*	Places the contents of the file specified by **file_index_2** at the end of the file specified by **file_index_1**. The type definitions **file_index_1** and **file_index_2** must be either 1, 2, or 3. See **file_index**.
CopySymbol (symbol, user_symbol);	*general function*	Copies the contents of the symbolic string indicated by **symbol** into the indicated **user_symbol**.
CreatePart DataBase();	*part-oriented netlist function*	Uses the instance file to create a part-oriented database IFORM can use. You call this function within the required function **WriteHeader()**.
int EndNode();	*part-oriented netlist function*	Returns 1 if the current node in the NETS table is **EndNode** and 0 otherwise. For information about the NETS table, see **part_symbol** in the section *Type definition reference.*

ExceptionsFor (format_constant, user_symbol);	*general function*	Modifies **user_symbol** according to the format specified in **format_constant**. IFORM sets **user_symbol** to the appropriate characters for three different standards: ADF, MOD, and EDIF. For example, to have IFORM parse symbols and convert them to symbols that meet EDIF standards, write this: `ExceptionsFor("EDIF", TempStr )`
int FindSymbolChar (variable, user_symbol);	*general function*	Finds the indicated **variable** in the string represented by **user_symbol**. The index into the string of the first occurrence of the character is returned if the character was found, otherwise 0FFFFH (–1) is returned. For these and other functions that access specified locations in a string, the first character in the string is at location zero.
int FirstChild();	*child data structure function*	Makes the first child in a child data structure current. The first sheet net, **ReferenceString, TimeStamp, PartName, PinNameString,** and **PinType** are also made current. The function returns 1 on success or 0 if the first child is empty. An empty first child means that the incremental netlist file is a leaf node containing no children.
FirstChildPin();	*child data structure function*	Makes the first sheet net current and loads the associated standard symbols.
FirstNet();	*net-oriented data structure function*	Makes the first net the current net. The first node on the net is made current and the associated standard symbols are loaded. Use this function only when creating format files for hierarchical netlists.

FirstNode();	*net-oriented data structure function*	Makes the first node on the current net current and loads the associated standard symbols. Use this function only when creating format files for hierarchical netlists.
int FirstPart();	*part data structure function*	Makes the first part current and loads the associated standard symbols. The first pin of the part also becomes current. The function returns 1 on success or 0 if there are no parts in this incremental netlist file. A file containing no parts occurs when the schematic is composed only of children or when the schematic is empty. In both cases, **FirstPart()** and **FirstChild()** return 0. Use this function only when creating format files for hierarchical netlists.
FirstPin();	*part data structure function*	Makes the first pin of the current part current. The associated standard symbols are loaded. Use this function only when creating format files for hierarchical netlists.
int FirstPipe();	*pipe file function*	Sets the seek pointer to either zero or the start of the pipe file. This function returns 1 on success or 0 otherwise. On success, the **KeyWord** and **PipeLine** are loaded.
int getche();	*C-language function*	Returns a character from the keyboard.
int GetIndex (part_symbol);	*part-oriented netlist function*	Returns the current index for the table indicated by **part_symbol**. For example: `GetIndex(SIGNALS);` For a discussion of the tables, see the type definition **part_symbol**.

int getnum(); — *C-language function*

Returns an integer that is input from the keyboard. This function does not check the input to make sure it is valid.

int GetStandardSymbol (standard_symbol); — *general function*

Returns the value associated with **standard_symbol**. If **standard_symbol** is a string, then 0 is returned. Otherwise the integer or the **char** value cast as an integer is returned.

int GetSymbolChar (variable, user_symbol); — *general function*

Returns the character at the location **variable** in the string associated with **user_symbol**. For example, if **user_symbol** is **Part** and contains `PLD22V10`, this function returns `V`:

```
GetSymbolChar(5, Part);
```

For this and other functions that access specified locations in a string, the first character in the string is at location zero.

HandleNodeName (); — *required function for IFORM*

Allows access to several symbols, depending on which type of net it encounters. When IFORM reads the *.RES file and encounters a net, it calls this function. This table lists node types and the symbols **HandleNodeName()** can access:

Net type	*Symbols accessed by HandleNodeName()*
Label (L)	**TypeCode, NetNumber, SignalType, SignalNameString**
Port (P)	**TypeCode, NetNumber, SignalType, SignalNameString**
Power (S)	**TypeCode, NetNumber, SignalType, SignalNameString**
Node (N)	**TypeCode, NetNumber**
Unconnected (U)	**TypeCode, NetNumber**

Initialize();	*required function for IFORM and HFORM*	IFORM and HFORM call this function first. Here you can initialize symbols and variables, define character sets, write headers, and set number widths.
int IsKeyWord();	*pipe file function*	Returns 1 if the current standard symbol **PipeLine** contains a keyword and 0 if it does not.
int LastFile();	*child data structure function*	Returns 1 if the current file is the last file in the file stack, otherwise returns 0.
LoadFieldString (user_symbol);	*instance file function*	Uses the input **user_symbol** that represents a part reference and loads associated standard symbols from the instance file. The function returns 1 on success or 0 on end of file. **LoadFieldString** is restricted to part-oriented netlists only.
int LoadFirstPin();	*instance file function*	Loads the first pin in the current instance. This function must follow a **LoadInstance()** call. The function loads **PinNameString, PinNumberString,** and **PinType.** The function returns 1 on success or 0 otherwise.
int LoadInstance();	*instance file function*	Loads the standard symbols from the current entry in the instance file. This function returns 1 if the values were loaded correctly and 0 on end of file. This function may be called in a conditional statement.
int LoadPin();	*instance file function*	Loads the next pin in the current instance. A **LoadInstance()** call must precede this function. The function loads **PinNameString, PinNumberString, PinType, LibraryNameString,** and **LookupNameString.** The function returns 1 on success or 0 otherwise.

MakeInstanceFile();	*instance file function*	Creates an instance file for HFORM. This function must be called in the **Initialize()** loop. This function is valid only in *.C format files.
MakeLocalSignal (string_constant);	*general function*	Uses the current standard symbols to construct a local signal name and places it in the user symbol **LocalSignal**. **LocalSignal** is built by concatenating **SignalNameString**, the input **string_constant**, and the **SheetNumber**. The current number width—set with **SetNumberWidth()**—determines the width of LocalSignal. **LocalSignal** is one of the thirty-six user symbols.
int NextAccessType ();	*instance file function*	Makes the next instance that contains an access field different than the current access field current. The function returns 1 on success or 0 otherwise. When you use this function with **SetAccessType()**, **SetAccessType()** should occur first. For a list of the access fields, see **access_constant** in the section *Type definition reference*.
int NextChild();	*child data structure function*	Makes the next child current and loads the associated standard symbols. This function returns 1 on success or 0 at end of list. This function must follow the **FirstChild()** function.
int NextChildPin();	*child data structure function*	Makes the next sheet net current. This function returns 1 on success or 0 at end of list. This function must follow the **FirstChildPin()** function.

int NextInstance();	*instance file function*	Makes the next instance that contains an access field matching the current access field current. If the current access field is not set, the next instance becomes current. When you use this function with the **SetAccessType()** function, **SetAccessType()** should precede **NextInstance()**. For a list of the access fields, see **access_constant** in the section *Type definition reference.*
int NextKeyWord ();	*pipe file function*	Accesses the next keyword in the pipe file. This function returns 1 on success or 0 otherwise. If the function returns 0, then no action is taken. If the function returns 1, then **KeyWord** contains the next keyword and **PipeLine** contains the line in which the keyword appears.
int NextNet();	*net-oriented data structure function*	Makes the next net current and makes the associated standard symbols current. This function returns 1 on success or 0 at end of list. The **FirstNet()** function must precede **NextNet()**. Use this function only when creating format files for hierarchical netlists.
int NextNode();	*net-oriented data structure function*	Makes the next node current and loads the associated standard symbols. This function returns 1 on success or 0 at end of list. The **FirstNode()** function must precede **NextNode()**. Use this function only when creating format files for hierarchical netlists.
int NextPart();	*part data structure function*	Makes the next part current and loads the associated standard symbols. This function returns 1 on success or 0 at end of list. The **FirstPart()** function must precede **NextPart()**. Use this function only when creating format files for hierarchical netlists.

int NextPin(); — *part data structure function*

Makes the next pin on the current part current and loads the associated standard symbols. This function returns 1 on success or 0 on end of list. The **FirstPin()** function must precede **NextPin()**. Use this function only when creating format files for hierarchical netlists.

int NextPipe(); — *pipe file function*

Makes the next pipe line current and loads **PipeLine** and **KeyWord**. This function returns 1 on success or 0 otherwise. The **FirstPipe()** function must precede **NextPipe()**.

int PackString (integer_constant, integer_constant, user_symbol, user_symbol);

int PackString (variable, variable, user_symbol, user_symbol); — *general functions*

Deletes a substring of the first **user_symbol**, starting with the character in the space specified by the first **integer_constant** and ending with the second **integer_constant**. Then it places the deleted substring into the second **user_symbol**. This function returns 1 on success or 0 otherwise. For example:

```
PackString(6, 4, SigNameStr, Str);
```

The second variation of the function behaves the same except the spaces are defined by variables. For example:

```
PackString(pos, len, SigNameStr, Str);
```

For these and other functions that access specified locations in a string, the first character in the string is at location zero.

PadSpaces (user_symbol, integer_constant); **PadSpaces (user_symbol, variable);**	*general functions*	Sets the length of **user_symbol** to the length specified by **integer_constant** or **variable**. The function either pads the string with spaces or truncates it.
int PinCount();	*part data structure function*	Returns the number of pins on the current part. Use this function only when creating format files for hierarchical netlists.
PostFile();	*required function for HFORM*	HFORM calls this function after parsing the current worksheet. All the data structures are loaded and available for use.
PostProcess();	*required function for HFORM*	HFORM calls this function after reading all the worksheets in the design. The pipe file functions are valid during this function call. Pipes are considered a post processing function.
PreFile();	*required function for HFORM*	HFORM calls this function before processing the next worksheet in the design. HFORM has not parsed the worksheet yet, but the filename and worksheet number are set. The function is called once for every file in the design.
int PreviousNode();	*net-oriented data structure function*	Makes the previous node current and loads the associated standard symbols. This function returns 1 on success or 0 on start of list. Used with **NextNode()**, this function allows bi-directional movement through net-oriented data structures. Use this function only when creating format files for hierarchical netlists.

int print (string_constant); **int print (standard_symbol);** **int print (user_symbol);** **int print(variable);**	*C-language functions*	Writes the contents of **string_constant, standard_symbol, user_symbol,** or **variable** to the screen. The function returns 0.
ProcessFieldStrings ();	*required function for IFORM*	IFORM calls this function.
int putch(variable)	*C-language function*	Writes **variable** to the screen. This function returns the value of **variable.**
int puts (string_constant); **int puts (user_symbol);** **int puts (standard_symbol)**	*C-language functions*	Prints a string or symbol and a newline to the screen. As listed at the beginning of this appendix, arrays are not supported, so the argument is either **string_constant, user_symbol,** or **standard_symbol.** This function returns 0.

PutSymbolChar (variable, integer_constant, user_symbol); **PutSymbolChar (variable, variable, user_symbol);**	*general functions*	Inserts the character in **integer_constant** into the string specified by **user_symbol** at the location specified by **variable.** The second form inserts the binary value contained in the second **variable** into the string specified by **user_symbol** at the location specified by the first variable. For these and other functions that access specified locations in a string, the first character in the string is at location zero.
RecordNode();	*part-oriented netlist function*	Adds the current net node to the node table. If **PinNumberString** is not a number, this function sets the pin number in the table to zero. On access, **PinNumberString** is set to **PinNameString.** The function is only valid within the **WriteNet()** required function.
RewindInstanceFile ();	*instance file function*	Makes the first instance in the instance file current and loads the associated standard symbols. A **MakeInstanceFile()** call must occur before this function is called.

SetAccessType (access_constant);
SetAccessType (user_symbol);

instance file function

Sets the access type to the value specified by **access_constant** or **user_symbol**, where the value of **user_symbol** must evaluate to one of four access fields.

You use this function with the function **int NextAccessType()**. This table shows the symbols accessed for each value:

Value of access_constant or user_symbol	*Symbol accessed*
PARTVALUE	**PartName**
LIBRARY	**LibraryNameString**
LOOKUP	**LookupNameString**
REFERENCE	**ReferenceString**

Symbols accessed.

For example:

```
SetAccessType("LIBRARY");
```

For information about the symbols listed in the table, see the section *Symbol reference.*

SetCharSet (string_constant);

general function

Sets the valid character set to be the input string. Using **SymbolInCharSet()**, you can check symbols against the list of valid characters.

SetFirst (part_symbol);

part-oriented netlist function

Makes the first entry in the table indicated by **part_symbol** current. For example:

```
SetFirst(SIGNALS);
```

int SetIndexByRef (user_symbol);

part-oriented netlist function

Accesses the PARTS table, making the part with the reference specified by **user_symbol** current. This function returns 1 on success or 0 otherwise.

int SetNext (part_symbol);	*part-oriented netlist function*	Makes the next index in the table indicated by **part_symbol** current. This function returns 1 on success or 0 otherwise. This function appears in loop constructs. For example: `SetNext(ALL);`
SetNumberWidth (integer_constant); **SetNumberWidth (variable);**	*general functions*	Sets the width of fields containing output number to the value specified by **integer_constant** or **variable.** Numbers are padded with zeros on the left up to this width when IFORM or HFORM write them to a file or to the screen. Except for the symbol **LocalSignal,** this function has no effect on how numbers are stored. Values of zero or one result in no padding. The default value is one. Wider numbers are not truncated.
int SetPartIndex (integer_constant); **int SetPartIndex (variable);**	*part data structure functions*	Makes the part specified by **integer_constant** or **variable** current and loads the associated standard symbols. This function returns 1 on success or 0 if the index is out of range. Use this function only when creating format files for hierarchical netlists.
SetPinMap (integer_constant, string_constant); **SetPinMap (variable, string_constant);**	*general functions*	Sets the index for the input pin map specified by **integer_constant** or **variable** to the input **string_constant.** This function accesses a pin map that allows strings to be associated with integer values. You can set up to 16 pin maps. You use these functions with **WriteMap().**

int SetPrevious (part_symbol); — *part-oriented netlist function*

Makes the previous index in the table indicated by **part_symbol** current. This function returns 1 on success or 0 otherwise. Loop constructs may include this function. For example:

```
SetPrevious(PARTS);
```

Use this function only when creating format files for hierarchical netlists.

int SetSignal(); — *part data structure function*

Finds the matching pin in the net data structure and sets the **SignalNameString, SignalType,** and **TypeCode** for the net associated with the pin. This function returns 1 if the pin exists or 0 if the pin is unconnected. You use this function to synchronize the part and net data structures. Use this function only when creating format files for hierarchical netlists.

SetSymbol (user_symbol, string_constant); — *general function*

Assigns the contents of **string_constant** to **user_symbol.**

int SetToIndex (part_symbol, variable); — *part-oriented netlist function*

Sets the index of the table specified by **part_symbol** to the input **variable.** Returns 1 on success or 0 otherwise. For example:

```
SetToIndex(NETS, 5);
```

SetTraversal (traversal_constant);

SetTraversal (user_symbol); — *traversal functions*

Sets the way HFORM traverses a design. A top down traversal is a depth first traversal that starts at the root. A bottom up traversal is a breadth first traversal that starts at the leaf nodes. The input value **traversal_constant** is either `"ROOT"` or `"LEAF"`, and **user_symbol** should evaluate to either `ROOT` or `LEAF`. For example:

```
SetTraversal("ROOT");
```

SortByNumber();	*instance file function*	Sorts only the instance file by reference number. As a side effect, the seek pointer is set to start of file, so if you have made a pass through the instance file using **LoadInstance**, a **SortByNumber()** call allows you to make another pass through the file using **LoadInstance**. Using this function on a part-oriented data structure causes the data structure to become invalid. See **CreatePartDatabase()**.
StripPath (string_symbol);	*general function*	Strips the path from the filename contained in **string_symbol**, and places the resulting filename in **string_symbol**.
int SwitchIsSet (string_constant);	*general function*	Checks to see if the switch specified by **string_constant** is set. This function returns 1 if the switch is set or 0 if not. The first character in the input **string_constant** is the only relevant character. The character must be an alphabetic character. Include this function during initialization.
int SymbolInCharSet (user_symbol);	*general function*	Checks the contents of the string associated with **user_symbol** against the current character set. The function returns 1 if all characters in the symbol are in the character set or 0 otherwise. The function **SetCharSet()** defines the character set.
int SymbolLength (symbol);	*general function*	Returns the length of the string represented by **symbol**.
ToUpper (user_symbol);	*general function*	Changes any lowercase alphabetic characters in **user_symbol** to uppercase.
WriteCrLf (file_index);	*general function*	Writes a newline to the file specified by **file_index**.

WriteHeader();	*required function for IFORM*	IFORM calls this function. It can do two things, depending on the functions you include: output a header or build a part database using the function **CreatePartDatabase()**.
WriteInteger (file_index, integer_constant); **WriteInteger (file_index, variable);**	*general functions*	Writes the specified **integer_constant** or **variable** to the file given in **file_index**.
WriteMap (file_index, integer_constant); **WriteMap (file_index, variable);**	*general functions*	Writes the pin map string at **integer_constant** or **variable** to the file given by **file_index**. See also the **SetPinMap()** functions.
WriteNet();	*required function for IFORM*	IFORM calls this function, setting **PinIndex, ReferenceString, PinNumberString, PartName, ModuleName,** and **PinType**. Then, depending on the functions you include, this function should either write the current net for net-oriented netlists or add the net to a part-oriented netlist by calling the function **RecordNet()**.
WriteNetEnding();	*required function for IFORM*	IFORM calls this function when it reaches the end of a net. Then this function outputs whatever you specified to terminate nets.
WriteNetListEnd();	*required function for IFORM*	IFORM calls this function after reading all the nets. The function should either output whatever you specify to terminate a net-oriented netlist or traverse a part-oriented data structure and output the netlist that was recorded by calls to **WriteNet()**.

