Schematic Design Tools

Reference Guide

Printed on recycled paper
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xxiii</td>
</tr>
<tr>
<td>Tools and tool sets</td>
<td>xxiii</td>
</tr>
<tr>
<td>Editors</td>
<td>xxiv</td>
</tr>
<tr>
<td>Processors</td>
<td>xxv</td>
</tr>
<tr>
<td>Librarians</td>
<td>xxv</td>
</tr>
<tr>
<td>Reporters</td>
<td>xxv</td>
</tr>
<tr>
<td>Transfers</td>
<td>xxv</td>
</tr>
<tr>
<td>User buttons</td>
<td>xxv</td>
</tr>
<tr>
<td>Configuration screens</td>
<td>xxvii</td>
</tr>
<tr>
<td>Prefix/Wildcard</td>
<td>xxvii</td>
</tr>
<tr>
<td>List boxes</td>
<td>xxvii</td>
</tr>
<tr>
<td>Source and Destination entry boxes</td>
<td>xxvii</td>
</tr>
<tr>
<td>Mouse techniques</td>
<td>xxviii</td>
</tr>
<tr>
<td>Left and right mouse buttons</td>
<td>xxviii</td>
</tr>
<tr>
<td>Keyboard equivalents</td>
<td>xxix</td>
</tr>
<tr>
<td>&quot;Enter&quot; and &quot;Type&quot;</td>
<td>xxix</td>
</tr>
<tr>
<td>About this guide</td>
<td>xxix</td>
</tr>
<tr>
<td>Conventions</td>
<td>xxx</td>
</tr>
</tbody>
</table>

### Part I: Configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1: Configure Schematic Tools</td>
<td>1</td>
</tr>
<tr>
<td>Display the Configure Schematic Design Tools screen</td>
<td>3</td>
</tr>
<tr>
<td>Driver Options</td>
<td>5</td>
</tr>
<tr>
<td>Printer/Plotter Output Options</td>
<td>6</td>
</tr>
<tr>
<td>Library Options</td>
<td>11</td>
</tr>
<tr>
<td>Name Table Location and Symbolic Data Location</td>
<td>12</td>
</tr>
<tr>
<td>Reference library</td>
<td>15</td>
</tr>
<tr>
<td>Worksheet Options</td>
<td>19</td>
</tr>
<tr>
<td>Macro Options</td>
<td>23</td>
</tr>
<tr>
<td>Hierarchy Options</td>
<td>25</td>
</tr>
<tr>
<td>Color and Pen Plotter Table</td>
<td>26</td>
</tr>
<tr>
<td>Template Table</td>
<td>30</td>
</tr>
<tr>
<td>Key Fields</td>
<td>37</td>
</tr>
<tr>
<td>Check Electrical Rules matrix</td>
<td>51</td>
</tr>
</tbody>
</table>
# Schematic Design Tools Reference Guide

Part II: Editors ........................................................................................................ 53

Chapter 2: Draft ........................................................................................................ 55
  Execution .................................................................................................................. 55
  Local Configuration ............................................................................................... 56
  Command reference ............................................................................................... 59
    Selecting commands ............................................................................................ 59
    Locating commands ............................................................................................ 59
  AGAIN ..................................................................................................................... 62
  BLOCK ..................................................................................................................... 63
    BLOCK Move ....................................................................................................... 64
    BLOCK Drag ......................................................................................................... 65
    BLOCK Fixup ...................................................................................................... 65
    BLOCK Get ......................................................................................................... 66
    BLOCK Save ....................................................................................................... 67
    BLOCK Import ..................................................................................................... 68
    BLOCK Export .................................................................................................... 69
    BLOCK ASCII Import ....................................................................................... 70
    BLOCK Text Export ........................................................................................... 71
  CONDITIONS .......................................................................................................... 72
    Worksheet Memory Size ..................................................................................... 72
    Hierarchy Buffer ................................................................................................. 73
    Macro Buffer ...................................................................................................... 73
    Active Library .................................................................................................... 73
    On-Line Library ................................................................................................ 73
  DELETE .................................................................................................................... 74
    DELETE Object .................................................................................................... 74
    DELETE Block ..................................................................................................... 75
    DELETE Undo ...................................................................................................... 75
  EDIT .......................................................................................................................... 76
    Editing techniques ............................................................................................... 76
    Editing labels ....................................................................................................... 77
    Editing module ports .......................................................................................... 78
    Editing power objects ........................................................................................ 79
    Editing sheet symbols ....................................................................................... 80
    Editing parts ........................................................................................................ 83
    Editing the title block ....................................................................................... 90
## Contents

Chapter 2: Draft (continued)

- Editing stimulus objects ................................................. 92
- Editing trace objects ......................................................... 92
- Editing vector objects ....................................................... 92
- Editing layout objects ....................................................... 92

FIND ...................................................................................... 93

GET ....................................................................................... 95

Getting a part by entering a part suffix ................................. 96
The outline symbol .................................................................. 96
Rotating and placing parts .................................................... 97

HARDCOPY .......................................................................... 99

- HARDCOPY Destination .................................................. 100
- HARDCOPY File Mode ...................................................... 101
- HARDCOPY Make Hardcopy ............................................. 101
- HARDCOPY Width of Paper .............................................. 101

INQUIRE ............................................................................... 102

JUMP ..................................................................................... 103

- JUMP A, B, C, D, E, F, G, H Tag ....................................... 103
- JUMP Reference ............................................................... 103
- JUMP X-Location .............................................................. 104
- JUMP Y-Location .............................................................. 105

LIBRARY ............................................................................. 106

- LIBRARY Directory .......................................................... 106
- LIBRARY Browse ............................................................. 107

MACRO ............................................................................... 108

- MACRO Capture .............................................................. 109
  - Valid macro keys .......................................................... 109
  - Nesting macros ............................................................. 111
  - Pause ............................................................................ 111
  - Debugging macros ........................................................ 111
  - Initial macros ............................................................... 112
- MACRO Delete ................................................................. 112
- MACRO Initialize ............................................................. 112
- MACRO List ..................................................................... 112
- MACRO Read ................................................................. 112
- MACRO Write ................................................................. 113
## Schematic Design Tools Reference Guide

Using macros ................................................................. 113

### Chapter 2: Draft (continued)

- Calling a macro ................................................................. 113
- Macro buffer ................................................................. 113
- Macro text files ............................................................... 114
  - Macro syntax .............................................................. 114
  - Macro comments ......................................................... 116
- Middle mouse button macros ................................................ 117
  - Assignment macros ....................................................... 117
  - Individual macros ......................................................... 118
- Creating efficient macros ................................................... 119

### PLACE

- PLACE Wire ................................................................. 120
- PLACE Bus ................................................................. 122
- PLACE Junction ............................................................ 123
- PLACE Entry (Bus) ........................................................ 124
- PLACE Label ............................................................... 125
- PLACE Module Port ..................................................... 127
- PLACE Power .............................................................. 129
- PLACE Sheet ............................................................... 131
- PLACE Text ................................................................. 133
- PLACE Dashed Line ...................................................... 134
- PLACE Trace Name ....................................................... 135
- PLACE Vector .............................................................. 136
- PLACE Stimulus ........................................................... 137
- PLACE NoConnect ......................................................... 139
- PLACE Layout .............................................................. 140

### QUIT

- QUIT Enter Sheet ........................................................... 143
- QUIT Leave Sheet .......................................................... 143
- QUIT Update File .......................................................... 144
- QUIT Write to File .......................................................... 144
- QUIT Initialize .............................................................. 144
- QUIT Suspend to System ................................................ 145
- QUIT Abandon Edits ....................................................... 145
- QUIT Run User Commands ............................................. 146
## Chapter 2: Draft (continued)

- **REPEAT** ................................................................. 147
- **SET** ........................................................................... 148
  - **SET Auto Pan** ......................................................... 149
  - **SET Backup File** .................................................... 149
  - **SET Drag Buses** ..................................................... 150
  - **SET Error Bell** ....................................................... 150
  - **SET Left Button** ..................................................... 150
  - **SET Macro Prompts** ............................................... 151
  - **SET Orthogonal** ..................................................... 151
  - **SET Show Pin Numbers** .......................................... 152
  - **SET Title Block** ..................................................... 152
  - **SET Worksheet Size** .............................................. 153
  - **SET X,Y Display** .................................................... 153
  - **SET Grid Parameters** ............................................. 154
  - **SET Repeat Parameters** .......................................... 155
  - **SET Visible Lettering** ............................................ 156
- **TAG** ........................................................................... 157
- **ZOOM** ......................................................................... 158
  - **ZOOM Center** ........................................................ 158
  - **ZOOM In** .................................................................. 158
  - **ZOOM Out** ............................................................. 158
  - **ZOOM Select** .......................................................... 158

## Chapter 3: Guidelines for creating designs .............................................................. 159

- **Label names** ............................................................. 159
  - **Wire labels** ............................................................. 159
  - **Bus labels** ............................................................... 160
  - **Multiple labels on a bus** ......................................... 162
  - **Combining labels** ................................................... 162
- **Intersheet connections** ................................................. 164
- **Splitting buses** .......................................................... 166
- **Handling and isolating power** ....................................... 167
  - **Connecting power objects with different names** ........... 169
  - **Connecting power objects to a module port** ............... 170
  - **Handling power in a hierarchy** ................................ 171
- **Handling physical connectors** ....................................... 175
Chapter 9: Creating a netlist (continued)

- The .RES file .......................................................... 203
- The .PIP file .......................................................... 203
- The linked connectivity database ..................................... 203
- The .LNF file .......................................................... 203
- The flat formatter: IFORM .............................................. 204
- The hierarchical formatter: HFORM .................................. 204
- Caveats ........................................................................ 205

Chapter 10: Create Netlist .................................................. 207

- Linked format ............................................................ 207
- Creating linked and flattened netlists .............................. 208
- Execution ................................................................. 209
- Local Configuration of Create Netlist .............................. 209
- Configure INET ......................................................... 211
- Configure ILINK ......................................................... 215
- Configure IFORM ........................................................ 217

Chapter 11: Create Hierarchical Netlist ................................. 221

- Hierarchical format ...................................................... 221
- Execution ................................................................. 222
- Local Configuration of Create Hierarchical Netlist .......... 223
- Configure INET ......................................................... 223
- Configure HFORM ...................................................... 223

Chapter 12: Select Field View ............................................. 225

- Execution ................................................................. 225
- Local Configuration ..................................................... 225

Chapter 13: Update Field Contents ..................................... 229

- Before you run Update Field Contents ........................... 230
  - Configuring Key Fields ............................................. 230
  - Configuring Update Field Contents ............................ 230
  - Creating an update file ............................................ 230
- Execution ................................................................. 233
  - During the updating process ..................................... 233
  - After the updating process ....................................... 233
- Local Configuration ..................................................... 234
Chapter 17: Edit Library

About Edit Library ............................................................. 259
Bitmaps and vectors ............................................................. 260
Editing a part with Edit Library ............................................. 262
   Editing existing parts to create new parts ......................... 263
Limit on a part's complexity ................................................ 263
   Limit of total library size ............................................... 264
Execution ............................................................................. 264
Local Configuration ............................................................ 265
Command reference ............................................................. 268
   Selecting commands ....................................................... 268
AGAIN .................................................................................. 271