WriteStdSymbol (file_index, standard_symbol);	*general function*	Writes **standard_symbol** to the file specified by **file_index.**
WriteString (file_index, string_constant);	*general function*	Writes the **string_constant** to the file indicated by **file_index.**
WriteSymbol (file_index, user_symbol);	*general function*	Writes **user_symbol** to the file specified by **file_index.**

Error and warning messages

When you run IFORM or HFORM with a netlist format that contains one or more errors, the process stops at the first error and an error message and line number display. When you run IFORM and HFORM under the ESP design environment, the file #ESP_OUT.TXT contains the error messages.

For information about warnings, errors, and the `Ignore warnings` option, see the entry **ExitType** in the reference portion of this appendix.

The errors messages include:

`Cannot SortByNumber after CreatePartDatabase`. Calling the function **SortByNumber()** scrambles the data in the part-oriented data structure.

`Closing comment, no opening comment`. A closing comment (`*/`) was found but no opening comment (`/*`) was found.

`Closing quote expected`. A string constant was found but not correctly terminated.

`Equal sign expected`. An assignment was attempted without the equal sign (=).

`Function expected`. A function definition or call was expected.

`Index out of range`. The index is out of range. Either a file index or a pin map index is not valid.

`Nested comments`. Two opening comments (`/*`) were found before a closing comment (`*/`) was found.

`No expression present`. An expression was expected but none was found.

`Not a string`. The value required should be a string constant.

`Not a variable`. An attempt to set or read a value that is not a variable.

`Opening comment, no closing comment`. An opening comment (`/*`) and end of file were found before a closing comment (`*/`) was encountered.

`Parameter error`. A general error for internal functions when a parameter type did not match.

`Parentheses expected`. A parentheses—either "(" or ")"—was expected but not found.

`RETURN without call`. A return statement was found before a function call was made.

`Semicolon expected`. The start of another statement was found before a terminating semicolon (;) was found.

`Symbol not found`. The intended symbol was not found in either the standard symbol table or the user symbol table.

`Symbol table overflow`. An attempt to add too many user defined symbols. Use fewer symbols.

`Syntax error`. General syntax error.

`Too many local variables`. The local variable table has overflowed. Try using fewer variables or try to make some of them global.

`Too many nested function calls`. The internal function call stack was exceeded. Reduce the call nesting depth.

`Type specifier expected`. A variable declaration needs to have a type (**int** or **char**) supplied.

`Unbalanced braces`. Found a statement requiring a matched set of curly braces ({ and }) before encountering a closing curly brace (}) such as in a function definition.

`Unbalanced parentheses`. An end of a statement was found before the final closing parentheses (")") was found.

`While expected`. Do-while statement ended the block, but a while was not found.



Plotter information

This appendix contains additional information you may need to set up, configure, and use your plotter with **Schematic Design Tools 386+**. The sections in this appendix are:

- Plotter cable wiring
- Plotter problems
- General plotter tips
- Plot language for HP plotters and printers
- HI plotters
- Calcomp plotters
- Notes on plotter and printer drivers

Plotter cable wiring

The **Plot Schematic** tool uses DOS BIOS calls to communicate with the serial port. It does not talk to the hardware directly. This is to ensure compatibility with all PCs and compatibles.

For this reason, additional wires other than TXD and RXD must be connected to implement hardware handshake.

Figure E-1 is a wiring diagram showing the connections necessary to connect a 25-pin connector to a plotter. Figure E-2 is a wiring diagram showing the connections necessary to connect a 9-pin connector to a plotter. Figure E-3 is a wiring diagram showing the connections necessary for a 25-pin connector to an IOline plotter.

Since this cable connects the TXD and RXD lines, it also works with software that communicates to the hardware directly.

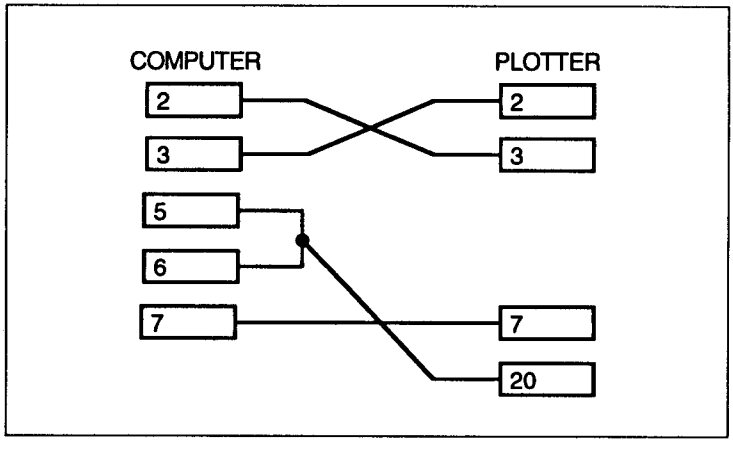


Figure E-1. 25-pin cable wiring diagram.

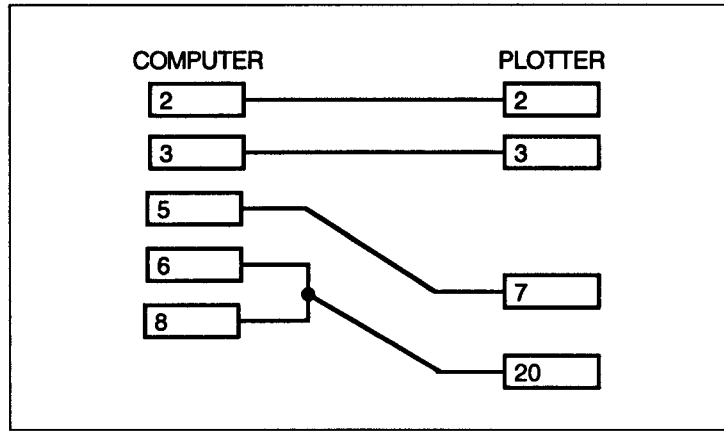


Figure E-2. 9-pin cable wiring diagram.

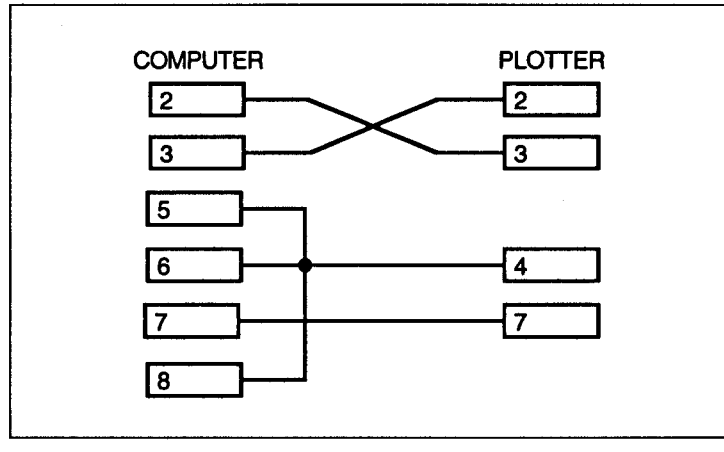


Figure E-3. 25-pin cable to IOline plotter.

Plotter problems

Most plotter problems are caused by incorrectly wired plotter cables. If you have difficulty with your plotter, check the following items before proceeding or calling OrCAD:

1. Wire the cable as shown in figures E-1, E-2, and E-3. If your plotter works with another software package, and does not work with OrCAD, the first item to check is the wiring of your cable. Chances are the other CAD packages only require the TXD and RXD signal lines. OrCAD requires additional connections.

   The cable must be wired as recommended.

2. Check for an open in the cable by performing a continuity check.

3. Read your plotter manual to be sure you understand how the plotter operates. Know how it is programmed for baud rate, parity, word length, and find out what these settings are.

4. Ensure that the plotter baud rate, parity, and word length settings correspond to the plotter configuration information.

5. Select the **View Reference Material** button on the **Schematic Design Tools** screen to see other reference material about plotting.

6. Use DOS to send a plot file to the plotter. This is useful for isolating whether the problem is in the serial port hardware or the plotter hardware. To do this, first send the worksheet to a plot file as outlined below:

   Use the **Plot Schematic** tool to plot to a file.

   Then, use the DOS MODE command to configure the serial channel as in the following example:

   ```
   MODE COM1:2400,N,8,1,P
   ```

   This assumes that you are using serial channel 1 (COM1) and have your plotter set for 2400 baud, no parity, 8 data bits, and 1 stop bit. For more information on the MODE command and asynchronous communications, see your DOS user's guide.

After the serial channel has been configured, send the plot file to the plotter using the DOS COPY command as follows:

```
COPY filename COM1:
```

Where *filename* is the name of the plot file.

If the plotter works, the problem may be in the plotter cable (check the wiring) or the hardware handshaking (check the **Printer/Plotter Output Options** section of the **Configure Schematic Design Tools** screen).

If the plotter does not work, there is a hardware problem. Check the serial card, serial channel configuration, plotter hardware, and the cable.

7. If you're using an IOline plotter, be sure you have PROM version 114 or greater.

General plotter tips

When making a plot, use the proper pens and paper designed for the plotter. Plotter paper has a "memory" to it. If it hangs on the plotter bed for a period of time, it will stretch. This effects the registration of the plot. Plotter paper is also temperature sensitive. Be sure that the paper is at room temperature before plotting. The longer the drawing takes to plot, the more care must be exercised with the paper.

The configuration of the plotter includes the ability to change the velocity of the pens. When the pen cannot draw at the speed the plotter is capable of moving, reduce the velocity. You will need to consult your plotter manual for the range to set the velocity. The velocity can be set only in whole number values.

When you make a plot with different pens, the plotter has a registration inaccuracy that must be considered. If you wish to have the highest quality plot, always use only one pen.

When you are directed by the program to change paper or pens, always wait until the plotter has finished the present plotting activity. Before sending a plot directly to the plotter, be sure that the plotter is on line, the pens are properly set up, and the paper size is correct. When you have a pen that must be manually changed, the **Plot Schematic** program will pause and inform you of the objects to be plotted with the new pen.

Make sure that the template Horizontal and Vertical dimensions are correct for your plotter. Some plotters have a larger margin requirement and, therefore, less usable plotting area. Check your plotters user's manual to get the actual plot area dimensions and change the values in the **Template Table** area of the **Configure Schematic Design Tools** screen accordingly. You have this problem if the top or right edge of your plot is clipped. In addition, you may use more plot area if your plotter has a usable plot area greater than that set in the **Template Table** section of the **Configure Schematic Design Tools** screen. If this is true, change the **Template Table.**

Plot language for HP plotters and printers

HP plotters have changed over time. The older plotters use a version of the HP plotting language called HPGL. The newer plotters and various HP printers support an improved version of this language called HPGL/2. There are two key differences between the two languages. First, HPGL supports coordinates between –32.76 inches and +32.76 inches, whereas HPGL/2 supports a coordinates between –99.99 inches and +99.99 inches. Second, HPGL/2 supports commands that either were not supported by the older plotters or had to be set using buttons on the plotter.

Offsets

By default, HP plotters set the origin (0,0) at the lower left corner of a plot. If you create a plot with a dimension larger than 32.76 inches, coordinates more 32.76 inches from the origin are ignored. On a 42-inch wide E-size drawing, for example, the right 10 inches are not "addressable."

The solution is to move the origin to the middle of the paper and offset the drawing commands so that the lower left corner of the drawing is drawn with negative X and Y coordinates. To do this, you must set offsets in the **Template Table** section of the **Configure Schematic Design Tools** screen.

For HP plotters, the offset values should be negative and one-half the size of the sheet. For example, an E-size drawing is 32 inches high by 42 inches wide, so you would set the **Plot Y Offset** to –16 inches and the **Plot X Offset** to –21 inches.

If your plotter supports only HPGL, you need to set offsets using the buttons on the plotter. If your plotter supports HPGL/2, you can use one of the HPGL/2 drivers—offset commands will be included in the data stream when **Plot Schematic** finds nonzero offset values in the **Template Table**.

Please note that, at present, you must use offsets for large drawings even with HPGL/2 drivers. This is a limitation with the current implementation of **Plot Schematic**.

If your HPGL-only plotter is on a network, be sure to coordinate with other users so your offset requirements do not inconvenience them. Normally, you warn other users, set the offsets, load the paper, ready the plotter, send the plotter data across the network, and reset the plotter to a default state. Depending on your machine's performance, you may want to run **Plot Schematic** and have it send the data directly to the plotter or have **Plot Schematic** create a file and send the file to the plotter (via the network) when the plotter is set up.

On a network, use of an HPGL/2 driver is very convenient because the set-up process is accomplished by commands embedded in the data stream.

HI plotters

The HI 40 Series defaults to 2400 baud, and the 50 Series defaults to 9600. Always check to make sure that the plotter baud rate, and data bit settings correspond to the **Printer/Plotter Output Options** section of the **Configure Schematic Design Tools** screen.

Be sure that the plotter is on line before beginning a plot. The HI plotters do not have a means to set the velocity to the power-up default. If you change any of the velocity settings of the pens in the configuration, you will need to change them all. The velocity ranges can be found in the plotter operation manual for your specific plotter.

On HI plotters that print on D and E size paper, you may need to change the horizontal dimension in the **Template Table** section of the **Configure Schematic Design Tools** screen. For example, you may need to change the D-size horizontal dimension from 32.2 inches to 32.0 inches.

Calcomp plotters

This section lists supported Calcomp plotters and provides setup information for the Calcomp 1043 plotter. You must have one of the listed controllers (supplied by Calcomp) for the plotter to work. See your Calcomp plotter documentation for configuration information.

△ *NOTE: OrCAD has followed Calcomp's recommended procedure for connecting a Calcomp plotter to the IBM PC. Thus, any cabling changes that Calcomp recommends should be followed instead of the cabling information in this manual.*

Table E-1 lists pinouts for connecting Calcomp plotters to AT and XT personal computers using DB25 connectors. If you are already using the plotter, the cable is probably already correct.

PC AT		*PC XT*	
PC	*Plotter*	*PC*	*Plotter*
1	8	8	8
2	3	3	3
3	2	2	2
4	20	20	20
5	7	7	7
6	6	6	6
7	4	4	4
8	5	5	5
9	22	22	22

Table E-1. Pinouts for Calcomp plotter cables.

Table E-2 lists intelligent and nonintelligent Calcomp plotters supported by **Schematic Design Tools 386+** and which plotter driver to use for each supported plotter. Following the table is a key to codes used in the table.

	Controllers				No Controller		
Supported plotters	906	907	907 Rev G	951/953	PCI	960 format	Ext 960 format
Nonintelligent							
1012	Xa	-	-	-	-	-	-
1037	Xa	Xc	Xc	-	-	-	-
1038	Xa	Xc	-	-	-	-	-
1039	Xa	Xc	-	-	-	-	-
1051	Xa	Xc	Xc	Xc	-	-	-
1055	Xa	2	2	2	-	Xa	-
1060	Xa	2	2	2	-	Xa	-
1065	Xa	2	2	2	-	Xa	-
960	Xa	2	2	2	-	Xa	-
970	Xa	2	2	2	-	Xa	-
5200	-	-	-	Xc	-	Xa	-
5500	-	-	-	Xc	-	-	-
5732	-	-	-	Xc	-	-	-
5734	-	-	-	Xc	-	-	-
5742	-	-	-	Xc	-	-	-
5744	-	-	-	Xc	-	-	-
5754	-	-	-	Xc	-	-	-
Intelligent							
945	-	-	1	1	1	-	Xb
945A	-	-	1	1	1	-	Xb
965	-	-	1	1	1	-	Xb
965A	-	-	1	1	1	-	Xb
1042	-	-	-	1	1	-	Xb
1042GT	-	-	-	1	1	-	Xb
1043	-	-	-	1	1	-	Xb
1043GT	-	-	-	1	1	-	Xb
1044	-	-	-	1	1	-	Xb
1044GT	-	-	-	1	1	-	Xb
1073	-	-	1	1	1	-	Xb
1075	-	-	1	1	1	-	Xb
1076	-	-	1	1	1	-	Xb
1077	-	-	1	1	1	-	Xb

Key:

1 Use CALCOMP1.DRV for **Schematic Design Tools 386+**.
2 Use CALCOMP2.DRV for **Schematic Design Tools 386+**.
X Plotter and configuration supported by Calcomp Company. No driver for **Schematic Design Tools 386+**.
a The 906 controller and 960 format lack commands to draw circles, arcs, and dashed lines.
b The extended 960 format is not supported.
c The remaining unsupported configurations result from differences in plotter step size. OrCAD's CALCOMP plotter drivers support only a step size of 2032.

Table E-2. Supported Calcomp plotters and corresponding OrCAD plotter drivers.

No.	*Selection*	*AutoCAD*	*OrCAD*
	Parity Bits	2 - Even 7	0 - None 8
1	Stop bits Clock	1 0 - INT	1 0 - INT
2	Interface	0 - Serial	0 - Serial
3	Host baud rate Mode	9600 PCI	9600 PCI
4	Term muting Checksum enable	No Yes	No No
5	Isochronous EOM	No 13	No 03
6	Direct control XON/XOFF	No No	Yes No
7	Term baud rate Duplex	9600 0 - Full	9600 0 - Full
8	Sync codes Sync code value	2 022	1 002
16	Enable optimization	Yes	No

Table E-3. Calcomp communications and plot management settings.

0—>1	*AutoCAD*	*OrCAD*
POS - 1	0	0
POS - 2	0	1
POS - 3	0	0
POS - 4	0	1
POS - 5	1	0
POS - 6	0	0
POS - 7	0	0
POS - 8	0	0

Table E-4. Switch settings.

Notes on plotter and printer drivers

This section contains notes on the following plotter and printer drivers:

- HP.DRV
- HPLASERx.DRV
- DXF.DRV
- Postscript drivers

HP.DRV

This driver may report a divide error on D- or E-size sheet if you do not set the **Plot X Offset** and **Plot Y Offset** on the **Configure Schematic Design Tools** screen. This is because the HP plotter expects values in plotter units, not inches. One plotter unit is ¼₀th of a millimeter, or approximately 0.00098 inch. The plotter unit value must be a 16-bit integer value (–32768 to +32767). Without offsets, **Plot Schematic** will try to use values larger than +32767.

The plotter driver is passed 16-bit unsigned integer values (0.000 to +65.535) and these are converted into plotter units after the offset is added. For example, the D-size horizontal dimension of 32.2 inches with an offset of 0.000 inches would be converted by the driver to:

$$\frac{32.200}{0.00098} = 32857 \text{ units}$$

This value exceeds the integer limit. To plot D- and E-size drawings, you need the full integer range. Therefore, the large paper plotters have the origin (0,0) in the center of the paper. The drawing coordinate system is adjusted by the **Plot X Offset** and **Plot Y Offset** on the **Configure Schematic Design Tools** screen to move the origin to the correct position. Typically, the offsets are –0.5 times the horizontal and vertical dimensions shown on the template table in **Configure Schematic Design Tools.**

Another consequence of not setting the **Plot X Offset** and **Plot Y Offset** on the **Configure Schematic Design Tools** screen is that plotters having the origin in the center will draw only on the upper right quadrant of the paper. If your drawing appears this way, set the **Plot X Offset** and **Plot Y Offset** on the **Configure Schematic Design Tools** screen to move the origin of the drawing to the lower left corner of the paper.

HPLASERx.DRV Some laser printers have a graphics printing limit. The **Schematic Design Tools 386+** drivers place page breaks after a given number of graphics lines. If you want to change this number, the source code is available on the 24-hour OrCAD BBS.

DXF.DRV This driver puts a drawing into a format usable by AutoCAD and some desktop publishing programs. Use **Plot Schematic** to plot to a file with this driver. In AutoCAD, use the ***dxfin*** command to read the file written by **Plot Schematic**. You will have to rename the file to have a .DXF extension. If you want colors (layers), enter the pen number in the **Pen** entry box in the **Color and Pen Plotter Table** section of the **Configure Schematic Design Tools** screen to put the different objects onto different layers.

△ *NOTE: Because of line definition restrictions, dashed lines will always be layer 0.*

PostScript plotter drivers

To create a PostScript file that can be printed by any PostScript compatible device on either paper or film, use the PSCRIPT.DRV plotter driver. To create ledger-size PostScript images, use the PSCRIPT2.DRV. For more information about PostScript, see the *PostScript Language Reference manual* and the *PostScript Language Tutorial and Cookbook*, both published by Addison-Wesley in 1985.

Encapsulated PostScript

To produce Encapsulated PostScript (EPS) files to import as illustrations into word processing, page layout, and other applications, use one of these four plotter drivers:

- ❖ EPS1.DRV Letter size, landscape
- ❖ EPS2.DRV Letter size, portrait
- ❖ EPS3.DRV Legal size, landscape
- ❖ EPS4.DRV Legal size, portrait

Files produced by these drivers can be used by any application that accepts EPS V2.0.

△ *NOTE: Because the four EPS drivers assume schematics are A-size, when using the landscape-oriented drivers, you may need to scale the plot—0.8 usually works—or select* ***Automatically scale and set X, Y offsets for specified sheet size on Plot Schematic's local configuration screen.***

As with other "standards," some applications interpret EPS a little differently than others. Usually the problem can be corrected with a minor adjustment to the plot file. If you have trouble importing EPS into an application, contact technical support at the developer of your application to determine its exact requirements for EPS. For more information about EPS, contact Adobe Systems Incorporated, 1585 Charleston Road, Mountain View, CA 94039.

Other PostScript

To produce a PostScript file for the special PostScript "environment" within the Macintosh version of Microsoft Word, use the PWORD.DRV plotter driver. PWORD files can be placed into Word documents or can be incorporated by reference using Word's include and Print Merge features. PWORD files start with Word's special *.para.* operator. For more information about Word's PostScript environment, see Microsoft's *Reference to Microsoft Word* manual.



Files and file extensions

This appendix describes the files used and created by **Schematic Design Tools**. The first section, **Design files**, lists the files that typically use the design name as a prefix and a default extension. The second section, **Other files**, lists files that follow other naming conventions.

Unless stated otherwise, all files are stored in your design directory. You can, however, choose to override OrCAD's recommended directory structure and store these files wherever you wish.

Design files

This section lists files alphabetically by default extensions.

Filename prefixes are usually the design name, however you can specify a different prefix if desired. Filename extensions are usually defined on **Local Configuration** screens. You can accept the default extension or enter a different one. It's a good idea to use default extensions to ensure consistent naming conventions.

.ABR Breakpoint file created by ASCTOVST, or by the user in **Edit File**. ASCTOVST in **To Digital Simulation** and **Compile Simulation Specification File** in **Digital Simulation Tools** read this text file and create a binary breakpoint file with a .BRK extension.

See *Chapter 14: Compile Simulation Specification File* in the *Digital Simulation Tools Reference Guide* for more information about .ABR file format.

.AST Stimulus file created by IBUILD from information in the netlist, or by the user in **Edit File**. ASCTOVST in **To Digital Simulation** and **Compile Simulation Specification File** in **Digital Simulation Tools** read this text file and create a binary stimulus file with a .STM extension.

See *Chapter 14: Compile Simulation Specification File* in the *Digital Simulation Tools Reference Guide* for more information about .AST file format.

.ATR Trace file created by IBUILD from information in the netlist, or by the user in **Edit File**. ASCTOVST in **To Digital Simulation** and **Compile Simulation Specification File** in **Digital Simulation Tools** read this text file and create a binary trace file with a .TRC extension.

See *Chapter 14: Compile Simulation Specification File* in the *Digital Simulation Tools Reference Guide* for more information about .ATR file format.

.BAK Backup of a schematic file created by **Cleanup Schematic, Annotate Schematic, Back Annotate, Update Field Contents, Draft,** and INET /W, before processing the schematic. This file is in binary format.

.BOM Bill of materials file created by **Create Bill of Materials.** This text file contains a list of all parts used in a design.

.BRK Binary breakpoint file created by ASCTOVST in **To Digital Simulation, Compile Simulation Specification File** in **Digital Simulation Tools, or Breakpoint Editor** in **Simulate** in **Digital Simulation Tools. Simulate** extracts breakpoint information from this file during simulation.

See *Chapter 13: Build Simulation Specification File* in the *Digital Simulation Tools Reference Guide* for more information about .BRK files.

.ERC Report file containing warnings and errors about your design's conformity to basic electrical rules. This file is created by **Check Electrical Rules**. You can use **Edit File** to view this text file or point at an error object on a schematic and select **Draft's INQUIRE** command.

.IGS IGES (Initial Graphic Exchange Specification) plot file of a single worksheet. **Plot Schematic** creates this text file.

.INF Incremental connectivity database file created by INET in **Create Netlist** and **Create Hierarchical Netlist**. INET creates a text file for each schematic in a design. The file contains information about the devices, connections, pipe commands, and title block. ILINK uses .INF files to create a linked connectivity database, and HFORM uses them to create a hierarchical netlist.

See *Chapter 10: Creating a Netlist* in the *Schematic Design Tools Reference Guide* for detailed information about .INF files.

.INS Instance file in the intermediate netlist structure created by ILINK in **Create Netlist** and **Create Hierarchical Netlist**. This binary file contains instance information for all the sheets in the design. IFORM uses the .INS, .RES, and .PIP files to create a flat netlist.

.INX Incremental connectivity database file created by INET in **Create Netlist** and **Create Hierarchical Netlist**. INET creates a text file containing a list of all the .INF files it creates and uses it to keep track of the .INF files.

.LIB Library of parts or other schematic symbols supplied with **Schematic Design Tools** or created by **Compile Library** or **Edit Library**. Libraries are binary files. Libraries are typically stored in the \ORCADESP\SDT\LIBRARY directory.

.MAC Macro file supplied with **Schematic Design Tools** or created by **Edit File, Draft,** or **Edit Library.** You can customize these text files. See *Macro Options* in *Chapter 1: Configure Schematic Tools* and the **MACRO** command in *Chapter 2: Draft* for more information about macros.

.MAP Supplemental netlist file created by IFORM in **Create Netlist** or HFORM in **Create Hierarchical Netlist.** Several netlist formats require a second text file. For example, in the SPICE format, a .MAP file contains a list of node numbers.

.NET Netlist file created by IFORM in **Create Netlist** or HFORM in **Create Hierarchical Netlist.** The format of this text file depends on which netlist format file you specify during local configuration.

.PIP Pipe commands file created by ILINK in **Create Netlist.** This text file contains pipe commands extracted from a schematic. If no pipe commands exist, ILINK does not create a file. IFORM in **Create Netlist** uses .PIP files.

.PLT Plot file created by **Plot Schematic** or **Print Schematic.** This binary file contains plot information designed for input to a device that accepts vector commands.

.PNN Persistent netname file created by IFORM. This binary file contains a list of system-generated netnames assigned to any unnamed nets.

.PRN Print file created by **Plot Schematic** or **Print Schematic.** This binary file contains print information designed for input to a device that accepts raster commands.

.RES Resolved file created by ILINK in **Create Netlist**. This binary file contains information about the connectivity of the parts in the .INF files. IFORM in **Create Netlist** uses .RES files.

.SCH Schematic worksheet file created by **Draft** in binary format.

.SRC Library source file created by **Decompile Library** or DECOMPOSER in **Archive Parts in Library**. This text file describes library parts using OrCAD's Symbol Description Language that **Compile Library** can use as input to create a library.

.STM Binary stimulus file created by ASCTOVST in **To Digital Simulation, Compile Simulation Specification File** in **Digital Simulation Tools**, or **Stimulus Editor** in **Simulate** in the **Digital Simulation Tools**. This file contains stimulus information used by **Simulate**.

.TRC Binary trace file created by ASCTOVST in **To Digital Simulation, Compile Simulation Specification File** in **Digital Simulation Tools**, or **Trace Editor** in **Simulate** in **Digital Simulation Tools**. This file contains trace information used by Simulate.

.TWG Tree list created by **Show Design Structure**. This text file lists the sheets in a design and shows the relationships between the sheets.

.XRF Cross-reference file created by **Cross Reference Parts**. This text file tells you where each part is located on a worksheet.

Other files

This section lists files that don't follow the naming convention of design name and default extension.

#ESP_OUT.TXT This file contains the screen output from the last tool executed in the design. This file is located in the design directory from which you were running the tool.

HARDCOPY.PRN Print file created by **Draft's HARDCOPY** command. For information about printing this file, see *HARDCOPY* in chapter 2. This file is located in the design directory.

ORCADESP.DAT Data file that stores configuration information for each design. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.

See *Chapter 5: Design environment technical information* in the *ESP Design Environment User's Guide* for more information about data files.

SDT.BCF Binary configuration file for **Schematic Design Tools**. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.

See *Chapter 5: Design environment technical information* in the *ESP Design Environment User's Guide* for more information about configuration files.

SDT.CFG Configuration file for **Schematic Design Tools**. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.

See *Chapter 5: Design environment technical information* in the *ESP Design Environment User's Guide* for more information about configuration files.

Reference files Reference files include information about **Schematic Design Tools** such as update information and listings of parts in libraries. You can access these files by selecting **View Reference**. Reference files are stored in the \ORCADESP\SDT directory.

Tutorial files

The installation software creates the TUTOR design and places files you use when completing tutorial activities in the *Schematic Design Tools 386+ User's Guide*. The tutor design is located in the \ORCAD\TUTOR directory.

Update file

An update file is a text file that you create using **Edit File**. **Update Field Contents** uses this file to update part fields.

The update file consists of pairs of strings. A string is delimited with single quotes and is followed by any number of spaces, tabs, or new lines. For example:

```
'74LS00'        '14DIP300'
'74LS138'       '16DIP300'
'74LS163'       '16DIP300'
'74LS373'       '20DIP300'
'74LS245'       '20DIP300'
'8259A'         '24DIP600'
```

Before using an update file with **Update Field Contents**, you must:

- Define which part field should match the first item in each pair in the update file. This is done in the **Key Fields** area of the **Configure Schematic Design Tools** screen.

  See *Chapter 1: Configure Schematic Tools* for more information about defining key fields.

- Define which part field is to be updated with the second item in each pair in the update file. This is done on the local configuration screen for **Update Field Contents**.

  See *Chapter 14: Update Field Contents* for more information about update files.

Was/Is file

A was/is file is a text file—containing old and new reference designators—that you create using **Edit File**. **Back Annotate** uses this file to update reference designators.

An entry in a was/is file begins with the old reference designator that you want to modify, is followed by any number of spaces, tabs, or new lines, and ends with the new reference designator value. For example:

```
R1    R5
R2    R12
R3    R6
C5    C1
C12   C2
U5C   U1A
U3B   U3A
```

In this example, each occurrence of R1 in the design changes to R5, R2 becomes R12, and so on.

See *Chapter 6: Back Annotate* for more information about Was/Is files.

Netlist format files

Compiled and uncompiled versions of netlist format descriptions used by IFORM in **Create Netlist** and HFORM in **Create Hierarchical Netlist**. See chapters 10, 11, and 12, and appendix A for more information about creating netlists.

formatName.C

Netlist format source files that OrCAD provides or you create with a text editor. These text file contain formatting information for a netlist in over thirty formats. IFORM and HFORM cannot use the information in these file when they create a netlist—they require the corresponding .EXE file. Netlist format source (.C) files are typically found in the directories \ORCADESP\SDT\NETFORMS\NETSRCF and \ORCADESP\SDT\NETFORMS\NETSRCH.

FformatName.EXE
HformatName.EXE

Compiled version of the flat or hierarchical netlist format source (.C) files. IFORM or HFORM extracts the netlist format guidelines from this binary file when creating a netlist. Compiled netlist format files are typically found in the \ORCADESP\SDT\NETFORMS directory.



Converting designs, worksheets, and libraries

Schematic Design Tools 386+ uses a 32-bit database to store designs; **Schematic Design Tools Release IV** uses a 16-bit database. You can move between these formats using the information in this appendix. If you need to upgrade from Release IV of the software, you will find instructions at the end of this appendix in *Upgrading from SDT IV*.

Converting designs

The CONVERT program creates a new design directory, copies all files from the current design directory to the new directory, and converts all schematics in the hierarchy or flat design.

CONVERT keeps you posted as it converts files; it tells you what it is about to do before it does it and gives you an opportunity to cancel the program

CONVERT accommodates any combination of these special situations:

- The design being converted contains schematics not related to the root schematic.
- You want to convert all of your designs at one time.
- You want to convert your designs without changing the name of the design.
- The design directory contains subdirectories with worksheet files in them.

The default CONVERT command converts schematic files from 16-bit to 32-bit format. To convert schematic files from 32-bit format to 16-bit format, simply use the /L switch.

CONVERT command syntax

CONVERT *Source Destination [Switches]*

The parameters for CONVERT are described in the following sections.

Source

Source is the name of the design directory to convert. This design must be in the path named by the ORCADPROJ environment variable. CONVERT descends into *Source's* subdirectories to *copy* files. However, it only *converts* worksheet files or libraries in *Source's* subdirectories if you use the /S switch.

Source is one of the following:

- **A directory name.** *Source* is a directory on the path defined by the ORCADPROJ environment variable. When converting designs, only worksheet files connected to a root schematic called *Source.sch* are converted.
- **A directory name that is a subdirectory on the path defined by the ORCADPROJ environment variable, followed by a wildcard.** This converts all worksheet files or libraries in the design directory that match the wildcard. These characters are valid wildcard characters.

  * matches zero or more characters
  ? matches one character

  The path specified can be either relative to the current directory or a full path. When a path relative to the current directory is specified, for example ..\FRED*.SCH, CONVERT looks for a subdirectory within the current directory named FRED.
- **A standalone wildcard.** If a wildcard string is used without a preceding design name, all worksheet files or libraries in the current directory that match the wildcard are converted.

  If you use the /A switch, *Source* must be a wild card. A design name is not allowed.

Destination

Destination is the name of the new design to copy files to. Files in *Source* are copied to *Destination;* files with names in the form *Source.** are renamed to have *Destination.** names. Other files in the *Source* directory are copied to *Destination* directory without their original names being changed.

When converting designs, the SDT.BCF configuration file is converted to 32-bit format. The SDT.CFG file is updated to contain new filenames. The original version of SDT.CFG is renamed SDT.BAK.

△ *NOTE: All pathnames specified in* Destination *must be absolute. You cannot use the relative path ".." to refer to the parent directory.*

Destination is one of the following:

- **A directory name.** CONVERT creates a directory called *Destination* on the path defined by the ORCADPROJ environment variable and places the converted files here. If the directory does not exit, CONVERT creates it. If the directory does exist, and CONVERT is converting designs, it displays a message to warn you of the problem. If the directory does exist and CONVERT is converting library files (/P was specified), then CONVERT places the converted library files in the existing directory.
- **A pathname followed by a directory name.** The design or libraries in the design are copied into the directory specified by the pathname. The design name may be the same as the *Source* name (so long as it is in another project directory).
- **A standalone pathname.** When converting designs, all design directories or libraries are copied into the directory specified by the pathname and then converted.