BODY .................................................................................. 272
   BODY Kind of Part? ......................................................... 273
   BODY Kind of Part? Block ............................................... 274
   BODY Kind of Part? Graphic ........................................... 274
   BODY Kind of Part? IEEE .............................................. 274
   BODY command reference ............................................. 276
   BODY <Block> commands .............................................. 276
      Body <Block> .............................................................. 276
      Body <Block> .............................................................. 276
   BODY <Graphic> commands ......................................... 277
      BODY <Graphic> Line .................................................. 277
      BODY <Graphic> Circle .............................................. 278
      BODY <Graphic> Arc .................................................. 279
      BODY <Graphic> Text ................................................ 280
      BODY <Graphic> IEEE Symbol ................................. 281
      BODY <Graphic> Fill .................................................. 283
      BODY <Graphic> Delete ............................................. 283
      BODY <Graphic> Erase Body ..................................... 284
      BODY <Graphic> Size of Body .................................... 284
      BODY <Graphic> Kind of Part ................................. 284
   BODY <IEEE> commands .............................................. 285
      BODY <IEEE> Line ...................................................... 285
      BODY <IEEE> Circle .................................................. 286
      BODY <IEEE> Text ...................................................... 287
Chapter 17: Edit Library (continued)

BODY <IEEE> IEEE Symbol...................................................... 288
BODY <IEEE> Delete............................................................... 289
BODY <IEEE> Erase Body......................................................... 289
BODY <IEEE> Size of Body...................................................... 289
BODY <IEEE> Kind of Part....................................................... 290

CONDITIONS.............................................................................. 291
Macro Buffer............................................................................. 292
Free System Memory................................................................. 292
Library....................................................................................... 292
Current Part............................................................................. 293

EXPORT.................................................................................... 294
GET PART................................................................................... 296
Getting a part by entering a part suffix....................................... 296

IMPORT....................................................................................... 297

JUMP......................................................................................... 298
JUMP A, B, C, D, E, F, G, H Tag............................................. 298
JUMP X-Location....................................................................... 298
JUMP Y-Location...................................................................... 299

LIBRARY.................................................................................... 300
LIBRARY Update Current......................................................... 300
LIBRARY List Directory............................................................. 301
LIBRARY Browse....................................................................... 302
LIBRARY Delete Part................................................................. 303
LIBRARY Prefix........................................................................ 304
About prefix definitions.............................................................. 304
0 through F.............................................................................. 305
Prefix and Short Prefix.............................................................. 305

MACRO..................................................................................... 306
Initial Macro............................................................................. 306

NAME....................................................................................... 307
NAME Add................................................................................. 308
NAME Delete.......................................................................... 308
NAME Edit.............................................................................. 308
NAME Prefix.......................................................................... 308

ORIGIN.................................................................................... 309
# Chapter 19: Creating a library source file with a text editor

- **Library source file** ......................................................... 327
- **Block part definitions** ....................................................... 327
- **Graphic part definitions** .................................................... 327
- **IEEE part definitions** ....................................................... 328
- **Prefix Definition** ............................................................ 328
  - Use of the prefix definition ................................................. 328
  - Constructing a prefix definition ....................................... 329
- **Part definition** ............................................................... 331
  - Three types of part definitions ......................................... 331
  - Components of a part definition ...................................... 331
- **Defining a block symbol** ............................................... 334
  - Part name string ............................................................ 335
  - Sheetpath keyword .......................................................... 335
  - Reference keyword .......................................................... 336
  - Grid unit size and parts/package ....................................... 337
  - Pin definitions ............................................................... 338
  - Pin type ............................................................................ 339
  - Selectively displaying pins .............................................. 340
  - Pin-grid array .................................................................... 342
- **Defining a graphic symbol** ............................................. 345
  - Defining a bitmap ........................................................... 345
  - Defining a vector .............................................................. 346
  - Graphic symbol considerations ....................................... 347
  - Converted form graphic symbol ...................................... 348
- **Defining an IEEE symbol** .............................................. 352
  - Part name string ............................................................. 352
  - Size and type definitions .................................................. 353
  - Pin definitions ............................................................... 353
  - Vector definitions ........................................................... 353
    - Defining a vector .......................................................... 354
    - IEEE standards ............................................................ 354
    - Pin placement .............................................................. 355
    - Building the IEEE body outline ..................................... 355
    - IEEE Vector Objects .................................................... 355
    - Placing ACTIVE_LOWs .................................................. 356
Chapter 25: Plot Schematic ................................................................. 407
  Execution ......................................................................................... 407
    Suppressing the title block, border, and text .................................. 408
    Sample output ................................................................................ 409
    Local Configuration ........................................................................ 410
Chapter 26: Print Schematic ............................................................... 417
  Execution ........................................................................................ 417
    Sample output .............................................................................. 418
    Local Configuration ....................................................................... 419
Chapter 27: Create Bill of Materials .................................................... 421
  Execution ....................................................................................... 421
    Sample output .............................................................................. 422
    Key fields .................................................................................. 423
    Local Configuration ...................................................................... 424
Chapter 28: Show Design Structure ..................................................... 429
  Execution ....................................................................................... 429
    Sample output .............................................................................. 429
    Local Configuration ...................................................................... 430
Part VI: Transfers ............................................................................. 431
Chapter 29: To PLD ........................................................................... 433
  Execution ....................................................................................... 434
    Running To PLD .......................................................................... 434
    Local Configuration of To PLD ....................................................... 435
    Configure FLDSTUFF .................................................................. 436
    Configure ANNOTATE ................................................................. 436
    Local Configuration of EXTRACT ............................................... 437
    About EXTRACT ....................................................................... 439
      Key fields ................................................................................ 439
      Unified documentation.............................................................. 439
      Make a custom symbol ............................................................ 439
      Defining the PLD's internal logic .............................................. 440
      Select a device ....................................................................... 441
      Record part type and value on the schematic ......................... 441
Chapter 30: To Digital Simulation ................................................................. 445
  Execution ................................................................................. 446
  Local Configuration of To Digital Simulation ........................................ 447
  Configure ANNOTATE ................................................................. 448
  Configure INET ........................................................................ 448
  Configure IBUILD ................................................................... 449
  Configure ASCTOVST .............................................................. 450

Chapter 31: To Layout ................................................................................. 453
  Execution ................................................................................ 454
  Local Configuration of To Layout ..................................................... 454
  Configure FLDSTUFF .................................................................. 455
  Configure ANNOTATE ................................................................. 455
  Configure INET ........................................................................ 455
  Configure ILINK ...................................................................... 455

Chapter 32: To Main .................................................................................... 457
  Execution ................................................................................ 457

Appendices .......................................................................................... 459

Appendix A: Command line controls .......................................................... 461
  Syntax .................................................................................... 461
  ANNOTATE ............................................................................. 462
  BACKANNO ........................................................................... 463
  CLEANUP .............................................................................. 463
  COMPOSER ............................................................................ 464
  CROSSREF ........................................................................... 464
  DECOMP ................................................................................ 464
  DRAFT .................................................................................. 465
  ERC ..................................................................................... 465
  EXTRACT .............................................................................. 466
  FLDATTRB ........................................................................... 466
  FLDSTUFF ............................................................................ 467
  HFORM ................................................................................ 468
  IFORM ................................................................................ 469
  ILINK .................................................................................. 470
  INET .................................................................................... 471

## Appendix A: Command line controls (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBARCH</td>
<td>472</td>
</tr>
<tr>
<td>LIBEDIT</td>
<td>473</td>
</tr>
<tr>
<td>LIBLIST</td>
<td>473</td>
</tr>
<tr>
<td>PARTLIST</td>
<td>474</td>
</tr>
<tr>
<td>PLOTALL</td>
<td>476</td>
</tr>
<tr>
<td>PRINTALL</td>
<td>477</td>
</tr>
<tr>
<td>SIMPLE</td>
<td>478</td>
</tr>
<tr>
<td>TREELIST</td>
<td>478</td>
</tr>
</tbody>
</table>

## Appendix B: Netlist formats

### Usage

- Types of netlist format files: 479
- Configuring for netlists: 481

### Flat netlists

- Example schematics: 481
- Algorex (ALGOREX.CCF): 487
- Allegro (ALLEGRO.CCF): 489
- AlteraADF (ALTERAAD.CCF): 490
- AppliconBRAVO (APPLBRAV.CCF): 494
- AppliconLEAP (APPLLEAP.CCF): 495
- Cadnetix (CADNETIX.CCF): 496
- Calay (CALAY.CCF): 498
- Calay (CALAY90.CCF): 500
- Case (CASE.CCF): 501
- CBDS (CBDS.CCF): 503
- ComputerVision (COMPVISION.CCF): 504
- DUMP (DUMP.CCF): 505
- EDIF (EDIF.CCF): 506
- EEDESIGNer (EEDESIGN.CCF): 510
- FutureNet (FUTURE.CCF): 511
- HiLo (HILO.CCF): 516
- IntelADF (INTELADF.CCF): 518
- Intergraph (INTERGRA.CCF): 521
- Mentor (MENTOR.CCF): 522
- MultiWire (MULTIWIREF.CCF): 524
- OrCAD/PCB II (PCBII.CCF): 525
Appendix B: Netlist formats (continued)

<table>
<thead>
<tr>
<th>Format</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrCAD Programmable Logic Design Tools (PLDNET.CCF)</td>
<td>527</td>
</tr>
<tr>
<td>OrCAD Digital Simulation Tools Model (VSTMODEL.CCF)</td>
<td>529</td>
</tr>
<tr>
<td>PADS ASCII (PADSASC.CCF)</td>
<td>531</td>
</tr>
<tr>
<td>PADS ASCII (PADSASC0.CCF)</td>
<td>533</td>
</tr>
<tr>
<td>PCAD (PCAD.CCF)</td>
<td>534</td>
</tr>
<tr>
<td>PCADnlt (PCADNLT.CCF)</td>
<td>536</td>
</tr>
<tr>
<td>PDUMP (PDUMP.CCF)</td>
<td>538</td>
</tr>
<tr>
<td>RacalRedac (RACALRED.CCF)</td>
<td>539</td>
</tr>
<tr>
<td>Scicards (SCICARDS.CCF)</td>
<td>541</td>
</tr>
<tr>
<td>SPICE (SPICE.CCF)</td>
<td>543</td>
</tr>
<tr>
<td>Tango (TANGO.CCF)</td>
<td>547</td>
</tr>
<tr>
<td>Telesis (TELESIS.CCF)</td>
<td>549</td>
</tr>
<tr>
<td>Vectron (VECTRION.CCF)</td>
<td>550</td>
</tr>
<tr>
<td>WireList (WIRELIST.CCF)</td>
<td>552</td>
</tr>
</tbody>
</table>

Hierarchical netlists

<table>
<thead>
<tr>
<th>Format</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Schematics</td>
<td>555</td>
</tr>
<tr>
<td>EDIF (EDIF.CCH)</td>
<td>559</td>
</tr>
<tr>
<td>HDUMP (HDUMP.CCH)</td>
<td>564</td>
</tr>
<tr>
<td>SPICE (SPICE.CCH)</td>
<td>564</td>
</tr>
</tbody>
</table>