  When converting libraries, the files are decompiled in the *Source* directory and then compiled into the *Destination* directory.

  This is only valid when you use the /A switch.

Switches

Switches are described in table G-1.

Switch	*Description*
/A	Process all directories that are in the path defined by the ORCADPROJ environment variable. After running CONVERT with the /A switch, you must set the ORCADPROJ environment variable to the *Destination*.
/B	Convert 16-bit design database to 32-bit design database. This switch cannot be used if the /L switch is used.
/L	Convert 32-bit design database to 16-bit design database. This is the default. This switch cannot be used if the /B switch is used.
/P	Convert libraries. Use this switch with either /B (to convert from 16-bit to 32-bit) or /L (to convert from 32-bit to 16-bit).
/S	Descend into subdirectories and convert all files that match the wildcard given in *Source*. You must include a wildcard in *Source* to specify which files to convert.

Table G-1. CONVERT switches.

Examples of design conversions

The following examples illustrate different ways you can use CONVERT to convert designs.

-
```
C:> convert fred tom
```

  This example copies all files and subdirectories from FRED to TOM on the path defined by the ORCADPROJ environment variable. It converts all worksheet files connected to the root schematic FRED.SCH to 32-bit worksheet files. It renames the converted root schematic to TOM.SCH.

-
```
C:> convert fred\*.sch tom /s
```

  This example creates a design directory named TOM and copies all files from FRED into TOM. It converts all worksheet files with a .SCH extension into 32-bit worksheet files. Because the /S switch is used, any worksheet files that exist in subdirectories are converted as well. The subdirectories are preserved and translated worksheet files are copied into the appropriate subdirectory.

- ```
  C:> convert *.s?? tom /s
  ```

  This example converts all files in the current directory and—because the /S switch is used—in any subdirectory of the current directory that match the *.S?? wildcard (for example, FRED.SCH, JOE.SHT, or JOHN.S01) to 32-bit worksheet files. All files are copied into a new design directory called TOM. The subdirectories and translated schematics are copied into the appropriate directory.

- ```
  C:> convert *.dwg orcad32 /a /s
  ```

  Because the /A switch is used, this example copies *all* design directories and—because the /S switch is used—all subdirectories within the ORCADPROJ path into the directory ORCAD32. The names of the design directories are preserved. All worksheet files with .DWG extensions in each design or design subdirectory are converted to 32-bit worksheet files. In this example, you would need to set your ORCADPROJ variable to ORCAD32 in order to access these designs from within the ESP design environment.

Converting individual worksheets

The 16TO32 and 32TO16 programs convert worksheet files between 16-bit and 32-bit formats. They can operate on an entire hierarchy based on the root sheet. These programs copy the converted files to a specified directory. They don't create a new directory or copy any other files to the specified directory.

16TO32 and 32TO16 command syntax

`16TO32` *source destination* [*switch*]

`32TO16` *source destination* [*switch*]

The parameters for 16TO32 and 32TO16 are described in the following sections.

Source

You usually run 16TO32 and 32TO16 from the *source* directory.

source is the path and filename of the worksheet file you plan to convert. If the worksheet file is the root of a design, all the worksheets referred to by the root worksheet or named in the !LINK list on the root worksheet are converted.

Destination

destination is the path and name of the converted worksheet file. If the worksheet file is the root of a design, all the worksheets referred to by the root worksheet or named in the !LINK list on the root worksheet are converted. The destination directory must already exist.

/O switch

There is only one switch for these two programs and it is described in table G-2.

Switch	*Description*
/O	Treat this file as a single sheet.

Table G-2. Switch for 16TO32 and 32TO16.

Converting libraries

Schematic Design Tools 386+ uses libraries in 32-bit format. You can use CONVERT with the /P switch to convert edited or custom libraries created with **Schematic Design Tools Release IV** from 16-bit format to 32-bit format. You can use CONVERT with the /P and /L switches to convert libraries from 32-bit format back to 16-bit format.

This is how the CONVERT program converts libraries:

1. CONVERT converts a library to a text-only source file using a program called DECOMP16 (this program is the same as the **Decompile Library** tool in **Release IV**) or DECOMP (**Decompile library**).

   For information about **Decompile Library**, see *Chapter 21: Decompile Library*. For information about using DECOMP16, see the section about DECOMP in *Appendix A: Command line controls.*

2. CONVERT then compiles the source file into a 32-bit library using COMPOSER which is **Compile Library** in **Schematic Design Tools 386+** or into a 16-bit library using a program called COMP16.

   For information about **Compile Library**, see *Chapter 24: Compile Library.*

△ ***NOTE: Compile Library in Schematic Design Tools 386+*** *removes any bitmaps from your libraries. If you are converting any parts that contain bitmaps, but don't contain vector definitions, you will have to redraw them unless you can make the conversion with MAKEVECT.*

There is a program in the customer-contributed area of OrCAD's 24-hour BBS that can make replacing bitmaps with vectors a little less painful. Called MAKEVECT, it may save you some time and trouble.

DECOMP16 and COMP16

Two Release IV programs are included with **Schematic Design Tools 386+**. DECOMP16 is the Release IV (16-bit) version of **Decompile Library** (DECOMP). COMP16 is the Release IV (16-bit) version of **Compile Library** (COMPOSER). These programs are available from the command line only. (See Appendix A for command line syntax and switches.) These files are useful when manually converting 16-bit library files to 32-bit ones and vice versa. The CONVERT program calls these as necessary when converting libraries.

Converting sheetpath parts

Since the worksheet files that sheetpath parts point to are stored in the \ORCADESP\SDT\LIBRARY\ directory, CONVERT can't find these files when it tries to descend into a hierarchy. To convert these worksheet files, use one of these methods:

- If you only have one sheetpath part worksheet file in your \ORCADESP\SDT\LIBRARY\ directory, run the 16TO32 utility on the file. 16TO32 is described later in this appendix.
- If you have more than one sheetpath part worksheet file in your \ORCADESP\SDT\LIBRARY\ directory, follow these steps:

  1. Create a temporary directory under your ORCADPROJ directory.
  2. Copy all of your sheetpath part worksheet files from your \ORCADESP\SDT\LIBRARY\ directory to the temporary directory.
  3. Run CONVERT in the temporary directory using this syntax:

     ```
     C:\tempdir> convert *.sch newdir /S
     ```

     This creates a new directory containing the 32-bit sheetpath part worksheet files.
  4. Move the 32-bit sheetpath part worksheet files to your \ORCADESP\SDT\LIBRARY\ directory.

Upgrading from SDT IV

The INSTALL program will convert the files listed below from 16-bit format to the 32-bit format required by **Schematic Design Tools 386+**:

- All designs and all custom library files that reside in directories under the directory specified by your ORCADPROJ environment variable, or in a directory you specify.
- All custom libraries that reside in the directory \SDT\LIBRARY\, which is in the location specified by the ORCADESP environment variable, or a directory you specify.

  If you have designs or custom libraries that reside in locations other than those described above, you must convert them to 32-bit format using the CONVERT program which is described in this appendix.

△ *NOTE: The directory structure recommended by OrCAD is shown in the* ESP Design Environment User's Guide *in* Chapter 5. *Using the standard OrCAD directory structure makes installing new software and updates easier and faster.*

For files not listed above, **Schematic Design Tools 386+** provides utility programs for converting designs and libraries between 16-bit and 32-bit formats. These programs are described in this appendix.

Custom netlist format files

If you are upgrading from **Schematic Design Tools Release IV**, and you have created custom netlist format files, you need to convert them separately. To convert, you need to edit the source code, then compile and link the code.

If you need technical information about format files, see *Appendix D: Creating a custom netlist format.*

First, copy custom netlist format files or OrCAD-provided netlist format files that you modified from your back-up disks into the appropriate directory:

- Flat netlists go in the directory \ORCADESP\SDT\NETFORMS\NETSRCF.
- Hierarchical netlists go in the directory \ORCADESP\SDT\NETFORMS\NETSRCH.

To convert these files, you need to make some minor changes to the format files and compile them as described below.

Save symbols added by an AddSymbol call

Change all symbols added using the call `AddSymbol ( "`*symbol_name*`" )` so that the symbol is saved. For example, if your format file read:

```
AddSymbol ( "MyNand" );
```

change the code so that it declares an `int MyNand`, and assign the value returned from the function to that `int`. For example:

```
static int MyNand;

Initialize ()
{
        MyNand = AddSymbol ( "MyNand" );
          ⋮
}
```

Note that `MyNand` is made global to the format file.

Add the standard access symbols

If you customized a part-oriented netlist, you need to add the standard access symbols to the format file. The possible symbols are:

```
SIGNALS     NODES
ALL         PARTS
NETS
```

The most commonly used symbols are SIGNALS, ALL, and NODES. You can make these symbols available by adding each signal and then setting its value to the string equivalent of its name. An example is shown at the top of the next page.

```
static int SIGNALS, ALL, NODES;
    ⋮
Initialize()
{
    SIGNALS = AddSymbol ( "SIGNALS" );
    SetSymbol ( SIGNALS, "SIGNALS" );

    ALL = AddSymbol ( "ALL" );
    SetSymbol ( ALL, "ALL" );

    NODES = AddSymbol ( "NODES" );
    SetSymbol ( NODES, "NODES" );
       ⋮
}
```

Compiling and linking a netlist format file

To compile and link a flat or hierarchical netlist format file, use OrCAD's DOS batch program called MKNETFMT.BAT, as explained in *Appendix D: Creating custom netlist formats.*

Custom libraries

If you are upgrading from **Schematic Design Tools Release IV**, and you have custom libraries that reside in locations other than \ORCADESP\SDT\LIBRARY\, you need to convert them separately.

1. Copy the OrCAD-provided libraries that you modified in Release IV from your backup disks into the directory \ORCADESP\SDT\LIBRARY.
2. Convert these libraries to 32-bit format using this command line:

   `CONVERT` *Source.lib Destination.lib* `/B /P`

CONVERT calls additional programs during library conversion. For information about this see *Converting libraries* elsewhere in this appendix.

Schematic Design Tools 386+ ignores bitmap definitions and requires vector definitions. If you have parts that have only bitmap definitions, you may need to redraw them; check the *NOTE* below. For more about bitmaps and vectors, see *Defining a graphic symbol* in *Chapter 22: Creating a library source file with a text editor.*

△ *NOTE: There is a program in the customer-contributed area of OrCAD's 24-hour BBS that can make replacing bitmaps with vectors a little less painful. Called MAKEVECT, it may save you some time and trouble.*

Installing OrCAD/PCB II

If you install OrCAD/PCB II or any of OrCAD's Release IV tool sets *after* installing **Schematic Design Tools 386+**, you need to replace or convert the tutorial files. (The Release IV and OrCAD/PCB II products use 16-bit versions of TUTOR's schematic files; the 16-bit schematics will not work with **Schematic Design Tools 386+**.) You can either reinstall **Schematic Design Tools 386+** and select the option to install just the tutorial files, or convert the 16-bit schematics to 32-bit ones, following the instructions in the *Converting designs* section of this appendix.

If you install OrCAD/PCB II after installing **Schematic Design Tools 386+**, there is one more file that will need to be replaced or converted. OrCAD/PCB II installs a 16-bit version of the schematic design library PCBDEV.LIB. As with the tutorial files, you have two options: either reinstall PCBDEV.LIB from the **Schematic Design Tools 386+** disks, or convert the 16-bit library to a 32-bit library, following the instructions in *Converting libraries* of this appendix.

A

analog ■ Circuitry where both voltage and frequency output vary continuously as a function of the input.

annotation ■ Assigning reference designators to components in a schematic.

ASCII ■ An acronym for *American Standard Code for Information Interchange;* a seven-bit code used to represent letters of the alphabet, the ten decimal digits, and other instructions used to edit text on a computer, such as Backspace, Carriage Return, Line Feed, etc.

B

bitmap ■ An image made up of dots (bits).

bulletin board system ■ A computer dedicated to maintaining messages and software and making them available over telephone lines. People *upload* (contribute) and *download* (gather) messages by calling the bulletin board from their own computers. Abbreviated BBS.

button ■ A pushbutton-like image that you click to initiate an action.

byte ■ A piece of computer data composed of 8 contiguous bits that are grouped together as a single unit.

C

CAE ■ An acronym for *computer aided engineering.*

check box ■ A small square button: ❑. Check boxes are used in lists of options when more than one option can be active at a time.

child ■ A worksheet containing circuitry referred to by a sheet, sheet part, or sheetpath part. The child may contain module ports that connect signals from this worksheet to signals on the parent. A child can also be a parent, if it contains a child.

complex hierarchy ■ A design in which two or more sheet symbols reference a single worksheet. Compare with *simple hierarchy.*

configuration ■ The information a program uses to operate. The configuration can be tailored to your needs.

connectivity database ■ The *connectivity database* consists of the incremental connectivity database (created by INET) and the linked connectivity database (created by ILINK). It describes the connectivity of a design, and is used to transfer a design to **Digital Simulation Tools** or **PC Board Layout Tools.** See *incremental connectivity database* and *linked connectivity database.*

cursor ■ A square marker inside a text field showing where characters typed on the keyboard will appear: ■ See *pointer.*

D

default ■ A value or setting provided by the software that is assumed to be correct in most cases and is used if no other value is entered.

design cycle ■ The process of conceiving, developing, testing, and producing a circuit.

Design Management Tools ■ Tools you access from the ESP design environment that create and modify designs and files, back up designs, and suspend to the system. For information about using these tools, see the *OrCAD ESP/Design Environment User's Guide.*

digital ■ Circuitry where data in the form of digits are produced by binary on and off or positive and negative electronic signals.

dpi ■ An abbreviation for *dots per inch.*

E

EDA ■ An acronym for *electronic design automation.*

editor ■ A tool used to create or modify a design file.

EMS ■ An acronym for Expanded Memory Standard.

entry box ■ A box indicating that something (text or numbers) should be entered using the keyboard: [ ]

F

flat design ■ A schematic structure in which output lines of one sheet connect laterally to input lines of another sheet through graphical objects called *module ports.* Flat designs are practical for small designs of three or fewer sheets. See *module port, schematic, hierarchical structure.*

H

hierarchical design ■ A schematic structure in which sheets are interconnected in a tree-like pattern vertically and laterally. At least one sheet, the root sheet, contains symbols representing other sheets, called subsheets.

I

incremental connectivity database ■ INET produces the *incremental connectivity database.* It consists of an incremental connectivity database file (.INF) for each sheet in the design and an .INX file. The .INF file is a description of connectivity on each sheet. The .INX file lists each sheet referenced in the design. The *incremental connectivity database* is used by ILINK to create an incremental netlist. See *connectivity database* and *incremental netlisting.*

incremental netlisting ■ A method of creating a netlist in which only changed worksheets are processed each time **Create Netlist** or **Create Hierarchical Netlist** runs.

initial macro ■ A macro that runs automatically when you run **Draft** or **Edit Library**. For the initial macro to work, you must configure **Schematic Design Tools 386+** to load a macro file containing the desired macro definition.

intermediate netlist structure ■ ILINK produces the *incremental netlist structure*. This consists of the .INS (instance) file, the .RES (resolved) file, and the .PIP file (contains pipe link commands). These files are used by IFORM to create a netlist in one of over 30 formats.

K

K ■ An abbreviation for *kilobyte*. See *kilobyte*.

key field ■ To tell **Draft** and other tools which fields you want to combine and compare, *key fields* are used. A key field lists the part fields to combine and compare. Key fields are defined on the **Configure Schematic Design Tools** screen.

kilobyte ■ One kilobyte is equal to 2^{10} (1,024) bytes. The prefix "kilo" is taken from the metric system, where it stands for "one thousand." Abbreviated *K*.

L

library ■ A collection of standard, often-used part symbols stored as templates to speed up design.

librarian ■ A tool used to manage or create library parts.

local configuration ■ Configuration settings for a particular button. Roughly synonymous with *command line switches*. The same tool can have different configuration in different places in the same design. For example, **Netlist** is configured differently under the **To Layout** button and under the **To Simulate** button.

M

MB ■ An abbreviation for *megabyte*. See *megabyte*.

macro ■ Series of commands you can execute automatically at the touch of a single key or key combination. Macros dramatically reduce the number of keystrokes required to perform complex or repetitive actions.

match string ■ A character string used to find, or point to, a library part. For all parts on the schematic, the character string is compared to the contents of a specific part field until a match is found.

megabyte ■ One megabyte is equal to 2^{20} (1,048,576) bytes. The prefix "mega" is taken from the metric system, where it stands for "one million." Abbreviated *MB*.

module port ■ Graphical objects that conduct signals between schematic worksheets. See *flat file.*

N

net ■ Just as signals are conducted between schematic worksheets through module ports, they are conducted into and out of sheet symbols through graphical objects called *nets.*

netlist ■ An ASCII file that lists the interconnections of a schematic diagram by the names of the signals, modules, and pins connected together on a PCB. The nodes in a circuit. See *incremental netlisting.*

P

PCB ■ An acronym for *printed circuit board.*

package ■ A physical component. A package may contain one or more sub-parts. For example, a 2N3905 transistor, a fuse, and a 74LS00 are packages.

pan ■ To change the portion of the worksheet being viewed by dragging the pointer from one location on the worksheet to another location. As you drag the pointer, the worksheet *pans* across the screen.

parent ■ A worksheet that contains hierarchical references to other worksheets. These references are either sheets, sheet parts, or sheetpath parts, which are, from the viewpoint of the parent worksheet, children.

part ■ A schematic symbol that represents an object. The object can be either a package or another worksheet. OrCAD schematics can have four kinds of parts: packages, sheets, sheet parts, and sheetpath parts.

part field ■ A slot for holding text or data to be associated with a part. Each part has two part fields reserved for part value and part reference. It has eight other part fields that can be used to store other useful information. See *key fields.*

pixel ■ Any of the little dots of light that make up the picture on a computer or television screen. The name is short for *pic*ture *el*ement. There more pixels there are in an area—the smaller and closer together they are—the higher the *resolution.* Sometimes pixels are just called dots.

pointer ■ An arrow on the screen that moves as you move the mouse: ↖ See *cursor.*

processor ■ A tool that subjects a design file to a specific process.

programmable logic device ■ A type of integrated circuit that contains fuses that can be blown, eliminating certain logical operations in the device and leaving others intact, giving the device one of many possible logical architectures or logical configurations.

prompt ■ A query from a program asking you to enter specific information.