Appendix C: Interpreting connectivity databases

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of connectivity databases</td>
<td>567</td>
</tr>
<tr>
<td>About the incremental connectivity database</td>
<td>568</td>
</tr>
<tr>
<td>About the linked connectivity database</td>
<td>569</td>
</tr>
</tbody>
</table>

Typographical conventions

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token</td>
<td>570</td>
</tr>
<tr>
<td>White space</td>
<td>570</td>
</tr>
<tr>
<td>Quoted token</td>
<td>570</td>
</tr>
<tr>
<td>String</td>
<td>570</td>
</tr>
<tr>
<td>Delimiter</td>
<td>570</td>
</tr>
<tr>
<td>Command</td>
<td>571</td>
</tr>
<tr>
<td>Character</td>
<td>571</td>
</tr>
<tr>
<td>Number</td>
<td>571</td>
</tr>
<tr>
<td>Sub-part code</td>
<td>571</td>
</tr>
<tr>
<td>Statement</td>
<td>571</td>
</tr>
</tbody>
</table>
Appendix C: Interpreting connectivity databases (continued)

Parameter.................................................................................................. 572
.INF format specification ................................................................................ 572
Sample .INF file............................................................................................ 598
Differences between .INF and .LNF files.................................................... 600

Appendix D: Creating a custom netlist format ............................................. 601
About netlist formats...................................................................................... 602
  Flat formats............................................................................................... 602
  Hierarchical formats .................................................................................. 602
  Part and net orientations .......................................................................... 602
  OrCAD-supplied formats .......................................................................... 602
  Customer-contributed formats .................................................................. 602
How to create a new format ........................................................................... 603
About format files.......................................................................................... 604
  Filenames.................................................................................................. 604
  Language.................................................................................................... 604
  Functions.................................................................................................... 605
  Standard symbols...................................................................................... 605
  User-defined symbols................................................................................ 605
Flat format....................................................................................................... 606
Hierarchical format........................................................................................ 609
Required functions........................................................................................ 610
Data functions................................................................................................ 610
  Data structures.......................................................................................... 611
  Instance files.............................................................................................. 613
Traversal functions........................................................................................ 614
Pipe file functions.......................................................................................... 614
General functions............................................................................................ 615
C-language functions...................................................................................... 616
Switches.......................................................................................................... 616
Standard symbol reference.......................................................................... 617
Type definition reference............................................................................. 623
Function reference......................................................................................... 625
Error and warning messages........................................................................... 643
Contents

Appendix E: Plotter information ................................................................. 645
  Plotter cable wiring ............................................................................. 646
  Plotter problems .................................................................................. 648
  Plotting to a printer .............................................................................. 650
  General plotter tips ............................................................................ 651
  HP plotters .......................................................................................... 652
  HI plotters ........................................................................................... 653
  Calcomp plotters ................................................................................. 654
  Notes on plotter and printer drivers ...................................................... 657
    HP.DRV .......................................................................................... 657
    HPLASERx.DRV ............................................................................ 658
    DXF.DRV ...................................................................................... 658
  PostScript plotter drivers ................................................................... 659
    Encapsulated PostScript .................................................................. 659
    Other PostScript .............................................................................. 660

Appendix F: Files and file extensions ...................................................... 661
  Design files .......................................................................................... 661
  Other files ........................................................................................... 668
    #ESP_OUT.TXT ............................................................................. 668
    HARDCOPY.PRN ........................................................................... 668
    ORCADESP.DAT ............................................................................ 668
    SDT.BCF ......................................................................................... 668
    SDT.CFG ......................................................................................... 668
  Reference files .................................................................................... 668
  Tutorial files ....................................................................................... 669
  Update file .......................................................................................... 669
  Was/Is file .......................................................................................... 670
  File extensions by tool set ................................................................... 671

Glossary ................................................................................................. 701

Index ...................................................................................................... 709
OrCAD's Schematic Design Tools 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, the tool set used to create schematic designs. For detailed information about the design environment, see the OrCAD/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 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 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 as shown in the figure on the next page.
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 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 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 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.

ESP places four user buttons inside every tool set. Chapter 4 of the OrCAD/ESP Design Environment User's Guide explains how to define a user button.
Configuration screens

Schematic Design Tools and the 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 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 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 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 (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 next category.
- Press <Enter> to select the button the pointer rests on.

"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,

  \[ v \ <\text{Esc}> \text{ means to press the escape key.} \]

  \[ v \ <\text{Ctrl}>\text{<S> means to press the control key, and while holding it down, press the "S" key.} \]

  \[ v \ <\text{S}>\text{<T> (notice the space between these two characters) means to press the "S" key and let it go, then press the "T" key.} \]

**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.

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

Part I explains how to customize Schematic Design Tools 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
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 ESP is already configured when installed, you can change the configuration whenever you want to change ESP parameters.

  The *OrCAD/ESP Design Environment User's Guide* provides detailed instructions for customizing the 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 remainder of this chapter provides detailed instructions for customizing the Schematic Design Tools 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.
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.

The menu shown at right 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, these lists can be moved up and down in a manner similar to the scrolling process used for the configuration screen.

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.

![Driver Options area of the Configure Schematic Design Tools screen.](image)

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 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 on your C: drive, the prefix is:

Driver Prefix C:\ORCADESP\DRV\n
This tells Schematic Design Tools 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.

<table>
<thead>
<tr>
<th>Available Display Drivers</th>
<th>Resolution</th>
<th>Colors</th>
<th>Adapter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 x 200</td>
<td>2</td>
<td></td>
<td>Color Graphics Adapter</td>
</tr>
<tr>
<td>640 x 200</td>
<td>16</td>
<td></td>
<td>EGA standard monitor</td>
</tr>
<tr>
<td>640 x 200</td>
<td>16</td>
<td></td>
<td>EGA Monochrome monitor</td>
</tr>
<tr>
<td>640 x 350</td>
<td>4</td>
<td></td>
<td>EGA (64K RAM)</td>
</tr>
<tr>
<td>640 x 350</td>
<td>16</td>
<td></td>
<td>EGA Enhanced monitor</td>
</tr>
</tbody>
</table>

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

△ NOTE: If a driver is not configured here, Schematic Design Tools uses the one selected during installation.
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.

![Available Printer Drivers list box.](image)

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 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 printer from the drivers displayed in figure 1-3, the following displays:

![Configured Printer Driver](image)
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.

![Available Plotter Drivers list box.](image)

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>.

\[ \text{NOTE: Only the drivers that are recognized by Schematic Design Tools appear in the list box. Custom drivers do not appear and need to be typed into the Configured Plotter Driver entry box.} \]

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 25: Plot Schematic.
Chapter 1: Configure Schematic Tools

Printer/Plotter Output Options

The Printer/Plotter Output Options area (figure 1-5) defines the ports to which your printer and plotter are connected. If you choose a serial port (COM1:, COM2:, COM3:, or COM4:), you define its baud rate, parity, number of stop bits, and number of data bits.

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.
The Library Options area (figure 1-6) defines the prefix Schematic Design Tools 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 schematic design tools load the libraries listed in the Configured Libraries list box when they run. The number of libraries loaded affects the total amount of system memory available for worksheet design. It is possible to configure Schematic Design Tools to load more libraries than can be placed in 640K system RAM. Usually, four to eight libraries are sufficient and leave enough memory for designs.

![Library Options area of the Configure Schematic Design Tools screen.](image)

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 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 (described in Chapter 2: Draft). For instructions on how to change the order of the configured libraries list, see Changing the library order in this chapter.

The Library Prefix is the disk drive or directory path where Schematic Design Tools 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 to look for libraries with the .LIB extension in the ORCADESP\SDT\LIBRARY subdirectory on the C: hard disk.

| Library Prefix | C:\ORCADESP\SDT\LIBRARY\*.LIB |
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 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. It is always stored in main memory. Its size can be configured to be 64–125K. For information about the active library, see **Active library size** in this section.

The reference library contains information about each configured library. You can configure Schematic Design Tools to store it in main memory, EMS 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** on the next page.
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, EMS memory, or on disk. If you place the name table in EMS, the increase in capacity is limited only by how much EMS memory is in your computer. EMS allows for 32 MB of memory. This will handle the 20,000 parts included with Schematic Design Tools many times over.

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

If you don't have EMS 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.

△ **NOTE:** Use Draft's CONDITIONS command to display the amount and location of memory used by the reference library.

Depending on the performance of your disk drive and your EMS implementation, you can expect the performance impacts shown in table 1-1.
## Chapter 1: Configure Schematic Tools

<table>
<thead>
<tr>
<th>Name Table Location</th>
<th>Main Memory</th>
<th>EMS Memory</th>
<th>Disk</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol Table Location</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>This is usually the most efficient configuration. Draft’s GET and LIBRARY Browse commands work fastest under this configuration.</td>
</tr>
<tr>
<td>Symbol Table Location</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Draft’s GET and LIBRARY Browse may be slightly slower than in the configuration above. You can add additional EMS memory to get more parts on line.</td>
</tr>
<tr>
<td>Symbol Table Location</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>This is slower yet, but is still tolerable. This is the best option for PCs without EMS.</td>
</tr>
<tr>
<td>Symbol Table Location</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>Performance in this configuration is degraded compared to the above configurations, but is still acceptable. It should only be used for three special cases:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Very large designs such as E-size drawings with many parts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• PCs with a small amount of EMS memory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• PCs with a small amount of available main memory. This can be caused by running multi-tasking software or a large network driver.</td>
</tr>
<tr>
<td>Symbol Table Location</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>This is the slowest configuration. It should only be used with portable computers that have 512K main memory. It is tolerable for long use only if your hard disk is fast.</td>
</tr>
</tbody>
</table>

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

The active library contains information about each part on the schematic. It always resides in main memory and can be configured to be 64–152K. 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.

The size of the active library can be between 64–152K. If your worksheet contains few parts, set the active library size to 64K. For example, if your design is a memory board with many 41256 chips or a few types of glue logic chips, the active library can be quite small. If your worksheet contains many different parts, you will have to increase the size of the active library.

\[\Delta\] **NOTE:** Use Draft's CONDITIONS command to display the amount of memory used by the active library. This will help you determine whether or not you need to increase or decrease the size of the active library.
Worksheet Options

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

![Worksheet Options](image)

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

Select any combination of the following options:

- **ANSI title block**


  The default title block is shown below.

![OrCAD Title Block](image)

**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.
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 to use ANSI Standard Y14.1-1980 grid references on worksheets. Table 1-2 shows both ANSI and common grid references.

<table>
<thead>
<tr>
<th>Sheet Size</th>
<th>ANSI References</th>
<th>Common References</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X Grid Range</td>
<td>Y Grid Range</td>
</tr>
<tr>
<td>A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>C</td>
<td>1..4</td>
<td>A..D</td>
</tr>
<tr>
<td>D</td>
<td>1..8</td>
<td>A..D</td>
</tr>
<tr>
<td>E</td>
<td>1..8</td>
<td>A..H</td>
</tr>
</tbody>
</table>

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 ESP. ESP 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:

Worksheet Prefix

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 sub-directory in the current design.

Examples

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

Worksheet Prefix A:

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

Worksheet Prefix C:\ORCAD\DESIGN5\

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

Worksheet Prefix SHEET\
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 here to the name. For example, if you enter “SHEET” while in Draft, Draft appends the extension specified here.