Q

quiet mode ■ An option on the local configuration screen for many tools. When quiet mode is selected, tracking information does not echo on the screen. Only execution messages and error messages (if any) display. If quiet mode is turned off (not selected), the tool displays intermediate tracking information in the monitor box at the bottom of the screen. For most applications you do not need to turn quiet mode on.

R

radio button ■ A small round button: ○. Radio buttons are used in lists of mutually exclusive options: only one button can be active at a time.

raster ■ An array of dots.

reporter ■ A tool that creates a report, but does not modify design data.

root directory ■ The main directory on your computer; the directory that the computer boots from.

root sheet ■ The worksheet at the top of a flat or hierarchical design. A design has only one root worksheet. A root worksheet may also be a parent in hierarchical designs.

S

schematic ■ A graphical representation of a circuit using a standard set of electronics symbols. See *flat design, hierarchical design,* and *root sheet.*

scroll buttons ■ Buttons used to move a directory in its window so that a different part is visible. The four scroll buttons are:


Page Up
Line Up
Page Down
Line Down

sheet ■ A schematic symbol referring to a worksheet located in the design area and containing circuitry. The connection points on sheets are called sheet nets. See *nets* and *sheet net.*

sheet net ■ The point at which a signal from a parent connects to a module port on a child. Sheets, sheetpath parts, and sheet parts each have sheet nets.

sheet part ■ A library part modified from a sheetpath part to represent a unique instance of a library circuit. Sheet parts refer to worksheets located in the design area as opposed to worksheets located in the library directory. The symbol resembles package symbols, but the pins are called sheet nets.

sheetpath part ■ A part representing a library circuit. The worksheet referenced by the sheetpath part is stored in the library directory. The pins on a sheetpath part are called sheet nets to distinguish them from pins on a package.

simple hierarchy ■ A one-to-one correspondence between sheet symbols and the schematic diagrams they reference. Each sheet symbol represents a unique subsheet. See *hierarchical design.*

sub-part ■ A gate or some other sub-division of a package. Each sub-part in a package has a unique reference designator comprised of a prefix common to all the parts in the package and a letter unique to each part. For example, an instance of a 74LS00 in a design with a package reference designator of `U15` would have reference designators for each of the four sub-parts as `U15A`, `U15B`, `U15C`, and `U15D`.

syntax ■ The formal structure of a language. Syntax includes the rules for making statements in the language, but excludes the meanings of the statements.

T

tag ■ A marked or saved location on a schematic or layout. You can use the **JUMP** command to go to a tag.

text export ■ The process of copying text from a schematic worksheet to an ASCII file.

text import ■ The process of copying text from an ASCII file to a schematic worksheet.

TTL ■ An acronym for *transistor transistor logic.*

tool ■ A tool is a computer program you can use to do some useful task. Tools are grouped into five categories: editors, processors, reporters, librarians, transfers.

tool set ■ A collection of tools designed to perform a suite of electronic design automation tasks. OrCAD tool sets include: **Schematic Design Tools, Programmable Logic Design Tools, Digital Simulation Tools,** and **PC Board Layout Tools.**

transfer ■ A tool that transfers design information from one tool set to another tool set. Also runs whatever processes are necessary to go from one tool set to another.

U

upload ■ The process of sending a file to another computer.

user button ■ A button that you can set up to perform whatever combination of functions you find useful (such as run programs or batch files). User button definitions are saved with the design files, so you can create design-specific buttons and not worry about overwriting user button definitions for other designs.

V

vector ■ A series of points with a specific function defined. For example, a vector for a line specifies a line function, a beginning point, and an ending point.

W

wildcard ■ A symbol that means any character or any sequence of characters (just as a wild card in poker can stand for any card). Wildcards are useful in searches.

worksheet ■ **Draft** calls the sheets of drafting paper on which the schematics are drawn *worksheets*. Worksheets appear on the computer screen as a rectangular area in which you can place parts and draw wires.

Z

zoom ■ The ability to change the view on the screen by making the objects appear larger or smaller.

16TO32 and 32TO16 *726*
3-state pins *see also pins*
  defined *333*
32-bit to 16-bit conversion *721*
<End> key *5*
<Enter> key *xxx, xxxi*
<Esc> key *xxx*
<Home> key *5*
<Page Down> key *5*
<Page Up> key *5*
<Space bar> key *xxxi*
<Tab> key *xxxi*
|LINK commands *see also pipe link*
  in flat designs *209*

A

ABR files *713, 721*
access symbols *731*
AccessChild function in netlist format files *676*
AccessKeyWord function in netlist format files *676*
AccessPart function in netlist format files *676*
access_constant type definition in netlist format files *672*
Active library size *17*
AddressLine1-AddressLine4 symbol in netlist format files *666*
AddSignalName function in netlist format files *676*
AddSymbol function in netlist format files *676*
AEX32 *see also Decompile Schematic*
  command line controls *485*
AGAIN command
  Draft *64*
  Edit Library *292*
AIM32 *see also Compile Schematic*
  command line controls *485*
Algorex netlist format *514*
Allegro netlist format *515*
alphanumeric pin numbers *270*
AlteraADF netlist format *516*
  including equations in the netlist *517*
analog, defined *735*
ANNOTATE *see also Annotate Schematic, 457*
  command line controls *486*
Annotate Schematic *189-194*
  before and after annotation *191*
  execution *189*
  Key Fields *40-42, 190*
  Local Configuration *192*
  part designation *38*
  preparing for simulation *193*
  processing complex hierarchies *193*
  reference designators *40-41*
  updating all reference designators *194*
  updating unannotated reference designators *194*
annotation, defined *735*
ANSI
  grid references *19*
  title block *18*
AppliconBRAVO netlist format *520*
AppliconLEAP netlist format *521*
Archive Parts in Schematic *264*
  COMPOSER *278*
  creating a library source file *280*
  execution *277*
  LIBARCH *278*
  Local Configuration *278*
    Configure LIBARCH *279*
  sheetpath parts *281*
  string files *280, 281*
ASCII *129, 735*
AST files *714*
ATR files *714*
Available Display Drivers *8*
Available Libraries *14*
Available Plotter Drivers *10*
Available Printer Drivers *9*

B

Back Annotate *195-197*
  execution *195*
  Local Configuration *196*
  selectively updates reference designators *189*
  Was/Is files *195*
back up files in Draft *152*
BACKANNO *see also Back Annotate*
  command line controls *487*
BAK files *200, 714*
batch files, netlist format *644*
batch files, running from Draft *149*
baud rate *11*
bidirectional pins *see also pins*
  defined *333*
bill of materials *see also Create Bill of Materials*
  include file format *448*
  including information from a text file *447*
BIOS and printing and plotting *11*
bitmap definition *357*
  graphic symbols *369*
  maximum number of bits *371*
bitmap definitions *732*
bitmap, defined *735*
BLOCK command in Draft *65-73*
  ASCII Import *72*
  Copy *74*
  Drag *67*
  Export *71*
  Fixup *67*
  Get *68*
  Import *70*
  Move *66*
  Save *69*
  Text Export *73*
block parts *see also parts*
  introduction *268*
  restrictions *293*
block symbol definitions *358-368*
  comments in *359*
BODY command in Edit Library *293-311*
  <Block> *297*
    Kind of Part *297*
    Size of Body *297*
  <Graphic> *298*
    Arc *300*
    Circle *299*
    Circle Center *299, 300*
    Delete *304*
    Erase Body *305*
    Fill *304*
    IEEE Symbol *302*
    Kind of Part *305*
    Line *298*
    Size of Body *305*
    Text *301*
  <IEEE> *306*
    Circle *307*
    Delete *310*
    Erase Body *310*
    IEEE Symbol *309*
    Kind of Part *311*
    Line *306*
    Size of Body *310*
    Text *308*
  Does Graphic Part have CONVERT? *295*
  IEEE *296*
  Is Part a GRID ARRAY? *295, 296*
  Kind of Part? *294*
  Number of Parts per Package *295*
  Place *296*
BOM files *714*
Border Text, character height *34*
border, worksheet *see worksheets*
boxes
  command line *xxxiii*
  entry *xxxiii*
  menu *xxxiii*
  prompt *xxxiii*

brackets *xxxii*
branch parameter stimulus statements in .INF files *633*
BRK files *714*
buffers
  Draft
    getting and placing objects *68*
    hierarchy *24, 75*
    macro *22, 75*
    saving objects *69*
  Edit Library
    examining buffer conditions *312*
Bulletin Board System *732*
bus mapping *171*
bus stimulus statements in .INF files *635*
bus trace statements in .INF files *627, 630*
bus vector statements in .INF files *632*
buses
  checking for overlapping or duplicate objects *199*
  combining labels *166*
  connecting to module ports *169*
  controlling stretching *153*
  dragging *153*
  drawing orthogonal *154*
  hotpoint *165*
  labeling *164, 166*
  labels *171*
  mapping *171*
  multiple labels *166*
  naming *164*
  placing entries in Draft *127*
  placing in Draft *125*
  short-cut for aligning *67*
  splitting *170*
  using labels *171*
buttons
  check box defined *735*
  defined *735*
  introduction *xxv*
  mouse *xxx*
  radio button defined *739*
  scroll *5*
buttons (continued)
  scroll button defined *739*
  selecting *xxv, xxxi*
  user defined *740*
byte, defined *735*

C

C files *506, 720*
Cadnetix netlist format *522*
CAE, defined *735*
Calay netlist format *524, 526*
Calcomp plotters *705*
CALCOMP1.DRV *706*
CALCOMP2.DRV *706*
Case netlist format *527*
cautions *xxxiii*
CBDS netlist format *529*
character height
  Comment Text *34*
  Label *34*
  Module Text *34*
  Part Field *33*
  Part Reference *33*
  Part Value *33*
  Pin Name *33*
  Pin Number *33*
  Power Text *34*
  Sheet Name *34*
  Sheet Net *34*
character in .INF files defined *611*
character strings in Symbol Description Language *380, 383*
check box, defined *735*
Check Design Integrity *203*
  Local Configuration *204*
Check Electrical Rules *405-414*
  configuration matrix *52*
  error messages *407-409*
  execution *405*
  forcing to process all sheets *414*
  Local Configuration *412*
  matrix explained *407*

Check Electrical Rules (continued)
option under Create Netlist *221*
restoring default settings *408*
type mismatches *418*
child data structure in netlist format files *661*
child instance statements in .INF files *620*
child, defined *735*
ChildPinCount function in netlist format files *677*
CHKDSK *4*
CLEANUP *see also Cleanup Schematic*
command line controls *487*
configuring Check Design Integrity *204*
Cleanup Schematic *199-202*
checking for errors *199*
execution *199*
Local Configuration *201*
reporting off-grid parts *202*
ClearSymbolicStrings function in netlist format files *677*
clock, computer *see system clock*
Color and Pen Plotter Table *25-28*
Pen speed *28*
Pen width *27*
colors, screen *25*
command in .INF files defined *611*
command line controls *483-504*
command line switches
16TO32 and 32TO16 *726*
CONVERT *724*
command reference
Draft *61-162*
Edit Library *346*
Comment Text, character height *34*
COMP16 *728*
CompareSymbol function in netlist format files *677*
Compile Library *264, 399-402, 727*
creating custom libraries *266, 267*
creating custom libraries with *399*
execution *399*
Local Configuration *400*
Compile Schematic *257-259*
execution *257*
Local Configuration *258*
Compile Simulation Specification File *472*
compiled library file *see also libraries*
.LIB extension *264*
defined *264*
compiled netlist format files *643*
compiler, netlist *see INET*
complex hierarchies *see also hierarchical*
Annotate Schematic *193*
converting to flat hierarchies *207*
creating netlists *207*
defined *735*
prerequisite for creating netlists *217*
Complex to Simple
flattening complex hierarchies *207*
prerequisite for Create Netlist *217*
COMPOSER *728, see also Compile LIbrary*
command line controls *488*
ComputerVision netlist format *530*
ConcatFile function in netlist format files *677*
CONDITIONS command
Draft *75*
Edit Library *312*
configuration, defined *735*
configurations file *723*
Configure Annotate Schematic *192-194*
Configure ASCTOVST *472*
Configure Back Annotate *196-197*
Configure Check Design Integrity *204*
Configure Check Electrical Rules *412-414*
Configure Cleanup Schematic *201-202*
Configure Compile Library *400-402*
Configure Compile Schematic *258-259*
Configure Convert Plot to IGES *423*
Configure Create Bill of Materials *446-450*
Configure Cross Reference Parts *416-420*
Configure Decompile Library *348*
Configure Decompile Schematic *250-252*
Configure Draft *58-60*
Configure Edit Library *286-288*

Configure Hierarchical Netlist Format *233*
Configure Incremental Netlist *219, 233*
Configure Library Archive *279-281, 402*
Configure List Library *274-276*
Configure Netlist Format *227-229*
Configure Netlist Linker *226*
Configure Plot Schematic *428-435*
Configure Print Schematic *440, 442*
Configure Schematic Design Tools *3-53*
Check Electrical Rules matrix *52*
Color and Pen Plotter Table *25-28*
Create Bill of Materials
Include File Combine *47*
Part Value Combine *47*
Driver Options *6-10*
Available Display Drivers *8*
Available Plotter Drivers *10*
Available Printer Drivers *9*
Driver Prefix *7*
Hierarchy Options *24*
buffer size *24*
Key Fields *37-51*
Annotate Part Value Combine *40*
Annotate Schematic *38*
Create Bill of Materials *38*
Include File Combine *47*
Create Netlist *38*
Extract Source from Schematic *38*
PLD Part Combine *48*
PLD Type Combine *48*
Netlist
Module Value Combine *46*
Part Value Combine *46*
Update Field Contents
Combine for Fields 1 through 8 *43*
Combine for Value *43*
Library Options *12-17*
Active library size *17*
Available Libraries *14*
Configured Libraries *12, 14-15, 16*
inserting a library *14*
Configure SDT (continued)
Library Options (continued)
Library Prefix *13*
Name Table Location *15*
removing a library *14*
Symbolic Data Location *15-16*
Macro Options *22-23*
Part Fields *28*
Printer/Plotter Output Options *11*
Template Table *29-36*
Worksheet Options *18-21*
Worksheet Prefix *20*
Configure Select Field View *235-237*
Configure Show Design Structure *452*
Configured Libraries *12, 14-15, 16*
connectivity database
.INF format specification *612*
character defined *611*
child instance statements *620*
command defined *611*
creating *207*
defined *735*
delimiter defined *611*
external statements *616*
file structure
branch parameter stimulus statements *633*
bus stimulus statements *635*
bus trace statements *627, 630*
bus vector statements *632*
module port, joined statements in .INF files *622*
pin statements *623*
pin stimulus statements *635*
pin trace statements *626, 628*
pin vector statements *631*
sample .INF file *638*
set parameter stimulus statements *633*
sheet net statements *624*
sheet net trace statements *626, 629*
signal stimulus statements *634*

connectivity database (continued)
file structure (continued)
signal trace statements *625, 628*
signal vector statements *631*
signal, joined statements *622*
header *613*
instance statements in .INF files *617*
interpreting *607*
joined statements *621*
link statements *614*
linked *211*
module port statements *615*
number defined *611*
overview *607-609*
parameter defined *612*
part instance statements *617*
pipe statements *637*
quoted token defined *610*
signal statements *616*
statement defined *612*
stimulus statements *633*
string defined *610*
sub-part code defined *611*
title block statements *613*
token defined *610*
trace statements *625, 628*
vector statements *631*
white space defined *610*
connectors
physical *132, 181-182*
placing, in Draft *181-182*
conventions, notation *xxxii*
CONVERT *721*
designs *726*
examples *724*
libraries *727*
sheetpath parts *728*
switches *724*
syntax *722*
convert parts *371*
Convert command in Draft *91*
defined *295*
convert parts (continued)
defining *295*
DeMorgan equivalent *371*
Convert Plot to IGES *421-423*
Local Configuration *423*
sample output *422*
converting
designs *721-725*
libraries *723, 727, 728*
worksheets *726*
coordinates
block symbol definitions *359*
displaying pointer in Draft *156*
JUMP command in Draft *106*
JUMP command in Edit Library *320*
SET command in Draft *151*
Symbol Description Language *395-396*
CopySymbol function in netlist format files *677*
Create Bill of Materials *443-450*
configuring Key Fields *445*
execution *443*
Include File Combine *47*
include file format *448*
including information from a text file *447*
Local Configuration *446*
Part Value Combine *47*
sample output *444*
Create Hierarchical Netlist *163, 231*
execution *232*
HFORM *231*
INET *231*
Local Configuration *233*
Create Netlist *163, 215-229*
execution *217*
Key Fields *38*
Local Configuration *218-229*
Configure IFORM *227-229*
Configure ILINK *226*
Configure INET *219*
reporting connected objects *220*

Create Netlist (continued)
reporting off-grid parts *220*
reporting unconnected objects *221*
CreatePartDatabase function in netlist format files *677*
Cross Reference Parts *415-420*
execution *415*
Local Configuration *416*
sample output *416*
CROSSREF *see also Cross Reference Parts*
configuring Check Design Integrity *204*
cursor, defined *735*
custom drivers, configuring *8*
custom libraries *12, see libraries*
custom netlist formats *see also netlists*
creating *641-695*
customer-contributed netlist formats *642*

D

dashed lines
as guide lines in Draft *123*
placing in Draft *139*
data bits *11*
data functions in netlist format files *660-663*
instance files *662*
data structures in netlist format files
child *661*
IFORM and HFORM *660*
net-oriented *661*
part *660*
part-oriented *662*
DateString symbol in netlist format files *666*
debugging schematics *531, 574, 603*
DECOMP *see also Decompile Library, 728*
command line controls *488*
DECOMP16 *728*
Decompile Library *264, 727*
creating custom libraries *267*
execution *347*
Local Configuration *348*
Decompile Schematic *249-252*
execution *249*
Local Configuration *250*
default, defined *736*
DELETE command in Draft *76-77*
Block *77*
Object *76*
Undo *77*
delimiter in .INF files defined *611*
design cycle, defined *736*
design environment *see also Design Management Tools*
introduction *xxv-xxviii*
Design Management Tools
Complex to Simple *207*
prerequisite for Create Netlist *217*
defined *736*
design structure
differences in types *209*
flat *209*
hierarchical *209*
pipe link commands *209*
|LINK commands *209*
designs
checking electrical rules *see Check Electrical Rules*
checking for duplicate reference designators *418*
checking for type mismatches *418*
checking for unused parts *419*
guidelines *163*
hierarchical *146*
listing of all parts *415*
reporting coordinates of parts *419*
reporting design structure *451-452*
Digital Simulation Tools *xxv*
annotating schematics for simulation *193*
creating hierarchical netlists *233*
creating netlists for *218*
Model netlist format *561*
netlist from INET *208*

digital, defined *736*
directory structure *643-644, 729*
libraries *732*
netlist format files *643*
sheetpath parts *728*
disk space requirements *4*
display drivers *see drivers*
DocumentNumber symbol in netlist format files *666*
double-click, defined *xxx*
Draft *57-162*
adding nets to sheet symbols *82*
backing up worksheets *152*
changing label size *79*
changing module port type *80*
changing objects *84-95*
changing orientation *81*
labels *79*
parts *85-92*
changing part orientations *91*
changing reference designators *86*
changing sheet symbol size *84*
changing size
labels *79*
text *78, 79*
changing style
labels *79*
module ports *80*
parts *85-92*
power objects *81*
text *78, 79*
changing worksheets *146, 147*
command line controls *489*
commands *see Draft commands*
configuring initial macros *23*
configuring locally *58-60*
connecting buses to module ports *169*
connecting power *173*
connecting signals without wires or buses *163*
controlling panning across the screen *152*
deleting nets from sheet symbols *82*
Draft (continued)
displaying convert form *91*
displaying pointer coordinates *156*
displaying text *159*
displaying worksheets with more detail *162*
editing
label names *79*
labels *79*
layout objects *95*
module ports *80*
nets on sheet symbols *83*
objects *78-95*
part fields *88*
part orientations *91*
part reference designators *85*
part values *87, 89*
power objects *81*
reference designator locations *86*
reference designators *86*
sheet symbol filenames *83*
sheet symbols *82*
stimulus objects *95*
text *78, 79*
title blocks *93-94*
trace objects *95, 140*
vector objects *95*
erasing objects *76-77*
execution *57*
exporting objects to a file *71*
exporting text to a file *73*
finding and reporting errors *406*
fixing wires and buses *67*
getting and placing objects from a buffer *68*
hierarchy buffer *75*
importing objects from a file *70*
importing text from a file *72*
isolating power *176-180*
labeling buses *164*
left mouse button *153*
loading macros automatically *23*
loading parts *98-101*

Draft (continued)
Local Configuration *58*
locating commands *61*
locating objects *96*
macros
buffer *75, 116*
calling *116*
creating *112-122*
debugging *114*
initial *115*
macro buffer *22*
nesting *114*
pause *114*
syntax *117*
text files *117*
using three-button mouse *120*
valid macro keys *112*
memory *75*
moving objects *66, 67*
moving reference designators *86*
naming nets on sheet symbols *83*
orthogonal wires and buses *67*
placing
bus entries *127*
buses *125*
connectors *181-182*
dashed lines *139*
junctions *126*
labels *128*
Layout objects *145*
module ports *131*
multiple objects *150*
no-connects *144*
pipe link commands *209*
power objects *133*
sheet symbols *135*
stimulus objects *142*
text *137*
trace objects *140*
vector objects *141*
wires *123*

Draft (continued)
power objects *173*
changing name *81*
changing type *81*
creating different *174*
isolating *176*
power supplies, creating different *175*
power, isolating *180*
quitting without saving changes *148*
repeating commands *64*
reporting design structure *451-452*
restoring deletions *77*
rotating parts *91-92*
saving objects in a buffer *69*
saving worksheets *147*
screen colors, setting *25*
setting error bell *153*
setting macro prompts *154*
showing pin numbers *155*
status information, displaying *75*
stretching buses *153*
tasks quick reference *63*
updating reference designators *189*
Draft commands *61, 76, 77-162*
AGAIN *64*
BLOCK *65-73*
ASCII Import *72*
Copy *74*
Drag *67*
Export *71*
Fixup *67*
Get *68*
Import *70*
Move *66*
Save *69*
Text Export *73*
CONDITIONS *75*
DELETE *76-77*
EDIT *37, 78-95*
Add-Net *82*
Delete *82*

Draft commands (continued)
Filename *83*
Label *79*
Label Larger *79*
Label Name *79*
Label Orientation *79*
Label Smaller *79*
Module Port *80*
Module Port Name *80*
Module Port Style *80*
Module Port Type *80*
Name *83*
Net *83*
Orientation *91*
Part *41, 85*
Part Fields *88*
Part Value *87, 89*
Power *81*
Power Name *81*
Power Orientation *81*
Power Type *81*
Reference *85*
Size *84*
FIND *96*
GET *16, 98-101*
Convert *100, 295*
Down *101*
Mirror *101*
Over *101*
Place *100*
Rotate *100*
Up *101*
HARDCOPY *102-104*
Block Print *103*
Destination *103-104*
File Mode *104*
Print Worksheet *102*
Scale *105*
Width of Paper *104*
INQUIRE *105, 406*

Draft commands (continued)
JUMP *106-108*
Reference *106*
X-Location *107*
Y-Location *108*
LIBRARY *109-110*
Browse *16, 110*
Directory *109*
MACRO *111-122*
Capture *112-115*
Delete *115*
Initialize *115*
List *115*
Read *115*
Write *116*
PLACE *123-144*
Bus *125*
Dashed Line *139*
Entry (bus) *127*
Junction *126*
Label *128*
Layout *145*
Module Port *131-132*
No-Connect *144*
Power *133-134*
Sheet *135-137*
Stimulus *142-143*
Text *137*
Trace *140*
Vector *141*
Wire *123*
quick reference *62*
QUIT *146-149*
Abandon Edits *148*
Enter Sheet *146*
Initialize *147*
Leave Sheet *146*
Run User Commands *149*
Suspend to System *148*
Update File *147*
Write to File *147*

Draft commands (continued)
REPEAT *150*
SET *151-159*
Auto Pan *152*
Backup File *152*
Drag Buses *153*
Error Bell *153*
Grid Parameters *157*
Left Button *153*
Macro Prompts *154*
Orthogonal *154*
Repeat Parameters *158*
Show Pins *155*
Title Block *155*
Visible Lettering *159*
Worksheet Size *156*
X,Y Display *156*
TAG *161*
ZOOM *162*
Center *162*
In *162*
Out *162*
Select *162*
DRAFTUSR file *149*
Driver Options *see drivers*
Driver Prefix *7*
drivers *6-10*
custom drivers, configuring *8*
Driver Options
Available Display Drivers *8*
Available Plotter Drivers *10*
Available Printer Drivers *9*
Driver Prefix *7*
supported by ESP *185*
DUMP netlist format *531*
DXF.DRV *709*

E

EDA, defined *736*
EDIF hierarchical netlist format *597*
EDIF netlist format *532*
named properties *172*
EDIT command in Draft *37, 78-95*
Add-Net *82*
Delete *82*
Filename *83*
Label *79*
Label Larger *79*
Label Name *79*
Label Orientation *79*
Label Smaller *79*
Module Port *80*
Module Port Name *80*
Module Port Style *80*
Module Port Type *80*
Name *83*
Net *83*
Orientation *91*
Part *41, 85*
Part Fields *88*
Part Value *87, 89*
Power *81*
Power Name *81*
Power Orientation *81*
Power Type *81*
Reference *85*
Size *84*
Edit File *183*
configuring an editor *183*
execution *183*
M2EDIT *183*
Edit Library *283, 345-346*
changing the definition of a part *322*
command reference *346*
configuring initial macros *23*
creating custom libraries *266*
creating parts *266*
editing parts *266*
execution *286*

Edit Library (continued)
introduction *283*
library objects *313*
loading macros automatically *23*
Local Configuration *286*
macro buffer *22*
part suffix *317*
placing objects off-grid on parts *303*
status information, displaying *312*
system memory available *312*
Edit Library commands
AGAIN *292*
BODY *293, 294, 297-311*
<Block> *297*
Kind of Part *297*
Size of Body *297*
<Graphic> *298*
Arc *300*
Circle *299*
Circle Center *299, 300*
Delete *304*
Erase Body *305*
Fill *304*
IEEE Symbol *302*
Kind of Part *305*
Line *298*
Size of Body *305*
Text *301*
<IEEE> *306*
Circle *307*
Delete *310*
Erase Body *310*
IEEE Symbol *309*
Kind of Part *311*
Line *306*
Size of Body *310*
Text *308*
Place *296*
CONDITIONS *312*
EXPORT *315*
GET PART *284, 317*
IMPORT *319*
Edit Library commands (continued)
JUMP *320-321*
X-Location *320*
Y-Location *321*
LIBRARY *322*
Browse *324*
Delete Part *325*
Prefix *326*
Update Current *284, 322*
MACRO *328*
Initial Macro *328*
NAME *329-330*
Add *330*
Delete *330*
Edit *330*
Prefix *330*
ORIGIN *331*
PIN *332*
Add *332*
Delete *332*
Move *334*
Name *332*
Pin-Number *332*
Repeat *335*
Shape *334*
Type *333*
quick reference *290*
QUIT *336-338*
Abandon Edits *338*
Initialize *337*
Suspend to System *338*
Update File *284, 336*
Write to File *284, 337*
REFERENCE *339*
repeating *292*
selecting *289*
SET *340*
Auto Pan *340*
Backup File *341*
Error Bell *341*
Left Button *342*
Macro Prompts *342*
Power Pins Visible *343*

Edit Library commands (continued)
SET (continued)
Show Body Outline *308, 343*
Visible Grid Dots *343*
tasks *291*
ZOOM *346*
Center *346*
In *346*
Out *346*
Select *346*
editors *55-185*
defined *736*
introduction *xxvi*
EEDesigner netlist format *538*
electrical rules *see Check Electrical Rules*
Encapsulated PostScript drivers *710*
EndNode function in netlist format files *677*
enter, defined *xxxi*
entry box, defined *736*
environment variables, C compiler *653*
EPSxDRV *710*
equations included in the netlist
AlteraADF netlist format *517*
IntelADF netlist format *547*
ERC *see also Check Electrical Rules*
ERC files *715*
errors *see also Check Electrical Rules*
Already at the Root Level *146*
checking for on schematics *199, 407-409*
correcting errors in Draft *105*
in multi-sheet designs *210*
messages reported by IFORM and HFORM *694-695*
preventing errors when placing parts *156*
reported by HP.DRV *708*
setting error bell in Draft *153*
setting error bell in Edit Library *341*
Tag does not exist *320*
update files *242*
ESP *xxv-xxviii*
ExceptionsFor function in netlist format files *678*
EXE files *506, 720*
ExitType symbol in netlist format files *666*
EXPORT command in Edit Library *315*
extensions
.BAK *200*
.C *506*
.EXE *506*
.INF *208, 608*
.INS *211, 662*
.INX *208, 608*
.LIB *264*
.LNF *211*
.PIP *211, 663*
.RES *211*
.SRC *264*
.TWG *452*
external statements in .INF files *616*
external views *89*
EXTRACT *457*
command line controls *490*
Programmable Logic Device Tools *456*
To PLD *456, 459*
Extract Source from Schematic
Key Fields *38*
PLD Part Combine *48*
PLD Type Combine *48*

F

Field
Key Fields *37*
Part Fields *37*
Configure Schematic Design Tools *28*
editing and moving *88*
tips *45*
Part Value *37*
Fields
Key Fields *240*
Part Fields, configuring character height *33*
FieldString1-FieldString8 symbol in netlist format files *667*

file extensions *see extensions*
file stack, rebuilding in INET 222
file structure *see design structure*
FileName symbol in netlist format files *667*
file_index type definition in netlist format files *672*
FIND command in Draft *96*
FindSymbolChar function in netlist format files *678*
FirstChild function in netlist format files *678*
FirstChildPin function in netlist format files *678*
FirstNet function in netlist format files *678*
FirstNode function in netlist format files *679*
FirstPart function in netlist format files *679*
FirstPin function in netlist format files *679*
FirstPipe function in netlist format files *679*
flat design, defined *736*
flat file structure *see design structure*
flat netlists *see also netlists*
  creating *215-229*
  creating from hierarchical designs *217*
  formats *642*
  formatting *213*
flat source files, netlist format *644*
FLDATTRB *see also Select Field View*
  command line controls *490*
FLDSTUFF *see also Update Field Contents, 457*
  command line controls *491*
format file *see netlists*
format_constant type definition in netlist format files *673*
functions in netlist format files
  AccessChild *676*
  AccessKeyWord *676*
  AccessPart *676*
  AddSignalName *676*
  AddSymbol *676*
  ChildPinCount *677*
  ClearSymbolicStrings *677*
  CompareSymbol *677*
  ConcatFile *677*
functions in netlist format files (continued)
  CopySymbol *677*
  CreatePartDatabase *677*
  EndNode *677*
  ExceptionsFor *678*
  FindSymbolChar *678*
  FirstChild *678*
  FirstChildPin *678*
  FirstNet *678*
  FirstNode *679*
  FirstPart *679*
  FirstPin *679*
  FirstPipe *679*
  general *664*
  getche *679*
  GetIndex *679*
  getnum *680*
  GetStandardSymbol *680*
  GetSymbolChar *680*
  HandleNodeName *680*
  Initialize *681*
  IsKeyWord *681*
  LoadFieldString *681*
  LoadFirstPin *681*
  LoadInstance *681*
  LoadPin *681*
  MakeInstanceFile *682*
  MakeLocalSignal *682*
  NextAccessType *682*
  NextChild *682*
  NextChildPin *682*
  NextInstance *683*
  NextKeyWord *683*
  NextNet *683*
  NextNode *683*
  NextPart *683*
  NextPin *684*
  NextPipe *684*
  OrCAD-defined *664*
  PackString *684*
  PadSpaces *685*
  PinCount *685*
  PostFile *685*

functions in netlist format files (continued)
PostProcess *685*
PreFile *685*
PreviousNode *685*
print *686*
ProcessFieldStrings *686*
putch *686*
puts *686*
PutSymbolChar *687*
RecordNode *687*
RewindInstanceFile *687*
SetAccessType *688*
SetCharSet *688*
SetFirst *688*
SetIndexByRef *688*
SetNext *689*
SetNumberWidth *689*
SetPartIndex *689*
SetPinMap *689*
SetPrevious *690*
SetSignal *690*
SetSymbol *690*
SetToIndex *690*
SetTraversal *690*
SortByNumber *691*
SwitchIsSet *691*
SymbolInCharSet *691*
SymbolLength *691*
ToUpper *691*
WriteCrlf *691*
WriteHeader *692*
WriteInteger *692*
WriteMap *692*
WriteNet *692*
WriteNetEnding *692*
WriteNetListEnd *692*
WriteStdSymbol *693*
WriteString *693*
WriteSymbol *693*
FutureNet netlist format *539*

G

GENDRIVE *185*
GET command in Draft *16, 98-101*
Convert *100*
Down *101*
Mirror *101*
Over *101*
Place *100*
Rotate *100*
Up *101*
GET PART command in Edit Library *284, 317*
getche function in netlist format files *679*
GetIndex function in netlist format files *679*
getnum function in netlist format files *680*
GetStandardSymbol function in netlist format files *680*
GetSymbolChar function in netlist format files *680*
global power *173*
graphic parts *see also parts, 293*
defining *369*
introduction *269*
grid *see also coordinates, GRIDARRAY*
dots
displaying in Draft *157*
displaying in Edit Library *343*
distance between *157, 343*
jumping in Edit Library *320*
references, ANSI *19*
setting in Draft *157*
staying on grid in Draft *157*
unit lengths in Edit Library *355*
unit size *361*
unit size in Edit Library *358*
GRIDARRAY
part definition *356*
parts *361*
ground, isolating *176-180*

H

HandleNodeName function in netlist format files *680*
HARDCOPY command in Draft *102-104*
- Block Print *103*
- Destination *103-104*
- File Mode *104*
- Print Worksheet *102*
- Scale *105*
- Width of Paper *104*