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 the extension specified here.

△ 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.
Chapter 1: Configure Schematic Tools

Macro Options

A macro is a series of commands that executes automatically at the touch of a single key or key combination. Using Schematic Design Tools, 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 start Draft or Edit Library.

<table>
<thead>
<tr>
<th>Macro Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Buffer Size</td>
</tr>
<tr>
<td>Draft Macro File</td>
</tr>
<tr>
<td>Draft Initial Macro</td>
</tr>
<tr>
<td>Edit Library Macro File</td>
</tr>
<tr>
<td>Edit Library Initial Macro</td>
</tr>
</tbody>
</table>

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 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
```
Schematic Design Tools Reference Guide

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 <A> in the Draft Initial Macro entry box:

Draft Initial Macro ^A

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

24
Chapter 1: Configure Schematic Tools

Hierarchy Options

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

![Hierarchy Options](image)

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.

<table>
<thead>
<tr>
<th>Hierarchy Buffer Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
</tr>
</tbody>
</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.

<table>
<thead>
<tr>
<th>Color and Pen Plotter Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Body</td>
</tr>
<tr>
<td>Pin Number</td>
</tr>
<tr>
<td>Pin Name</td>
</tr>
<tr>
<td>Part Reference</td>
</tr>
<tr>
<td>Part Value</td>
</tr>
<tr>
<td>1st Port Field</td>
</tr>
<tr>
<td>2nd Port Field</td>
</tr>
<tr>
<td>3rd Port Field</td>
</tr>
<tr>
<td>4th Port Field</td>
</tr>
<tr>
<td>5th Port Field</td>
</tr>
<tr>
<td>6th Port Field</td>
</tr>
<tr>
<td>7th Port Field</td>
</tr>
<tr>
<td>8th Port Field</td>
</tr>
<tr>
<td>Wire</td>
</tr>
<tr>
<td>Bus</td>
</tr>
<tr>
<td>Junction</td>
</tr>
<tr>
<td>Power Object</td>
</tr>
<tr>
<td>Power Text</td>
</tr>
<tr>
<td>Sheet Body</td>
</tr>
<tr>
<td>Sheet Name</td>
</tr>
<tr>
<td>Sheet Net</td>
</tr>
<tr>
<td>Module Port</td>
</tr>
<tr>
<td>Module Text</td>
</tr>
<tr>
<td>Label</td>
</tr>
<tr>
<td>Comment Text</td>
</tr>
<tr>
<td>Dashed Line</td>
</tr>
<tr>
<td>Title Block</td>
</tr>
<tr>
<td>Title Text</td>
</tr>
<tr>
<td>Command Prompt</td>
</tr>
<tr>
<td>Grid Dots</td>
</tr>
<tr>
<td>Trace Object</td>
</tr>
<tr>
<td>Test Vector Obj.</td>
</tr>
<tr>
<td>Stimulus Object</td>
</tr>
<tr>
<td>Error Object</td>
</tr>
<tr>
<td>No Connect Obj.</td>
</tr>
<tr>
<td>Layout Object</td>
</tr>
<tr>
<td>Sheet Filename</td>
</tr>
</tbody>
</table>

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

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, as shown in the figure below.

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

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.
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 \(\frac{1}{100}\) of an inch).

If buses or object fills have white spaces when plotting, reduce the pen width to correct the condition. In some cases, the correct setting for your plotter can be determined only by experimentation.

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.
Chapter 1: Configure Schematic Tools

1st Part Field through 8th Part Field

The names used to label the 1st Part Field through the 8th Part Field can be changed. Look at the Color and Pen Plotter Table, and notice that the names 1st Part Field through 8th Part Field each have a box around them. This is because you can change these names. To do this, simply click inside the box and edit the name. The new name appears in Draft when using 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 8th Part Field entry fields. If you change these headings in the Color and Pen Plotter Table, they are also changed in the 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 (American) 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.

---

**Template Table**

<table>
<thead>
<tr>
<th>Units</th>
<th>Inches</th>
<th>Millimeters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>9.700</td>
<td>248.00</td>
</tr>
<tr>
<td>Vertical</td>
<td>7.200</td>
<td>183.00</td>
</tr>
<tr>
<td>Pin-to-Pin</td>
<td>1.000</td>
<td>25.40</td>
</tr>
<tr>
<td>Pin Number</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Pin Name</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Part Reference</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Part Value</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>1st Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>2nd Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>3rd Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>4th Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>5th Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>6th Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>7th Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>8th Part Field</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Power Text</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Sheet Net</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Module Text</td>
<td>0.600</td>
<td>15.20</td>
</tr>
<tr>
<td>Label</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Comment Text</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Title Block</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Border Text</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>X Border Width</td>
<td>1.000</td>
<td>25.40</td>
</tr>
<tr>
<td>Y Border Width</td>
<td>1.000</td>
<td>25.40</td>
</tr>
<tr>
<td>Plot X Offset</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Plot Y Offset</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Roll Form Size</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Spacing Ratio</td>
<td>1.000</td>
<td>25.40</td>
</tr>
</tbody>
</table>

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

Units
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.

Draft has five worksheet sizes built in: A through E (ISO) 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.

**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 on the next page 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.

\[ \triangle \] 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.

<table>
<thead>
<tr>
<th>Sheet Size</th>
<th>Outside Border</th>
<th>Inside Border</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal</td>
<td>Vertical</td>
</tr>
<tr>
<td>A</td>
<td>11.0</td>
<td>8.5</td>
</tr>
<tr>
<td>B</td>
<td>17.0</td>
<td>11.0</td>
</tr>
<tr>
<td>C</td>
<td>22.0</td>
<td>17.0</td>
</tr>
<tr>
<td>D</td>
<td>34.0</td>
<td>22.0</td>
</tr>
<tr>
<td>E</td>
<td>44.0</td>
<td>34.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sheet Size</th>
<th>Outside Border</th>
<th>Inside Border</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal</td>
<td>Vertical</td>
</tr>
<tr>
<td>A4</td>
<td>297</td>
<td>210</td>
</tr>
<tr>
<td>A3</td>
<td>420</td>
<td>297</td>
</tr>
<tr>
<td>A2</td>
<td>594</td>
<td>420</td>
</tr>
<tr>
<td>A1</td>
<td>841</td>
<td>594</td>
</tr>
<tr>
<td>A0</td>
<td>1189</td>
<td>841</td>
</tr>
</tbody>
</table>

Table 1-4. ISO horizontal and vertical dimensions of worksheets in millimeters.
Chapter 1: Configure Schematic Tools

<table>
<thead>
<tr>
<th>Sheet Size</th>
<th>Outside Border</th>
<th>Inside Border</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal</td>
<td>Vertical</td>
</tr>
<tr>
<td>A</td>
<td>9.7</td>
<td>7.2</td>
</tr>
<tr>
<td>B</td>
<td>15.2</td>
<td>9.7</td>
</tr>
<tr>
<td>C</td>
<td>20.2</td>
<td>15.2</td>
</tr>
<tr>
<td>D</td>
<td>32.2</td>
<td>20.2</td>
</tr>
<tr>
<td>E</td>
<td>42.2</td>
<td>32.2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sheet Size</th>
<th>Outside Border</th>
<th>Inside Border</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal</td>
<td>Vertical</td>
</tr>
<tr>
<td>A</td>
<td>246.4</td>
<td>182.9</td>
</tr>
<tr>
<td>B</td>
<td>386.1</td>
<td>246.4</td>
</tr>
<tr>
<td>C</td>
<td>513.1</td>
<td>386.1</td>
</tr>
<tr>
<td>D</td>
<td>817.9</td>
<td>513.1</td>
</tr>
<tr>
<td>E</td>
<td>1071.9</td>
<td>817.9</td>
</tr>
</tbody>
</table>

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 and object scaling. The default value (0.1 inch) is a half-size template; a full-size template results when the pin-to-pin spacing is set to 0.2 inch. It also changes the distance between grid dots.

**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.
<table>
<thead>
<tr>
<th><strong>Part Value</strong></th>
<th>Height of a part’s Part Value characters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1st Part Field through 8th Part Field</strong></td>
<td>Height of the characters used to show the information contained in the part fields. Notice that the names 1st Part Field through 8th Part Field each have a box around them, 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 8th Part Field entry fields. Changing these headings in the Template Table also changes them in the Color and Pen Plotter Table.</td>
</tr>
<tr>
<td><strong>Power Text</strong></td>
<td>Height of power object characters.</td>
</tr>
<tr>
<td><strong>Sheet Name</strong></td>
<td>Height of sheet name characters.</td>
</tr>
<tr>
<td><strong>Sheet Net</strong></td>
<td>Height of sheet net name characters.</td>
</tr>
<tr>
<td><strong>Module Text</strong></td>
<td>Height of module port name characters.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Height of label characters.</td>
</tr>
<tr>
<td><strong>Comment Text</strong></td>
<td>Height of comment text characters.</td>
</tr>
<tr>
<td><strong>Title Block</strong></td>
<td>Height of title block characters.</td>
</tr>
<tr>
<td><strong>Border Text</strong></td>
<td>Height of the characters used to show text in the border. For readable results, do not make border text larger than the border width.</td>
</tr>
</tbody>
</table>
### Chapter 1: Configure Schematic Tools

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X Border Width</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Y Border Width</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **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:  
  \[ \text{Plot X Offset} = - \frac{1}{2} \text{ page width} \]  
  \[ \text{NOTE: This is a negative number.} \] |
| **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:  
  \[ \text{Plot Y Offset} = - \frac{1}{2} \text{ page height} \]  
  \[ \text{NOTE: This is a negative number.} \] |
| **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:  
  \[ \text{Spacing Ratio} = \frac{\text{Pin-to-pin spacing} \times 0.08}{\text{Height (H)} \times 0.10} \]                                           |
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.

\[
\text{Aspect Ratio} = \frac{\text{Height (H)}}{\text{Width (W)}}
\]

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 character spacing of 1.026 inch.

\[
\text{Spacing Ratio} = \frac{0.2 * 0.08}{0.156 * 0.10} = 1.026
\]

Part pin names are right-justified when spacing ratios other than the default spacing ratio are used.

**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. This produces character spacing of 0.08 inch.

\[
\text{Spacing Ratio} = \frac{0.1 * 0.08}{0.06 * 0.10} = 1.333
\]
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.

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, 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.

The other eight fields are named **1st Part Field** through **8th Part Field**.

△ **NOTE:** You can change the names used to identify **1st Part Field** through **8th Part Field**. See the section 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. 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, Draft and several other Schematic Design 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 Draft and the other 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 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 PLD uses two.

![Figure 1-16. Key Fields area of the Configure Schematic Design Tools screen.](image-url)
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 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 field.

The total length of a key field combination should not exceed 127 characters. For example, if your key field characters are V,1 the total length of the data stored in the Part Value field plus the data in the 1st Part Field, plus the comma 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.

![Diagram](image)

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 12: Select Field View.

![Diagram](image)

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 V1 into the Annotate Part Value Combine edit field. For example:

Annotate Part Value Combine V1
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 74LS08's, you place the value "MOD_1" in Part Field 1, and on the other three 74LS08's 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.](image-url)
Chapter 1: Configure Schematic Tools

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 field(s) 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 field(s) 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'`  
  - `'74LS138'`  
  - `'14 pin Inverter Package'`  
  - `'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 13: Update Field Contents*.

Combine for Value and Combine for Fields 1 through 8

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), or a digit 1-9 (to specify a part field); 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 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 13: 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 leave this field set to "V" for most netlist formats.

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**

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** 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.
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. An example of how to use this key field follows.

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 PLD

PLD Part Combine and PLD Type Combine

Use these key fields in conjunction with Extract PLD, one of the processes in the To PLD transfer to Programmable Logic Design Tools. Extract PLD 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 data is placed immediately after the word "Part:" following the pin information in the .PLD file. The PLD Type Combine data is placed immediately after the word "Type:" in the .PLD file.

Example

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

Assume the Extract PLD key fields are configured as shown below.

<table>
<thead>
<tr>
<th>Extract PLD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PLD Part Combine</td>
<td>2</td>
</tr>
<tr>
<td>PLD Type Combine</td>
<td>1</td>
</tr>
</tbody>
</table>
When you run Extract PLD 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] |
|           | { |
|           | In ->2, RESET     |
|           | In ->0, MIL & n==3 & RC   |
|           | In ->1, MIL' & n==2 & RC   |
|           | In ->n+1, n<9 & RC' & RESET' & (CIN # SET) |
```

Figure 1-21. Extract PLD output (continued on next page).
\begin{verbatim}
| 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 } |
\end{verbatim}

Figure 1-21. Extract PLD output (continued from previous page).
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).