HARDCOPY.PRN *718*
HDUMP netlist format *603*
header in .INF files *613*
helpful files, viewing *185*
HFORM *see also Create Hierarchical Netlist, 214, 641, 643*
- command line controls *492*
- configuring Create Hierarchical Netlist *233*
- Create Hierarchical Netlist *231*
- data structures in netlist format files *660*
- incremental connectivity database *214*
- output from INET *208*
- required functions in netlist format files *659*
- traversal functions in netlist format files *663*

HI plotters *704*
hierarchical
- designs
  - changing parts into sheets *89*
  - converting to flat *217*
  - defined *736*
  - guidelines for creating *163*
  - inter-sheet connections *168*
  - leaving subsheets *146*
  - memory allocated to hierarchy buffer *24*
  - moving between worksheets *146*
  - numbering reference designators *193*
  - pipe link commands *209*
  - plotting sheets with multiple references *425*
  - power considerations *177-180*
  - printing sheets with multiple references *437*
  - sheet symbols *82, 135*
- file structure *see design structure*
- netlists *see also netlists*
  - creating *231*
  - formatting *214, 592-605*
  - interpreting .INF files *613*

hierarchical netlist formats *642*
hierarchical source files, netlist format *644*
Hierarchy Options *24*
- buffer size *24*

HiLo netlist format *544*
hotpoint
- buses *165*
- defined *129*

HP.DRV *708*
HPLASERx.DRV *709*

I

IBUILD *471, see also To Digital Simulation*
IEEE parts *see also parts*
- body outline *377*
- defining *373-378*
- introduction *269*
- pin placement *377*
- size limits *376*

IFORM *see also Create Netlist, 213, 641, 643*
- configuring Create Netlist *218, 227-229*
- data structures in netlist format files *660*
- intermediate netlist structure *213*
- output from ILINK *211, 213*
- overriding incremental processing *228*
- required functions in netlist format files *659*

IGES format, converting from plot file *421-423*
IGS files *715*
ILINK *see also Create Netlist, To Layout, 211*
checking for single-node nets *226*
command line controls *495*
configuring Create Netlist *218-226*
INS files *211, 212*
netlists for PC Board Layout Tools *211*
output from INET *208*
overriding incremental processing *226*
PIP files *211, 212*
RES files *211, 212*
IMPORT command in Edit Library *319*
include file format for Create Bill of Materials *448*
incremental annotation
configuration option *194*
defined *194*
incremental connectivity database *see also connectivity database, netlists*
contents *608*
defined *206, 736*
files comprising *208*
INF format *607*
incremental design *205-207*
incremental netlist process, defined *205*
incremental netlisting, defined *737*
INET *see also Create Netlist*
assigning net names to unconnected pins *220*
checking electrical rules *221*
command line controls *497*
comparing time stamp *208*
configuring Check Design Integrity *204*
configuring Create Hierarchical Netlist *233*
configuring Create Netlist *218, 219*
Create Hierarchical Netlist *231*
creates .INF and .INX files *608*
creating only reports *222*
incremental nature *210*
INET (continued)
INF files *208*
introduction *208-209*
INX files *208, 209*
netlists for simulation *208*
overriding incremental netlisting *222*
reporting off-grid parts *220*
speeding processing *222*
to HFORM *208*
to ILINK *208*
INF files *608, 715*
defined *208*
extension *208*
sample *638*
INF format specification *612*
initial macro
configuring *23*
defined *737*
Initialize function in netlist format files *681*
input pins *see also pins*
defined *333*
INQUIRE command in Draft *105*
viewing error messages *406*
INS files *211, 212, 662, 715*
data functions in netlist format files *662*
extension *211*
instance file *see INS files*
instance statements in .INF files *617*
integer_constant type definition in netlist format files *673*
IntelADF netlist format *546*
header information *546*
including equations in the netlist *547*
Intergraph netlist format *549*
intermediate netlist structure *see also connectivity database, netlists*
defined *211, 737*
INX files *608, 715*
defined *209*
extension *208*
using *210*
IsKeyWord function in netlist format files *681*

J

joined statements in .INF files *621*
JUMP command
  Draft *106-108*
  Edit Library *320*
junctions
  checking for overlapping or duplicate objects *199*
  placing in Draft *126*

K

K, defined *737*
Key Fields *37-51, see also Configure Schematic Design Tools*
  Annotate Part Value Combine *40*
  Annotate Schematic *38, 190*
  Create Bill of Materials *38, 445*
    Include File Combine *47*
    Part Value Combine *47*
  Create Netlist *38*
  Extract Source from Schematic *38*
    PLD Part Combine *48*
    PLD Type Combine *48*
  key field, defined *737*
  Netlist
    Module Value Combine *46*
    Part Value Combine *46*
  netlist formats *507*
  Update Field Contents *38*
    Combine for Fields 1 through 8 *43*
    Combine for Value *43*
keyboard commands *xxxi, 5*
KeyWord symbol in netlist format files *667*
keywords in Symbol Description Language *380, 383*
kilobyte, defined *737*

L

labels
  checking for overlapping or duplicate objects *199*
  configuring character height *34*
  connecting signals without wires *163*
  editing in Draft *79*
  placing in Draft *128*
  reporting connected *220*
LastFile symbol in netlist format files *681*
layout objects
  editing in Draft *95*
  INQUIRE command in Draft *105*
  placing in Draft *145*
LIB files *264, 715*
LIBARCH *see also Archive Library*
  command line controls *498*
LIBEDIT *see also Edit Library*
  command line controls *499*
LIBLIST *see also List Library*
  command line controls *499*
librarians *261-402*
  defined *737*
  introduction *xxvii*
libraries *12, 727*
  Active library size *17*
  adding parts *315, 319*
  bitmaps *727*
  block parts *268*
  browsing through parts *324*
  changing order *15*
  compiled library file, defined *264*
  compiling *399*
  CONDITIONS screen in Edit Library *313*
  configured *12-15, 16*
  configuring libraries to load *264*
  contents *263*
  creating *399*
  creating with a text editor *351*

libraries (continued)
custom *12, 399*
creating *265-271*
creating with Compile Library *399*
creating with string files *280, 281*
defined *737*
deleting parts *325*
design-specific *277*
editing *347*
file extensions *264*
graphic parts *269*
IEEE parts *269*
inserting a library *14*
introduction to *263*
Library Options
Active library size *17*
Available Libraries *14*
Configured Libraries *12, 14-15, 16*
inserting a library *14*
Library Prefix *13*
Name Table Location *15-16*
removing a library *14*
Symbolic Data Location *15-16*
library source file defined *264*
listing parts *265, 273, 275*
listing parts in Draft *109*
loading order *12*
objects *313*
order searched *265*
part names, duplicate *12*
parts *see parts*
parts found in each library *185*
removing a library *14*
saving *322, 336*
saving memory *277*
sheetpath designator *271*
status of current library *313*
system memory *264*
viewing parts in Draft *110*
writing to files *337*

LIBRARY command
Draft *109-110*
Browse *16, 110*
Directory *109*
Edit Library *322*
Browse *324*
Delete Part *325*
Prefix *326*
Update Current *284, 322*
library files *see libraries*
Library Options *see also libraries*
Available Libraries *14*
Configured Libraries *14*
Library Prefix *13*
library source file *see also libraries*
.SRC extension *264*
creating *351*
creating and modifying *264*
Decompile Library *347*
defined *264*
editing *266*
LibraryNameString symbol in netlist format files *667*
lines *see dashed lines, wires*
link statements in .INF files *614*
linked connectivity database *see also connectivity database, netlists*
defined *206*
INF format *607*
linker response files *644*
list box *5*
List Library *273-276*
execution *273*
Local Configuration *274*
sample report *274*
string files *276*
use *265*
LoadFieldString function in netlist format files *681*
LoadFirstPin function in netlist format files *681*

loading order, libraries *12*
LoadInstance function in netlist format files *681*
LoadPin function in netlist format files *681*
Local Configuration
  Compile Simulation Specification
local configuration, defined *737*
LocalSignal symbol in netlist format files *667*
LookupNameString symbol in netlist format files *668*

M

M2EDIT *183, 185*
  Edit File *183*
  View Reference *185*
MAC files *716*
MACRO command
  Draft *111-122*
    Capture *112-115*
    Delete *115*
    Initialize *115*
    List *115*
    Read *115*
    Write *116*
  Edit Library *328*
    Initial Macro *328*
macros
  <Ctrl><Break> *114*
  about *111*
  comments *119*
  configuring for macros *116*
  defined *111, 737*
  deleting *115*
  Draft
    buffer *75, 116*
    calling *116*
    creating *112-122*
    debugging *114*
    initial *115*
    nesting *114*
    pause *114*
macros (continued)
  Draft (continued)
    syntax *117*
    text files *117*
    using three-button mouse *120*
    valid macro keys *112*
  Edit Library *328*
    initial *328*
  efficient *122*
  enabling prompts *154*
  initial *23*
  interrupting macros *114*
  listing *115*
  loading *115*
  loading automatically *23*
  macro buffer *22*
  Macro Options *22-23*
  using *116*
main menu commands *xxxii*
MakeInstanceFile function in netlist format files *682*
MakeLocalSignal function in netlist format files *682*
MAP files *716*
match string *240, see also Update Field Contents*
mB, defined *408, 737*
megabyte, defined *737*
memory *see system memory*
  requirements *4*
Mentor netlist format *550*
mI, defined *408*
MKNETFMT.BAT syntax *646*
mO, defined *408*
module port statements *615*
  in .INF files *622*
module ports
  as connection between worksheets *181-182*
  checking connections *222, 414*
  checking for overlapping objects *199*
  configuring character height *34*
  connecting to buses *169*

module ports (continued)
connecting worksheets *168*
defined *738*
editing in Draft *80*
placing in Draft *131*
reporting unconnected *414*
unconnected, reporting *221*
Module Value *39*
ModuleName symbol in netlist format files *668*
mouse
setting left button in Draft *153*
three-button and macros *120*
moving *see mouse, keyboard commands, JUMP command*
mU, defined *408*
MultiWire netlist format *552*

N

NAME command in Edit Library *329-330*
Add *330*
Delete *330*
Edit *330*
Prefix *330*
name table, configuring location *15-16*
named properties, EDIF *172*
NC, defined *408*
NET files *716*
net, defined *738*
net-oriented data structure in netlist format files *661*
NetCode symbol in netlist format files *668*
netlist format files *see netlists, 720*
batch *644*
compiled *643*
linker response *644*
object *644*
source
flat *644*
hierarchical *644*
netlists
compiler (INET) *208*
compiling several times *210*
creating *205-210*
defined *738*
external views *89*
flat
creating *215-229*
multiple sheets *208*
one-sheet *208*
sheetpath parts *220*
uses for *215*
flat formats *507-591*
for PC boards *215, 218*
for simulation *215, 218*
format files
creating custom *641, 695*
required functions *659*
formats *505-605*
Algorex *514*
Allegro *515*
AlteraADF *516*
AppliconBRAVO *520*
AppliconLEAP *521*
Cadnetix *522*
Calay *524, 526*
Case *527*
CBDS *529*
ComputerVision *530*
creating custom *641-695*
customer-contributed formats *642*
DUMP *531*
EDIF *532*
EDIF hierarchical *597*
EEDesigner *538*
file extensions *506*
flat *507-591, 642*
FutureNet *539*
HDUMP *603*
hierarchical *592-605, 642*
HiLo *544*
IntelADF *546*
Intergraph *549*

netlists (continued)
formats (continued)
introduction *505*
Key Fields *507*
Mentor *550*
MultiWire *552*
OHDL *554*
OrCAD Digital Simulation Tools Model *561*
OrCAD Programmable Logic Design Tools *559*
OrCAD-supplied *642*
OrCAD/PCB II *557*
PADS ASCII *563, 565, 567, 568*
part and net orientations *642*
PCAD *570*
PCADnlt *572*
PDUMP *574*
RacalRedac *575*
Scicards *577*
selecting *228*
SPICE *579*
SPICE hierarchical *603*
Tango *583, 585*
Telesis *587*
types of format files *506*
Vectron *588*
WireList *590*
formatting *213*
hierarchical
complex *208*
creating *231*
formats *592-605*
formatting *214*
simple *208*
incremental design *205-207*
incremental netlist process
defined *205*
HFORM *206*
IFORM *206*
ILINK *206*
INET *206*
time stamp *206*
netlists (continued)
intermediate netlist structure *212*
defined *211*
linking *211*
overriding incremental processing *222*
selecting formats *228*
speeding processing *222*
SPICE format *212*
NetNameString symbol in netlist format files *668*
NetNumber symbol in netlist format files *668*
NetType symbol in netlist format files *669*
NextAccessType function in netlist format files *682*
NextChild function in netlist format files *682*
NextChildPin function in netlist format files *682*
NextInstance function in netlist format files *683*
NextKeyWord function in netlist format files *683*
NextNet function in netlist format files *683*
NextNode function in netlist format files *683*
NextPart function in netlist format files *683*
NextPin function in netlist format files *684*
NextPipe function in netlist format files *684*
no-connect objects
defined *144*
placing in Draft *144*
notation conventions *xxxii*
notes *xxxiii*
number in .INF files defined *611*
numeric constants in Symbol Description Language *380, 383*

O

object files, netlist format *644*
objects
in libraries *313*
overlapping or duplicate, checking for *199*
placing in Draft *123*
off-grid parts
preventing *157*
reporting *202, 220, 414*
OHDL netlist format *554*
OHDL source code *460*
open collector *see also pins*
defined *334*
open emitter *see also pins*
defined *334*
OrCAD Digital Simulation Tools Model netlist format *561*
OrCAD Programmable Logic Design Tools netlist format *559*
OrCAD-supplied netlist formats *642*
OrCAD/PCB II netlist format *557, 733*
ORCADESP.DAT file *718*
ORCADPROJ environment variable *723*
order of libraries *12*
Organization name *21*
Organization symbol in netlist format files *669*
orienting parts *see parts*
ORIGIN command in Edit Library *331*
origin on plotted schematics *35*
orthogonal wires and buses *67*
drawing *154*
orthogonal, defined *67*
output pins *see also pins*
defined *333*

P

package, defined *738*
PackString function in netlist format files *684*
PADS ASCII netlist format *563, 565, 567, 568*
PadSpaces function in netlist format files *685*
pan, defined *738*
parallel port *11*
parameter in .INF files defined *612*
parent, defined *738*
parity *11*
part data structure in netlist format files *660*
part definition *355-357*
components of *355*
construction *355*
sheetpath parts *355*
types of *355*
Part Fields *37*
abbreviations in key fields *39*
changing names of *28*
changing visibility globally *236*
Configure Schematic Design Tools *28*
configuring character height *33*
defined *738*
editing and moving *88*
tips *45*
part instance statements in .INF files *617*
part name string *373*
part reference *see reference designator*
Part Reference, character height *33*
Part Value field *37*
part values
abbreviation in key fields *39*
changing visibility globally *236*
configuring character height *33*
editing and moving *87*
part-oriented data structure in netlist format files *662*
PartIndex symbol in netlist format files *669*
PARTLIST *see also Create Bill of Materials*
command line controls *501*

PartName symbol in netlist format files *669*
parts
adding pins *332*
block *293*
body
block parts *268*
graphic parts *269*
IEEE parts *269*
body outline display *343*
body types *268*
changing kind of part *297*
checking for duplicate reference designators *418*
checking for overlapping or duplicate objects *199*
checking for type mismatches *418*
checking for unused parts *419*
components *268*
constraints *293*
convert defined *295*
convert, defining *295*
converted forms *100*
creating *284*
creating a bill of materials *443-450*
cross reference listing *415*
defined *738*
defining pins *333*
deleting pins *332*
displaying convert form *91*
drawing block parts *297*
drawing graphic parts *298-305*
drawing IEEE parts *306-311*
duplicate names *12*
editing *85, 284*
editing prefix definitions *326*
graphic *293*
size limits *351*
guidelines for creating *296*
IEEE *293, 294, 296, 361*
defining *296, 373-378*
locating *109, 110*
moving pins *334*
names *271*
parts (continued)
naming *329*
naming pins *332*
numbering pins *332*
object for unconnected pins *144*
part fields, changing visibility globally *236*
part name string *358*
part values, changing visibility globally *236*
parts per package, swapping in Draft *92*
pin names *271*
pin number
alphanumeric *270*
pin numbers *270*
pin shapes *270*
pin types *270*
pin-grid array *366*
pins *see pins*
placing on grid *118*
power objects *173, 175*
preventing off-grid *157*
reference designators *271, 339*
reference designators, changing visibility globally *236*
reporting connected *220*
reporting coordinates *419*
reporting off-grid parts *202, 220, 414*
rotating and mirroring in Draft *91*
sheetpath *271, 359*
showing pin numbers in Draft *155*
sizing bodies *296, 297*
specifying pin shapes *334*
status of the current part *313*
suffixes *317*
types *293*
updating reference designators *189*
parts per package *358*
checking for unused parts *419*
choosing *295*
specifying zero *364*
updating reference designators *190*

part_symbol type definition in netlist format files *674*
passive pins *see also pins*
defined *333*
pathnames
full *722, 723*
relative *722, 723*
PC Board Layout Tools *145, 163*
creating linked connectivity databases *206*
creating netlists for *218*
output from ILINK *211*
PCAD netlist format *570*
PCADnlt netlist format *572*
PCB II netlist format *557*
PCB, defined *738*
PCBDEV.LIB *733*
PDUMP netlist format *574*
Pen speed, configuring *28*
Pen width, configuring *27*
pin-grid array parts *295, 296, see also parts*
PIN command in Edit Library *332*
Add *332*
Delete *332*
Move *334*
Name *332*
Pin-Number *332*
Repeat *335*
Shape *334*
Type *333*
pin definition *356, see also pins-357, 362*
Pin Name, character height *33*
Pin Number, character height *33*
pin numbers, enabling/disabling
Draft *155*
pin statements in .INF files *623*
pin stimulus statements in .INF files *635*
pin trace statements in .INF files *626, 628*
pin vector statements in .INF files *631*
Pin-to-Pin distance *33*
PinCount function in netlist format files *685*
PinIndex symbol in netlist format files *669*
PinNameString symbol in netlist format files *670*
PinNumberString symbol in netlist format files *670*
pins *see also pin definition*
bidirectional *333*
displaying pins *364*
input *333*
names *271*
numbers *270*
alphanumeric *270*
object for unconnected *144*
open collector *334*
open emitter *334*
output *333*
passive *333*
pin definitions *362*
pin-grid array *366*
power *173, 333*
power objects *175*
power pin visibility *343*
reporting unconnected *414*
shapes *270*
showing pin numbers in Draft *155*
three-state *333*
types *270*
unconnected, reporting *221*
PinType symbol in netlist format files *670*
PIP files *211, 212, 663, 716*
extension *211*
pipe commands
in .INF files *208*
using in flat designs *209*
pipe file *see PIP files*
pipe file functions *663*
pipe link commands *see pipe commands, PIP files*
pipe statements in .INF files *637*
PipeLine symbol in netlist format files *670*

PLACE command in Draft *123, 145*
Bus *125*
Dashed Line *139*
Entry (bus) *127*
Junction *126*
Label *128*
Layout *145*
Module Port *131-132*
No-Connect *144*
Power *133-134*
Sheet *135-137*
Stimulus *142-143*
Text *137*
Trace *140*
Vector *141*
Wire *123*
Plot Schematic *425-435*
execution *425*
Local Configuration *428*
Plot X and Y Offset *35*
plotting to a file *430*
plotting to a printer *430*
sample output *427*
scaling output *432-433*
setting offsets *434*
suppressing title block, border, and text *426*
PLOTALL *see also Plot Schematic*
command line controls *502*
plotter drivers *10*
plotter pens, configuring *25*
plotter, defined *425*
plotters *see printing and plotting*
PLT files *716*
pointer, defined *738*
PostFile function in netlist format files *685*
PostProcess function in netlist format files *685*
PostScript plotter drivers *710*
power
connecting *173*
isolating *176-180*
objects *173*
connecting *173, 175*
defined *173*
editing in Draft *81*
global objects *173*
names *175*
placing in Draft *133*
supplies, creating different *174-175*
power pins *see also pins*
configuring visibility *343*
defined *333*
Power Text, character height *34*
PreFile function in netlist format files *685*
prefix definition *see also parts*
construction *353*
Symbol Description Language *384*
PreviousNode function in netlist format files *685*
print function in netlist format files *686*
Print Schematic *437-442*
execution *437*
Local Configuration *440*
printing to a file *441*
sample output *438*
suppressing pin numbers *441*
PRINTALL *see also Print Schematic*
command line controls *503*
Printed Circuit Board Layout Tools *xxv*
printer, defined *425*
printing and plotting *11, see also HARDCOPY command*
common sheet dimensions *31*
configuring plotter pens *25*
creating an IGES file *421-423*
Plot Schematic *425-435*
Plotter drivers *10*
plotters *11*
moving origin *35*

printing and plotting (continued)
plotting *697-711*
cabling *698*
Calcomp plotters *705*
HI plotters *704*
hints *702*
HP driver *708*
HP LaserJet driver *709*
MODE command *700*
PC/AT cabling *699*
PC/XT cabling *698, 699*
plotting
DXF driver for AutoCAD *709*
to a file *430*
to a printer *430*
problems *700*
Print Schematic *437-442*
printer drivers *9*
Printer/Plotter Output Options *11*
printers *11*
printing grid references and pin numbers *441*
printing to a file *441*
scaling output *432*
setting offsets *434*
suppressing title block, border, and text *426*
PRN files *716*
ProcessFieldStrings function in netlist format files *686*
processors *187-247*
defined *738*
introduction *xxvi*
Programmable Logic Design Tools *xxv, 455*
Compile Source *460*
netlist format *559*
To PLD
EXTRACT *456*
work screen *457*
programmable logic device, defined *738*
prompt, defined *738*
properties, named *172*
PSCRIPT.DRV *710*
PSCRIPT2.DRV *710*
putch function in netlist format files *686*
puts function in netlist format files *686*
PutSymbolChar function in netlist format files *687*
PWORD.DRV *711*

Q

quiet mode, defined *739*
QUIT command
Draft *146-149*
Abandon Edits *148*
Enter Sheet *146*
Initialize *147*
Leave Sheet *146*
Run User Commands *149*
Suspend to System *148*
Update File *147*
Write to File *147*
Edit Library *336-338*
Abandon Edits *338*
Initialize *337*
Suspend to System *338*
Update File *284, 336*
Write to File *284, 337*
quotation marks *xxxii*
quoted token in .INF files defined *610*

R

RacalRedac netlist format *575*
radio button, defined *739*
RAM requirements *4*
raster commands in printing *425*
READ.ME files, viewing *185*
RecordNode function in netlist format files *687*
REFERENCE command in Edit Library *339*
reference designators *271, 358, 360*
changing visibility globally *236*
checking for duplicates *418*
editing and moving *85*

reference designators (continued)
location of *360*
modifying *195*
updating automatically *189*
Reference field *37, see also Key Fields*
abbreviation for reference in key fields *39*
reference files *718*
viewing *185*
ReferenceString symbol in netlist format files *670*
REPEAT command in Draft *150*
reporters *403-452*
defined *739*
introduction *xxvii*
RES files *211, 212, 717*
extension *211*
resolved file *see RES files*
Revision symbol in netlist format files *671*
RewindInstanceFile function in netlist format files *687*
root directory, defined *739*
root sheet, defined *739*
Rotate command in Draft *91*
Run User Commands command in Draft *149*

S

sB, defined *408*
SCH files *717*
Schematic Design Tools *xxv*
schematic structure *see design structure*
schematics *see also Draft, Draft commands, worksheets*
defined *739*
Scicards netlist format *577*
screen colors in Draft, configuring *25*
scroll buttons *5*
defined *739*
SDL *see Symbol Description Language*
SDT.BCF file *718, 723*
SDT.CFG file *718*
Select Field View *37, 235-237*
execution *235*
Local Configuration *235*
Part Fields *236*
serial port *11*
SET command
Draft *151-159*
Auto Pan *152*
Backup File *152*
Drag Buses *153*
Error Bell *153*
Grid Parameters *157*
Left Button *153*
Macro Prompts *154*
Orthogonal *154*
Repeat Parameters *158*
Show Pins *155*
Title Block *155*
Visible Lettering *159*
Worksheet Size *156*
X,Y Display *156*
Edit Library *340*
Auto Pan *340*
Backup File *341*
Error Bell *341*
Left Button *342*
Macro Prompts *342*
Power Pins Visible *343*
Show Body Outline *308, 343*
Visible Grid Dots *343*
set parameter stimulus statements in .INF files *633*
SetAccessType function in netlist format files *688*
SetCharSet function in netlist format files *688*
SetFirst function in netlist format files *688*
SetIndexByRef function in netlist format files *688*
SetNext function in netlist format files *689*
SetNumberWidth function in netlist format files *689*

SetPartIndex function in netlist format files *689*
SetPinMap function in netlist format files *689*
SetPrevious function in netlist format files *690*
SetSignal function in netlist format files *690*
SetSymbol function in netlist format files *690*
SetToIndex function in netlist format files *690*
SetTraversal function in netlist format files *690*
sheet net statements in .INF files *624*
sheet net trace statements in .INF files *626, 629*
sheet nets
  adding to sheet symbol *135*
  configuring character height *34*
  defined *739*
  deleting *135*
  editing *136*
sheet parts
  defined *739*
  names, editing and moving *89*
sheet symbols
  adding sheet nets *135*
  changing filename *136*
  changing size *137*
  deleting sheet nets *135*
  Draft
    adding nets *82*
    changing size *84*
    deleting nets *82*
    naming nets *83*
  editing filenames *83*
  editing in Draft *82*
  editing sheet nets *136*
  naming *136*
  nets, editing in Draft *83*
  placing in Draft *135*
  viewing represented worksheet *146*
SheetNumber symbol in netlist format files *671*
sheetpath designator *271*
sheetpath parts
  Archive Parts in Schematic *281*
  converting *728*
  creating netlists *220*
  defined *740*
  part definitions *355*
  referencing *359*
sheets *see also worksheet*
  configuring border width *34*
  configuring character height of name *34*
  configuring size *21*
  defined *739*
  size
    American *29*
    built-in *30*
    International Standards Organization *29*
  units of measure *30*
SheetSize symbol in netlist format files *671*
shelling to system *see system*
shortcuts
  aligning wires and buses *67*
  changing visibility of part information globally *236*
  checking for unconnected nodes *226*
  cutting wires *118*
  debugging flat netlists *223*
  efficient macros *122*
  placing multiple objects *150*
  placing parts on grid *118*
  repeating commands in Draft *64*
  repeating commands in Edit Library *292*
  using macros *111-122*
Show Design Structure *451-452*
  execution *451*
  Local Configuration *452*
  sample output *451*
sI, defined *408*

signal statements in .INF files *616, 622, 625, 628*
signal stimulus statements in .INF files *634*
signal vector statements in .INF files *631*
SignalNameString symbol in netlist format files *671*
signals
  connecting buses to module ports *169*
  connecting with buses *166*
  connecting with labels *128, 163*
  connecting with module ports *131*
  connecting with sheet symbols *135*
  global objects *173*
  splitting multiple signals off buses *170*
SignalType symbol in netlist format files *671*
SIMPLE *see also Complex to Simple, Design Management Tools*
  command line controls *504*
simple hierarchy, defined *740*
simulation
  creating connectivity databases *206*
single-node nets, checking for *226*
sO, defined *408*
SortByNumber function in netlist format files *691*
source files, netlist format
  flat *644*
  hierarchical *644*
spacebar *xxxi*
Spacing Ratio *35*
SPICE netlist format *212, 579, 603*
  node names *580*
SRC files *264, 717*
standard_symbol type definition in netlist format files *674*
statement in .INF files defined *612*
stimulus objects
  editing in Draft *95*
  INQUIRE command in Draft *105*
  placing in Draft *142*
stimulus statements in .INF files *633*
STM files *717*
stop bits *11*
string files
  created by List Library *276*
  creating with Archive Parts in Schematic *280, 281*
  defined *281*
  source for Archive Parts in Schematic *280*
string in .INF files defined *610*
string_constant type definition in netlist format files *675*
sU, defined *408*
sub-part code in .INF files defined *611*
sub-parts
  defined *740*
subsheets
  exiting *146*
  viewing *146*
switches *see command line controls*
  16TO32 and 32TO16 *726*
  CONVERT *724*
switches for netlist format files *665*
SwitchIsSet function in netlist format files *691*
Symbol Description Language
  character strings *380, 383*
  creating a library source file with Archive Parts in Schematic *280*
  defining converts in *397*
  defining parts in *386-388*
  defining pins in *389-394*
  defining vectors in *395-396*
  keywords *380, 383*
  numeric constants *380, 383*
  prefix definition *384*
  specification *379-397*
symbol size *356, see also parts*
symbol type definition in netlist format files *675*
Symbolic Data Location *15-16*
SymbolInCharSet function in netlist format files *691*
SymbolLength function in netlist format files *691*

symbols in netlist format files
AddressLine1-AddressLine4 *666*
DateString *666*
DocumentNumber *666*
ExitType *666*
FieldString1-FieldString8 *667*
FileName *667*
KeyWord *667*
LastFile *681*
LibraryNameString *667*
LocalSignal *667*
LookupNameString *668*
ModuleName *668*
NetCode *668*
NetNameString *668*
NetNumber *668*
NetType *669*
Organization *669*
PartIndex *669*
PartName *669*
PinIndex *669*
PinNameString *670*
PinNumberString *670*
PinType *670*
PipeLine *670*
ReferenceString *670*
Revision *671*
SheetNumber *671*
SheetSize *671*
SignalNameString *671*
SignalType *671*
TimeStamp *671*
TitleString *671*
TotalSheets *672*
TypeCode *672*
syntax
16TO32 and 32TO16 *726*
CONVERT *722*
MKNETFMT.BAT *646*
syntax diagrams
how to read *379-381*
syntax, defined *740*
system
clock, importance when creating netlists *209*
memory
available in Edit Library *312*
hierarchy buffer *24*
saving *277*
running system commands from Draft *149*
suspending to from Draft *148*
suspending to from Edit Library *338*
system requirements *4*

T

TAG command
Draft *161*
Edit Library *345*
tag, defined *740*
Tango netlist format *583, 585*
Telesis netlist format *587*
Template Table *29-36*
text
configuring spacing ratio for Draft *35*
displaying in Draft *159*
editing before placing in Draft *138*
placing in Draft *137*
rotating in Draft *137*
sizing in Draft *138*
View Reference *185*
text editor
creating custom libraries *266*
Edit File *183*
text export, defined *740*
text import, defined *740*
three-state pins *see also pins*
defined *333*
time stamp
defined *671*
importance in incremental netlist process *206*
TimeStamp symbol in netlist format files *671*

title block statements in .INF files *613*
title blocks
ANSI *18*
configuring character height *34*
creating custom *155*
displaying standard on worksheets *155*
editing ANSI *94*
editing in Draft *93*
OrCAD *18*
suppressing during printing and plotting *426*
suppressing lines *26*
suppressing lines and text *27*
suppressing text *27*
types *18*
using pre-printed paper *27*
TitleString symbol in netlist format files *671*
To Digital Simulation
IBUILD *471*
Local Configuration *469*
To Layout
Execute *476*
Local Configuration *476*
To Main *479*
To PLD *455-466*
Execution *456*
EXTRACT *456, 459*
Local Configuration *457*
token in .INF files defined *610*
tool set *xxv*
defined *740*
tool, defined *740*
TotalSheets symbol in netlist format files *672*
ToUpper function in netlist format files *691*
trace objects
editing in Draft *95, 140*
INQUIRE command in Draft *105*
placing in Draft *140*
trace statements in .INF files *625, 628*
transfers *453-479*
defined *740*
introduction *xxvii*
To Digital Simulation
To Main *479*
traversal functions in netlist format files *663*
traversal_constant type definition in netlist format files *675*
TRC files *717*
TREELIST *see also Show Design Structure* command line controls *504*
troubleshooting schematics *531, 574, 603*
TTL, defined *740*
tutorial files *719, 733*
TWG files *717*
type definitions in netlist format files
access_constant *672*
file_index *672*
format_constant *673*
integer_constant *673*
part_symbol *674*
standard_symbol *674*
string_constant *675*
symbol *675*
traversal_constant *675*
user_constant *675*
variable *675*
type, defined *xxxi*
TypeCode symbol in netlist format files *672*

U

unconditional annotation
configuration option *194*
defined *194*
units of measure on sheets *30*
Update Field Contents *239-247*
creating update reports *246*
execution *243*
Key Fields *38*
Combine for Fields 1 through 8 *43*
Combine for Value *43*
Local Configuration *244, 247*
Part Fields *245*
update file *240, 245*
update file strings, length *241*
update files *719*
update file *see also Update Field Contents*
configuring *245*
creating *240*
errors *242*
update string *241*, *see also Update Field Contents*
upload, defined *740*
user buttons *xxviii*
defined *740*
introduction *xxviii*
User commands in Draft *149*
user_constant type definition in netlist format files *675*
utility programs
16TO32 and 32TO16 *726*
CONVERT *721*

V

variable type definition in netlist format files *675*
VCC *see power objects*
VDD *see power objects*
vector commands in plotting *425*
vector definition
creating *369, 375*
defined *357, 369*
graphic symbols *369*
vector definitions *732*
vector objects
editing in Draft *95*
INQUIRE command in Draft *105*
placing in Draft *141*
vector statements in .INF files *631*
vector, defined *741*
Vectron netlist format *588*
View Reference *185*
views, external *89*

W

Was/Is file *720*, *see also Back Annotate*
defined *195*
format *195*
white space in .INF files defined *610*
wildcard characters *722*
wildcard, defined *741*
WireList netlist format *590*
wires
checking for overlapping or duplicate objects *199*
connecting signals without wires *163*
drawing orthogonal *154*
labels *163*
macro for cutting *118*
placing in Draft *123*
reporting unconnected *221, 414*
short-cut for aligning *67*

Worksheet Options *18-21*
Worksheet Prefix *20*
worksheets *see also Draft*
back annotating reference designators *195*
checking for overlapping or duplicate objects *199*
cleaning up large worksheets *199*
compiling from text *249, 257*
configuring default extension *21*
configuring worksheet prefix *20*
connecting signals without wires *163*
connections between sheets *168*
default size *21*
defined *741*
document number *21*
global power *173*
inter-heet connections *168*
maximum dimensions *29*
organization name *21*
revision number *21*
setting size *156*
size *30-32*
ANSI *31*
ISO *31*
OrCAD default *32*
scale factors for Plot Schematic *432*
suppressing border during printing and plotting *426*
tagged locations *161*
title *21*
updating reference designators *190*
WriteCrLf function in netlist format files *691*
WriteHeader function in netlist format files *692*
WriteInteger function in netlist format files *692*
WriteMap function in netlist format files *692*
WriteNet function in netlist format files *692*
WriteNetEnding function in netlist format files *692*
WriteNetListEnd function in netlist format files *692*
WriteStdSymbol function in netlist format files *693*
WriteString function in netlist format files *693*
WriteSymbol function in netlist format files *693*

X, Y

X and Y Border Width *34*
XRF files *717*

Z

ZOOM command
Draft *162*
Center *162*
In *162*
Out *162*
Select *162*
Edit Library *346*
Center *346*
In *346*
Out *346*
Select *346*
zoom, defined *741*