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

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.

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.

- 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 in Draft's local configuration, see the next section of this chapter.

⚠️ NOTE: Schematic Design Tools sets the root schematic of a design to the source file specified on Draft's local configuration screen.

For information about Draft's commands, see the Command reference section in this chapter.
With the Schematic Design Tools screen displayed, select Draft. A menu displays. Select Local Configuration.

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 OrCAD/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.
**Chapter 2: Draft**

**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 path and name of the worksheet.

If you enter the path and name of a worksheet (rather than selecting it from the Files list box), the next time you display Draft’s local configuration screen the Prefix/Wildcard entry box will reflect the extension you entered in the Source entry box.

⚠️ **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.

- 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.
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.

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.

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.
<table>
<thead>
<tr>
<th>Command</th>
<th>Menu commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGAIN</td>
<td>None</td>
</tr>
<tr>
<td>BLOCK</td>
<td>Move</td>
</tr>
<tr>
<td></td>
<td>Drag</td>
</tr>
<tr>
<td></td>
<td>Fixup</td>
</tr>
<tr>
<td></td>
<td>Get</td>
</tr>
<tr>
<td></td>
<td>Save</td>
</tr>
<tr>
<td></td>
<td>Export</td>
</tr>
<tr>
<td></td>
<td>ASCII Import</td>
</tr>
<tr>
<td></td>
<td>Text Export</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>None</td>
</tr>
<tr>
<td>DELETE</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>Block</td>
</tr>
<tr>
<td></td>
<td>Undo</td>
</tr>
<tr>
<td>EDIT</td>
<td>Edit</td>
</tr>
<tr>
<td></td>
<td>Jump</td>
</tr>
<tr>
<td>FIND</td>
<td>None</td>
</tr>
<tr>
<td>GET</td>
<td>None</td>
</tr>
<tr>
<td>HARDCOPY</td>
<td>Destination</td>
</tr>
<tr>
<td></td>
<td>Make Hardcopy</td>
</tr>
<tr>
<td></td>
<td>File Mode</td>
</tr>
<tr>
<td></td>
<td>Width of Paper</td>
</tr>
<tr>
<td>INQUIRE</td>
<td>None</td>
</tr>
<tr>
<td>JUMP</td>
<td>A–H tags</td>
</tr>
<tr>
<td></td>
<td>X location</td>
</tr>
<tr>
<td></td>
<td>Y location</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>Directory</td>
</tr>
<tr>
<td></td>
<td>Browse</td>
</tr>
<tr>
<td>MACRO</td>
<td>Capture</td>
</tr>
<tr>
<td></td>
<td>Initialize</td>
</tr>
<tr>
<td></td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Write</td>
</tr>
<tr>
<td></td>
<td>List</td>
</tr>
<tr>
<td>PLACE</td>
<td>Wire</td>
</tr>
<tr>
<td></td>
<td>Module Port</td>
</tr>
<tr>
<td></td>
<td>Trace Name</td>
</tr>
<tr>
<td></td>
<td>Power</td>
</tr>
<tr>
<td></td>
<td>Vector</td>
</tr>
<tr>
<td></td>
<td>Sheet</td>
</tr>
<tr>
<td></td>
<td>Stimulus</td>
</tr>
<tr>
<td></td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>NoConnect</td>
</tr>
<tr>
<td></td>
<td>Dashed Line</td>
</tr>
<tr>
<td></td>
<td>Layout</td>
</tr>
<tr>
<td>QUIT</td>
<td>Enter Sheet</td>
</tr>
<tr>
<td></td>
<td>Write to File</td>
</tr>
<tr>
<td></td>
<td>Abandon Edits</td>
</tr>
<tr>
<td></td>
<td>Leave Sheet</td>
</tr>
<tr>
<td></td>
<td>Initialize</td>
</tr>
<tr>
<td></td>
<td>Run User</td>
</tr>
<tr>
<td></td>
<td>Update File</td>
</tr>
<tr>
<td></td>
<td>Suspend to System</td>
</tr>
<tr>
<td></td>
<td>Commands</td>
</tr>
<tr>
<td>REPEAT</td>
<td>None</td>
</tr>
<tr>
<td>SET</td>
<td>Auto Pan</td>
</tr>
<tr>
<td></td>
<td>Macro Prompts</td>
</tr>
<tr>
<td></td>
<td>X, Y Display</td>
</tr>
<tr>
<td></td>
<td>Backup File</td>
</tr>
<tr>
<td></td>
<td>Orthogonal</td>
</tr>
<tr>
<td></td>
<td>Grid Parameters</td>
</tr>
<tr>
<td></td>
<td>Drag Buses</td>
</tr>
<tr>
<td></td>
<td>Show Fins</td>
</tr>
<tr>
<td></td>
<td>Repeat Parameters</td>
</tr>
<tr>
<td></td>
<td>Error Bell</td>
</tr>
<tr>
<td></td>
<td>Title Block</td>
</tr>
<tr>
<td></td>
<td>Visible Lettering</td>
</tr>
<tr>
<td></td>
<td>Left Button</td>
</tr>
<tr>
<td></td>
<td>Worksheet size</td>
</tr>
<tr>
<td></td>
<td>TAG A–H tags</td>
</tr>
<tr>
<td>ZOOM</td>
<td>Center</td>
</tr>
<tr>
<td></td>
<td>Out</td>
</tr>
<tr>
<td></td>
<td>Select</td>
</tr>
</tbody>
</table>

Table 2-1. Draft menu commands.
<table>
<thead>
<tr>
<th>Category</th>
<th>Task</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entering and editing objects and data</td>
<td>Put parts, connections, and hierarchical sheets on your worksheet.</td>
<td>PLACE</td>
</tr>
<tr>
<td></td>
<td>Get parts from loaded libraries.</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td>Edit or change parts on the worksheet.</td>
<td>EDIT</td>
</tr>
<tr>
<td></td>
<td>Identify and change a section of the worksheet.</td>
<td>BLOCK</td>
</tr>
<tr>
<td></td>
<td>Erase objects or blocks of objects.</td>
<td>DELETE</td>
</tr>
<tr>
<td>Setting locations and conditions</td>
<td>Set the status of Draft parameters.</td>
<td>SET</td>
</tr>
<tr>
<td></td>
<td>Identify and remember locations on the worksheet.</td>
<td>TAG</td>
</tr>
<tr>
<td>Navigating on the screen</td>
<td>Move the pointer to a specified location on the worksheet.</td>
<td>JUMP</td>
</tr>
<tr>
<td></td>
<td>Locate a string of text characters.</td>
<td>FIND</td>
</tr>
<tr>
<td></td>
<td>Change the location and amount of detail you see on the screen.</td>
<td>ZOOM</td>
</tr>
<tr>
<td>Repeating repetitive or complex tasks</td>
<td>Repeat the last main menu command.</td>
<td>AGAIN</td>
</tr>
<tr>
<td></td>
<td>Define macros (recorded commands).</td>
<td>MACRO</td>
</tr>
<tr>
<td></td>
<td>Duplicate the last entered object, label, or text string.</td>
<td>REPEAT</td>
</tr>
<tr>
<td>Showing status or other information</td>
<td>Display part list directories and parts in loaded libraries.</td>
<td>LIBRARY</td>
</tr>
<tr>
<td></td>
<td>Display memory available for the worksheet, hierarchy buffer, and macro buffer.</td>
<td>CONDITIONS</td>
</tr>
<tr>
<td></td>
<td>Display text associated with worksheet objects.</td>
<td>INQUIRE</td>
</tr>
<tr>
<td>Printing your schematic</td>
<td>Send a schematic to the printer or a file.</td>
<td>HARDCOPY</td>
</tr>
<tr>
<td>Leaving the program</td>
<td>Save your worksheet, enter or leave hierarchical worksheets, leave the root sheet, or leave Draft.</td>
<td>QUIT</td>
</tr>
</tbody>
</table>

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.
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.

\[\text{△}\]

**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.

\[\Delta\]

*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
```

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. The pointer is in the same position as it was when the block of objects was exported with **BLOCK Export**. 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 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 text.

Select BLOCK ASCII Import. Draft displays:

ASCII 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.

NOTE: 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 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.

\(\Delta\) **NOTE:** Only text objects can be exported. Labels or other characters associated with other objects cannot be exported.
CONDITIONS

CONDITIONS monitors your computer's memory, and the memory available for the worksheet, hierarchy buffer, and macro buffer. When you select CONDITIONS, a status window displays (figure 2-2). The CONDITIONS status window does not respond to mouse commands. Press any key to return to the main menu level.

You can print a copy of the CONDITIONS status window by pressing <Print Screen>. Make sure the Disable <Print Screen> key function option on Draft's local configuration screen is not selected.

<table>
<thead>
<tr>
<th></th>
<th>Location</th>
<th>Allocated</th>
<th>Used</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worksheet Memory</td>
<td>Main</td>
<td>365</td>
<td>0</td>
<td>163360</td>
</tr>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchy Buffer</td>
<td>Main</td>
<td>1024</td>
<td>0</td>
<td>1024</td>
</tr>
<tr>
<td>Macro Buffer</td>
<td>Main</td>
<td>9192</td>
<td>128</td>
<td>9064</td>
</tr>
<tr>
<td>Active Library</td>
<td>Main</td>
<td>157124</td>
<td>55124</td>
<td>102000</td>
</tr>
<tr>
<td>Reference Library</td>
<td>Main</td>
<td>59526</td>
<td>20000</td>
<td>39526</td>
</tr>
<tr>
<td>Name Table</td>
<td>EMS</td>
<td>250000</td>
<td>20000</td>
<td>230000</td>
</tr>
<tr>
<td>Symbol Information</td>
<td>EMS</td>
<td>250000</td>
<td>20000</td>
<td>230000</td>
</tr>
</tbody>
</table>

Figure 2-2. CONDITIONS status window.

For each of the items listed, Draft displays the location (main memory, EMS memory, or on disk), and the amount of memory—in bytes—allocated, used, and still available. The amount of memory allocated to any given item is specified on the Configure Schematic Design Tools screen. See Chapter 1: Configure Schematic Design Tools for information on how to configure these items.

Worksheet Memory Size

This shows how much memory your worksheet uses and how much memory is available in your system. Blank worksheets use memory for border and title block information.
Chapter 2: Draft

Hierarchy Buffer

This shows the status of the hierarchy buffer. Draft uses the hierarchy buffer to keep track of sheet names when managing a hierarchical design. If the hierarchical design becomes too deep for the buffer, Draft is unable to keep track of all sheets in the hierarchy. You set the hierarchy buffer size when you configure Schematic Design Tools. For details, see Chapter 1: Configure Schematic Tools. For most applications, it is not necessary to change the size of the hierarchy buffer.

Macro Buffer

This shows the status of the macro buffer. Draft stores macros (whether created on line or loaded from a macro file) in the macro buffer. If your macros are too large, the macro buffer is unable to store them. You set the macro buffer size when you configure Schematic Design Tools. For details, see Chapter 1: Configure Schematic Tools.

Active Library

This shows the status of the active library. The active library is the temporary library Draft creates to hold both the name table and the symbol information for each part on the worksheet. Draft uses one active library.

Draft builds this library by copying information from the other libraries as a schematic is loaded or when you get a new part using Draft's GET command, and discards it when you exit Draft.

On-Line Library

This shows the status of the on-line library. The on-line library is split into two parts: the Name Table and the Symbol Table. These parts contain information about each part in every library configured to load with Draft.

The Name Table contains a list of the parts in each library. The Symbol Table contains all of the symbol information for each part in each library.

The symbolic data for on-line libraries can be placed in EMS memory or kept on disk. One advantage of using EMS memory is that GET and LIBRARY Browse commands are faster in Draft. A disadvantage is that you must have or install EMS memory.
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:

```
Delete  Find  Jump  Zoom
```

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:

```
Delete which Object?
```

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 by restoring an accidentally deleted object or block of objects.

Select **DELETE Undo**. The object(s) deleted with the last **DELETE Object** or **DELETE Block** command is restored.
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. The menu that displays depends on what 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.

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 label’s orientation. Select the desired orientation: Horizontal or Vertical.

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.
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 2-3). Select the type of module port: Input, Output, Bidirectional, or Unspecified.

**Style** Select Style to change how the module port looks (figure 2-3). 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.

<table>
<thead>
<tr>
<th>Input name</th>
<th>Input module port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output name</td>
<td>Output module port</td>
</tr>
<tr>
<td>Bidirectional</td>
<td>Bidirectional module port</td>
</tr>
<tr>
<td>Unspecified</td>
<td>Unspecified module port</td>
</tr>
</tbody>
</table>

*Figure 2-3. 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.

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.
Schematic Design Tools Reference Guide

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 | 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 2-4 shows the types of net symbols. Select the appropriate type.

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.
Chapter 2: Draft

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 2-4 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>. 
**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 2-5 illustrates a library part with its default reference designator and part value.

![Figure 2-5. CMOS 4013 library part.](image)

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

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

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.
Eight part fields may be placed on the schematic worksheet by selecting them from the menu. Use 1st Part Field through 8th Part Field 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.

**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.

**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.
Sheet Part Name

A sheet part name is a filename. Once you assign a part a sheet part name, it is no longer a part. It functions as a sheet and can be used to descend into different sheets in a hierarchy.

Select Sheet Part Name to edit, move, or change the visibility of an object's sheetpart name. When you select Sheet Part Name, Draft displays the menu shown above.

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 User's Guide.

Name

Select Name to edit a sheetpart 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. If the file is not in the current design directory, specify the full pathname.

Edit the sheetpart name using the editing techniques described at the beginning of the EDIT section. When you finish editing the sheetpart 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.

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.
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 17: **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.

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 User’s Guide*

### Edit title block

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision code</td>
<td>Add or edit the revision code (3 characters maximum)</td>
</tr>
<tr>
<td>Title of sheet</td>
<td>Add or edit a title (44 characters maximum)</td>
</tr>
<tr>
<td>Document number</td>
<td>Add or edit a document number (36 characters maximum)</td>
</tr>
<tr>
<td>Sheet number</td>
<td></td>
</tr>
<tr>
<td>Number of sheets</td>
<td></td>
</tr>
<tr>
<td>Organization name</td>
<td></td>
</tr>
<tr>
<td>1st Address Line</td>
<td></td>
</tr>
<tr>
<td>2nd Address Line</td>
<td></td>
</tr>
<tr>
<td>3rd Address Line</td>
<td></td>
</tr>
<tr>
<td>4th Address Line</td>
<td></td>
</tr>
</tbody>
</table>

**OrCAD**

3175 N.W. Aliseek Drive
Hillsboro, Oregon 97124
(503) 630-9881

**Demonstration Worksheet**

<table>
<thead>
<tr>
<th>Size</th>
<th>Document Number</th>
<th>REV</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>191-0005</td>
<td>A</td>
</tr>
</tbody>
</table>

**Date:** November 12, 1990
**Sheet:** 1 of 1

**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?”
<table>
<thead>
<tr>
<th><strong>Sheet Number</strong></th>
<th>Select <strong>Sheet Number</strong> to add or edit a sheet number (any number up to 32767). Draft displays “Sheet Number?”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Sheets</strong></td>
<td>Select <strong>Number of Sheets</strong> to add to the number of worksheets (any number up to 32767). Draft displays the prompt “Number of Sheets?”</td>
</tr>
<tr>
<td><strong>Organization Name</strong></td>
<td>Select <strong>Organization Name</strong> to add or edit an organization name (up to 44 characters). Draft displays the prompt “Organization Name?”</td>
</tr>
<tr>
<td><strong>Address Lines</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

**NOTE:** If you are using an ANSI1 title block (defined on the Configure Schematic Design Tools screen), two additional areas are included in the title block: FSCM NO 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.

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 a layout directive, place the pointer where the layout symbol connects to the wire or pin on which it was placed. Select Edit. The prompt “Layout Directive?” displays followed by the current layout directive.

Edit the layout directive using the editing techniques described at the beginning of the EDIT section. When you finish editing, press <Enter>. The edited name displays on the worksheet.
FIND locates a string of text characters any-where 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.

\[\text{NOTES: FIND works only in the worksheet you are editing.}\]

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

<table>
<thead>
<tr>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select GET. Draft displays &quot;Get ?&quot;</td>
<td>Select GET. Draft displays &quot;Get ?&quot;</td>
</tr>
<tr>
<td>Enter the desired part name <em>exactly</em> 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.</td>
<td>Press &lt;Enter&gt;. 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, and then press &lt;Enter&gt;. After you select a library, Draft displays a menu listing the library parts. Move the highlight to the part name you want, then press &lt;Enter&gt; to retrieve the part. Draft puts the part's outline on the screen.</td>
</tr>
</tbody>
</table>

Once the part outline displays, you need to place it on the worksheet. For information on rotating and placing parts on the worksheet, see Rotating and placing parts in this section.
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 14: 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 2-6). 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.

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.

\[\triangle\] **NOTE:** The Convert command only displays for TTL parts.

---

**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 27). Each time you select Rotate, the part rotates an additional 90°.

**Convert**

This command only displays for TTL parts.

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.
Normal Select Normal to rotate an object to its original position, as shown in the library (figure 2-7). This command also returns mirrored or converted parts to their original form.

Up Select Up to turn an object 90° counterclockwise (figure 2-7). This is equivalent to rotating it once from its normal position.

Over Select Over to turn an object 180° counterclockwise (figure 2-7). This is equivalent to rotating it twice from its normal position.

Down Select Down to turn an object 270° counterclockwise (figure 2-7). This is equivalent to rotating it three times from its normal position.

Mirror Select Mirror to get a mirror image of an object (figure 2-7).
HARDCOPY uses fast draft printing mode to send a worksheet to the printer or to a file. HARDCOPY does not produce scaled or compressed output.

HARDCOPY is not used to output to plotters. To plot a schematic, use the Plot Schematic reporter described in Chapter 25: Plot Schematic. To print a schematic in scaled mode, use the Use Scale Factor option, as described in the Local Configuration section of Chapter 19: Plot Schematic.

To make a fast draft print, HARDCOPY assumes a working resolution of 100 units per inch. Therefore, if you are using a printer with 100 dpi (dots per inch) resolution, your HARDCOPY print is at full scale.

For example, there are four Hewlett-Packard LaserJet printer resolutions available with Schematic Design Tools (75, 100, 150, and 300 dpi). If your printer's resolution is 300 dpi, your print will be 100:300, or \( \frac{1}{3} \) the actual size. Similarly, if you are using a Toshiba printer with a resolution of 180 dpi, your print will be 100:180, or \( \frac{5}{6} \) the actual size.

To print a schematic, select HARDCOPY from the main menu. Draft displays the menu shown at right.

\[ \text{HARDCOPY} \]

\begin{center}
\begin{tabular}{|l|}
\hline
Hardcopy Destination \\
\hline
File Mode \\
\hline
Make Hardcopy Width of Paper \\
\hline
\end{tabular}
\end{center}

\[ \text{NOTES: Some versions of DOS and BIOS do not support ports COM3: and COM4:. If your printer is connected to one of these serial ports and HARDCOPY does not print, try changing to a different serial port or to a parallel port.} \]

For printing the entire design, use the Print Schematic reporter (described in Chapter 20: Print Schematic) or Plot Schematic reporter (described in Chapter 19: Plot Schematic).

For most display drivers, you can also use the Print Screen (<Shift><Print Screen>) key to send the area of the worksheet displayed on the screen to the printer.
### 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 the menu shown above.

<table>
<thead>
<tr>
<th>Destination</th>
<th>LPT:</th>
<th>File</th>
</tr>
</thead>
</table>

**LPT:**
Select **LPT:** to send your worksheet to the printer port specified on the Configure Schematic Design Tools screen.

**File**
Select **File** to send the worksheet to a file. **Draft** displays the prompt "Destination of Hardcopy?" followed by the default filename HARDCOPY.PRN.

Change the default filename using the editing techniques described at the beginning of the EDIT section. When you finish editing, 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 return to the main menu level, press <Esc>.

Files created this way can later be sent to the printer using the DOS COPY command. For example, if you created a printer file called MY.PRN, you can print it with the following DOS command:

```
COPY MY.PRN prn: /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.
File Mode determines whether subsequent HARDCOPY commands append or replace the contents of any existing hardcopy file.

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.

Make Hardcopy prints a hardcopy of the worksheet displayed on the screen. It either sends it to your printer or to a file, depending on the setting of File Mode.

To send a copy of your worksheet to your printer, first make sure your printer has power and is on line. Then select **Make Hardcopy**. Draft displays:

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

After a few seconds, the worksheet starts printing.

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

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

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

**Wide**
Select Wide if you have wide paper (13 inches wide).
INQUIRE displays text associated with stimulus, trace, vector, layout, 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 and edit 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 (greater than 80 characters), scroll the text left and right using the <←> or <→> 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.
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.

<table>
<thead>
<tr>
<th>Jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Tag</td>
</tr>
<tr>
<td>B Tag</td>
</tr>
<tr>
<td>C Tag</td>
</tr>
<tr>
<td>D Tag</td>
</tr>
<tr>
<td>E Tag</td>
</tr>
<tr>
<td>F Tag</td>
</tr>
<tr>
<td>G Tag</td>
</tr>
<tr>
<td>H Tag</td>
</tr>
<tr>
<td>Reference</td>
</tr>
<tr>
<td>X-Location</td>
</tr>
<tr>
<td>Y-Location</td>
</tr>
</tbody>
</table>

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.

\[ \text{NOTE: The error message "Tag does not exist" displays if the tag you select has not been set.} \]

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 2-1 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.
JUMP X-Location

The X-Location command moves the pointer a specific distance along the X-axis. Each step represents \( \frac{1}{10} \) (0.1) inch on the worksheet if SET Grid Parameters Stay On Grid is turned on. Otherwise it is \( \frac{1}{100} \) (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 \( \frac{1}{40} \) (0.1) inch on the worksheet if the SET Grid Parameters Stay On Grid command is turned on; otherwise it is \( \frac{1}{100} \) (0.01) inch.

The 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

<table>
<thead>
<tr>
<th>Library</th>
<th>Directory</th>
<th>Browse</th>
</tr>
</thead>
</table>

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, printed, 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.

Which Library?
- PCBDEV.LIB
- TTL.LIB
- DEVICE.LIB

To?
- Screen
- Printer
- File

Select Screen. Draft displays the library directory on the screen.

Select Printer. Draft prints the library directory on the printer connected to the printer port specified on the Configure Schematic Design Tools screen.

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.

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 17: Edit Library for details.
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 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 entered keystrokes as you select commands in Draft.

* Write a macro script in a text editor.

To create or change macros, select the MACRO command from the main menu level menu. The MACRO menu at right appears.
MACRO Capture

To create a macro, select Capture. Draft displays "Capture macro?"

Press the key or keys to use to call the macro. The key(s)
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.
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>`.

<table>
<thead>
<tr>
<th>Alt + Ctrl+ Shift+ Key</th>
<th>Alt+ Ctrl+ Shift+ Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F1</td>
</tr>
<tr>
<td>B</td>
<td>F2</td>
</tr>
<tr>
<td>C</td>
<td>F3</td>
</tr>
<tr>
<td>D</td>
<td>F4</td>
</tr>
<tr>
<td>E</td>
<td>F5</td>
</tr>
<tr>
<td>F</td>
<td>F6</td>
</tr>
<tr>
<td>G</td>
<td>F7</td>
</tr>
<tr>
<td>H</td>
<td>F8</td>
</tr>
<tr>
<td>I</td>
<td>F9</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>N</td>
<td>5</td>
</tr>
<tr>
<td>O</td>
<td>6</td>
</tr>
<tr>
<td>P</td>
<td>7</td>
</tr>
<tr>
<td>Q</td>
<td>8</td>
</tr>
<tr>
<td>R</td>
<td>9</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>^</td>
</tr>
<tr>
<td>U</td>
<td>=</td>
</tr>
<tr>
<td>V</td>
<td>-</td>
</tr>
<tr>
<td>W</td>
<td>\</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Y</td>
<td>Right</td>
</tr>
<tr>
<td>Z</td>
<td>Pgup</td>
</tr>
<tr>
<td>Tab</td>
<td>Pgdn</td>
</tr>
</tbody>
</table>

*Table 2-4. Valid key combinations for macros.*
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.

**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 "Erase All Macros?"

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

**MACRO List**

MACRO List lists all the key combinations assigned to macros. To display the macro list, select List.

**MACRO Read**

MACRO Read loads a macro file into an area of memory called the *macro buffer*, where they 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 the prompt "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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Translation</th>
<th>Name</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>\</td>
<td>Backspace</td>
<td>[RUBOUT]</td>
</tr>
<tr>
<td>Ctrl</td>
<td>^</td>
<td>Escape</td>
<td>[ESC]</td>
</tr>
<tr>
<td>Shift</td>
<td>SHIFT</td>
<td>Home</td>
<td>[HOME]</td>
</tr>
<tr>
<td>Ctrl-Home</td>
<td>[MACROBREAK]</td>
<td>End</td>
<td>[END]</td>
</tr>
<tr>
<td>Enter</td>
<td>[ENTER]</td>
<td>Shift-Tab</td>
<td>[BACKTAB]</td>
</tr>
<tr>
<td>Up</td>
<td>[U]</td>
<td>Tab</td>
<td>[^]</td>
</tr>
<tr>
<td>Down</td>
<td>[D]</td>
<td>Insert</td>
<td>[INS]</td>
</tr>
<tr>
<td>Left</td>
<td>[L]</td>
<td>Delete</td>
<td>[DEL]</td>
</tr>
<tr>
<td>Right</td>
<td>[R]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 OrCAD 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.
Chapter 2: Draft

- Do not use curly braces ({} or {}) 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 remarks are also useful for quick reference. Follow these guidelines to add and preserve remarks in macro files.

- Place remarks before or after the macro, or in both positions. You may also insert paragraphs between macros. The remarks in these macros are shown in bold:

```plaintext
PLACE Junction (^P)=pjp()
(^P)=pjp() PLACE Junction
PLACE Junction (^P)=pjp() for Supersquig
project
```

- Do not use the left curly brace ( {) in remarks.

- Add remarks, macros, or edit existing remarks or macros with a text editor. Use the editor to alphabetize the list by remark or by macro name. Save the file in text-only format.

- Make a backup 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:

```plaintext
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 remarks to your macro files.
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")=mrrputjunct.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){}\}

Once you set up the macro files, you assign macros to the middle button using keyboard commands. For example:

\{(\b)=mrrmacro_b.mac(enter){}\}
\{(\d)=mrrmacro_d.mac(enter){}\}
\{(\g)=mrrmacro_g.mac(enter){}\}
\{(\j)=mrrmacro_j.mac(enter){}\}
\{(\k)=mrrmacro_k.mac(enter){}\}
\{(\s)=mrrmacro_s.mac(enter){}\}
\{(\w)=mrrmacro_w.mac(enter){}\}
\{(\z)=mrrmacro_z.mac(enter){}\}
**Individual macros**

PLACE Bus

```plaintext
{mmm}=pb{}
Saved in macro_b.mac
```

DELETE Block Begin

```plaintext
{mmm}=dbb(macbrok)eb{}
Saved in macro_d.mac
```

BLOCK Drag Begin

```plaintext
{mmm}=bdb(macbrok)e{macbrok}p{}
Saved in macro_g.mac
```

PLACE Junction Place

```plaintext
{mmm}=pjp{esc}{}
Saved in macro_j.mac
```

BLOCK Move Begin

```plaintext
{mmm}=bmb(macbrok)e{macbrok}p{}
Saved in macro_k.mac
```

BLOCK Save Begin

```plaintext
{mmm}=bsb(macbrok)e{}g{}
Saved in macro_s.mac
```

PLACE Wire Begin

```plaintext
{mmm}=pwb{}
Saved in macro_w.mac
```

ZOOM Out and Back

```plaintext
{mmm}=zo(macbrok)zi{}
Saved in macro_z.mac
```
Chapter 2: Draft
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 m~cros 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:
{Fl}={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 2-8
compares the
execution times
ofF1 and F2
Execution Time
with Macro
Prompts set On
Figure 2-8. Macro execution times for key
and Off.
and mouse macros.

119


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

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 2-9). You may also move to the end of the wire (point B in figure 2-9) 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 2-9. Drawing a wire.](image)

Continue drawing the wire. To end the wire, select either **End** or **New**.

![Diagram](image)

△ **NOTE:** See the Schematic Design Tools 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 the 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:

![Diagram of PLACE Bus commands]

To draw a bus, place the pointer at the worksheet location where you want the bus to start. Select **Begin**. **Draft** displays:

![Diagram of BEGIN command line]

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**.

⚠️ **NOTE:** Buses with module ports attached to them are automatically labeled with the same name as the module port, and therefore do not need an extra label.

**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).
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 \ commands 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. 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.

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 from **Horizontal** to **Vertical** or vice versa.

**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 characters with the alphanumeric keys. When you finish editing the label's value, press <Enter>. 
Select **Larger** to make a label larger. Select **Larger** as many times as you like until the label is large enough.

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 2-10 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 2-11 shows incorrect label positions. None of the label hotpoints are next to the wire.

---

**Figure 2-10. Correct label positions.**

**Figure 2-11. Incorrect label positions.**

The right-pointing arrows indicate label hotpoints.
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.

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.

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 2-12 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:

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.

△ **NOTE:** An unspecified module port must be specified when isolated power is transferred between worksheets. For more information, see Chapter 10: Create Netlist.

The figure below shows the four types of module ports with their default styles.

![Module Port Styles](image)

---

△ **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 10: 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 10: Create Netlist for information on handling isolated power such as in battery backup and other applications.
Figure 2-13 shows the four kinds of power objects and their orientations.

<table>
<thead>
<tr>
<th></th>
<th>Top</th>
<th>Bottom</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>VCC</td>
<td><img src="circle.png" alt="image" /></td>
<td>VCC0-</td>
<td>-VCC</td>
</tr>
<tr>
<td>Arrow</td>
<td>VCC</td>
<td><img src="arrow.png" alt="image" /></td>
<td>VCC+</td>
<td>-VCC</td>
</tr>
<tr>
<td>Bar</td>
<td>VCC</td>
<td><img src="bar.png" alt="image" /></td>
<td>VCC-</td>
<td>-VCC</td>
</tr>
<tr>
<td>Wave</td>
<td>VCC</td>
<td><img src="wave.png" alt="image" /></td>
<td>VCC+</td>
<td>-VCC</td>
</tr>
</tbody>
</table>

*Figure 2-13. 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 User's Guide.

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>**.

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. 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 change the orientation of the text from Horizontal to Vertical or vice versa.

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. 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, select **Begin**. Then move the pointer to draw the line. End the line using **End** and change direction using **Begin**. Use **New** to end the current line, and begin a new line at a different location with **Begin**.
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 described previously. When you finish editing the trace name, press <Enter>.

**Trace name format**
A trace may be placed on either a net 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. 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
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

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.

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>.

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.
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 |

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 or bus. Just move the pointer on the wire to place the stimulus.

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>.

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:0
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, 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 used to give a layout directive to PC Board Layout Tools. The layout directive specifies the routing conditions to be used with the net on which the layout object is placed. To keep the design from being cluttered, the layout directive is not visible on the screen, but the symbol is. To view the contents of the layout directive, use the INQUIRE command.

To place a layout object on the worksheet, select Layout. Draft displays the prompt: "Layout Directive?" Enter the layout directive. Layout directives and their format are described below.

Draft displays:

| Place | Value | Find | Jump | Zoom |

The Place and Value commands are described at the end of this section.

Layout directive format

The layout directive consists of a series of characteristics that describe the conditions the net is to have when routing in OrCAD's Release IV PC Board Layout Tools. There are six net characteristics. If any of the six are not specified, the net characteristics not specified default to the net conditions specified in the PC Board Layout Tools configuration. At least one net characteristic must be present and up to all six may be present. A minimum of one space is required between layout directives.

The six conditions are: width, isolation, via, layers, pattern, and strategy. A net characteristic is followed by net conditions enclosed in a pair of parenthesis ( ). The general format for a layout directive is:

NetCharacteristic(NetConditions)

When you have more than one layout directive, use a space to separate layout directives.

△ NOTE: Use the abbreviations: **in** to specify inches, and **mm** to specify millimeters. **in** and **mm** should not be followed by a period.
The layout directives are:

- **Width**(trackwidth units) where trackwidth is a decimal number and units is either in or mm.
  
  **Examples**
  
  Width(.010in)
  Width(.25mm)

- **Isolation**(metaltometal units, viatovia units) where metaltometal and viatovia are decimal numbers and units is either in or mm.

  **Examples**
  
  Isolation(.025in,.050in)
  Isolation(.60in,1.2mm)

- **Via**(type, diameter units, drillsize units) where type is one of Through, Buried, or Blind; diameter and drillsize are decimal numbers; and units is either in or mm.

  **Examples**
  
  Via(Through,.035in,.022in)
  Via(Buried,.80mm,.50mm)

- **Layer**(first,second) where first and second are unsigned whole numbers.

  **Examples**
  
  Layer(1,4)
  Layer(2,3)

- **Pattern**(type) where type is one of Tree, Chain, or Comb.

  **Example**
  
  Pattern(Tree)

- **Strategy**(type) where type is one of Normal, Flexible, Extensive, NoVia, 90Degree, or Power.

  **Example**
  
  Strategy(Extensive)
NOTE: For an explanation of the various conditions and their effects in routing, see the PC Board Layout Tools Reference Guide.

**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 layout directive. If you don’t want to place another layout object, press <Esc> to return to the main menu level.

NOTE: If you always stay on grid, the layout symbol is always correctly positioned to connect to a wire or bus. Just move the pointer on the wire to place the layout symbol.

**Value** Select **Value** to edit the layout directive. When you select Value, the prompt “Layout Directive?” displays followed by the current value. Edit the value using the editing techniques described previously. When you finish editing the label’s value, press <Enter>.  

142
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 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:

Enter Leave Find Jump Zoom

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 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 clear the worksheet.

With a blank worksheet, Draft displays “Load File?” 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>>>.”

Press <Esc> to return to the main menu level.
Chapter 2: Draft

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 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

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  

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.

![Logic and Display Circuit](image)

*Figure 2-14. 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.

\(\triangle\) **NOTE:** Turning off Backup File can be dangerous. If your file should accidentally be damaged or erased, you will be unable to recover it.

149
### 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, 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 Pin Numbers**

When **Show Pin Numbers** 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 Pin Numbers**, 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).
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.

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.

<table>
<thead>
<tr>
<th>Zoom scale</th>
<th>Grid dot spacing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>¼ of an X or Y unit</td>
</tr>
<tr>
<td>2</td>
<td>⅛ of an X or Y unit</td>
</tr>
<tr>
<td>5</td>
<td>⅛ of an X or Y unit</td>
</tr>
<tr>
<td>10</td>
<td>1 X or Y unit</td>
</tr>
<tr>
<td>20</td>
<td>2 X or Y units</td>
</tr>
</tbody>
</table>

After you select Visible Grid Dots, you select either Yes or No. Yes displays grid dots; No does not.
Chapter 2: Draft

<table>
<thead>
<tr>
<th>SET Repeat Parameters</th>
<th>Repeat Parameters are used to determine how the REPEAT command works.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft displays the menu shown above.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set Repeat Parameters</th>
<th>X Repeat Step</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Y Repeat Step</td>
<td>+ 1</td>
</tr>
<tr>
<td></td>
<td>Label Repeat Delta</td>
<td>+ 1</td>
</tr>
<tr>
<td></td>
<td>Auto Increment Place</td>
<td>NO</td>
</tr>
</tbody>
</table>

**X Repeat Step**

X Repeat Step determines the number of unit steps in the X direction the object being repeated is offset from the original 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 $\frac{1}{10}$ of an X unit when SET Stay on Grid is turned on and $\frac{1}{100}$ 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 original 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 $\frac{1}{10}$ of a Y unit when SET Stay on Grid is turned on and $\frac{1}{100}$ 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 Repeat Delta**

Label Repeat 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 Repeat 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 Place**

When Auto Increment Place 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 Repeat Delta command.

When you select Auto Increment Place, 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.

A table is presented showing the various items and their default visibility:

<table>
<thead>
<tr>
<th>Item</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Field</td>
<td>YES</td>
</tr>
<tr>
<td>Pin Number</td>
<td>NO</td>
</tr>
<tr>
<td>Pin Name</td>
<td>NO</td>
</tr>
<tr>
<td>Label</td>
<td>NO</td>
</tr>
<tr>
<td>Text</td>
<td>YES</td>
</tr>
<tr>
<td>Module Port</td>
<td>NO</td>
</tr>
<tr>
<td>Power Value</td>
<td>YES</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>YES</td>
</tr>
<tr>
<td>Sheet Net</td>
<td>NO</td>
</tr>
<tr>
<td>Title Block</td>
<td>YES</td>
</tr>
</tbody>
</table>

Visible lettering for Scale 2
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 \( \frac{1}{2} \) of scale 1.
- **Scale 5** The third most detailed zoom scale, representing \( \frac{1}{5} \) of scale 1.
- **Scale 10** The fourth most detailed zoom scale, representing \( \frac{1}{10} \) of scale 1.
- **Scale 20** The least detailed zoom scale, representing \( \frac{1}{20} \) of scale 1.

To change the zoom scale, select ZOOM. Draft displays the menu shown at right.

<table>
<thead>
<tr>
<th>Zoom (present scale: 2)</th>
<th>Center (1)</th>
<th>In (1)</th>
<th>Out (2)</th>
<th>Select</th>
</tr>
</thead>
</table>

**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.

158
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.

**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.
NOTE: 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.

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. y must be greater than or equal to x. 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.

A bus label may be placed anywhere on the bus, as long as the label's hotpoint is touching the bus. 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:

BUSNAME\[x\]

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.

![Diagram of a labeled bus](image)

**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 CS\neg (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\$ placed next to it, and that label MEM11 has label W\E\ next to it. On the other side of the figure, C\$ and W\E\ are labels that have been placed on pins 8 and 10 of U2 and U3.

This example shows how to connect signals MEM10 and MEM11 to U2, by labeling them as C\$ and W\E\, respectively. In the case of U3, C\$ and W\E\ signals are connected to the 2114 device without being physically connected to the bus.
For Create Netlist and Create Hierarchical Netlist to list inter-worksheet 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.

[ADD[0..15] → ADD[0..15]]
[OUT[0..15] → ADD[0..15]]
[POEPT[0..15] → CNTL[0..15]]

Figure 3-5. Connecting buses to module ports.
Chapter 3: Guidelines for creating designs

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.
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.

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.
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 build the connectivity database 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 3-9 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.
In the example in figure 3-9, 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 library 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.

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 3-10. 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 net list.

Connecting power objects to a module port

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 3-11 shows three examples of power objects connected 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 3-12 through 3-14 show this approach to a battery backup application.

This design is a three-sheet hierarchy. The root sheet, shown in figure 3-12, 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 VOO 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 3-13) to the VDD pins of both devices.

Figure 3-14 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.
In the CMOS MEMORY sheet (figure 3-14), 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 3-13. 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.
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.
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 3-15 shows an example.

Make large connectors with alphanumeric pin names using a block symbol with zero parts per package.

Figure 3-16 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.
Figure 3-16. Source file for an IBM 62-pin edge connector.
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 *ORCAD/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 screen for the configured text editor displays.
View Reference runs a text editor in a reference material directory provided by OrCAD. This directory contains supplemental "read me" files of product information. 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
- List of parts found in each library

When you receive your design environment software 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 OrCAD/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. Select Execute. The screen for the configured text editor displays. Use the text editor to open and read the reference file of your choice.
**PART III: PROCESSORS**

**Schematic Design Tools** 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:** *Creating a netlist* gives an overall description of incremental netlisting.

**Chapter 10:** *Create Netlist* describes how to create a linked and flattened netlist.

**Chapter 11:** *Create Hierarchical Netlist* describes how to create a hierarchically formatted netlist.

**Chapter 12:** *Select Field View* describes how to use *Select Field View* to change visible attributes of specified fields on a worksheet.

**Chapter 13:** *Update Field Contents* describes how to use *Update Field Contents* to load user-defined information into the fields of parts on 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 packages. 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:

```
Annotate Schematic
Part Value Combine
```

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.
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).

![Configure Annotate Schematic](image)

**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.
Chapter 6: Annotate Schematic

Processing Options

You may select any combination of 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 numeric component of "?". This option resets all number to "?".

☐ 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.

☐ 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.
Select one of the following options:

- Incremental annotation (only update reference designators shown as ?)
  Updates only the reference designators that have a "?". When new parts are placed on a sheet, the reference is not assigned a numeric value, but rather is given the unassigned designation of "?". 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
  Causes Annotate Schematic to continue running when it encounters warnings, instead of halting.
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.
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).

![Configure Back Annotate](image)

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:

- **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.

**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
   
   Causes Back Annotate to continue running when it encounters warnings, instead of halting.
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 the schematic is “cleaned up,” the Schematic Design Tools screen displays.

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 this the menu that displays.

Select Configure CLEANUP. A configuration screen displays (figure 8-1).

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 output of Cleanup Schematic is placed. This entry is optional.
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.

- **Repeat **CLEANUP** if sheet is too large to complete in one pass**
  
  Causes **Cleanup Schematic** to repeat until the whole worksheet is completed.

- **Ignore warnings**
  
  Causes **Cleanup Schematic** to continue running when it encounters warnings, instead of halting in the middle of execution.
Traditionally, EDA tools exchanged design information using netlists and translators. Netlists and translators are limited, though, because they contain only a small part of the information in the complete design database.

With the support of the ESP design environment, OrCAD tools now relate in a new and powerful way—by exchanging design information primarily through the design database itself, rather than using netlists and translators.

To exchange information with tools created by third-party vendors with their own proprietary input formats, Schematic Design Tools can, as before, 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. This means less time waiting for the tools—time that can be spent designing.

The incremental netlist process is composed 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 (C, Pascal, or Modula-2 for instance). As in the process of creating a computer program, the schematic (source) files are compiled into an intermediate form, and are then linked together 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 processed by a compiler (called 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 other 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 2.00 and Spice).

△ NOTE: Schematic Design Tools includes a programming language that you use to design your own netlist formats, along with source code for the formats already supported.

Creating a netlist

Figure 9-1 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 or formatted or both to produce flat or hierarchical netlists for use by other tools.
**OrCAD's Digital Simulation Tools** uses the incremental connectivity database directly for input. **OrCAD's PC Board Layout Tools** uses the linked version of the connectivity database. It does not use the formatting process.

![Diagram](image)

**Figure 9-1.** 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 **OrCAD 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 9-2.

Figure 9-2. 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 re-compiled. 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.

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 via the \PLACE Sheet command, 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

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.
The linker: ILINK

ILINK performs one of two processes:

- It creates the intermediate netlist structure. This consists of the .INS, .RES, and .PIP files. These files are used by IFORM.
- It creates the linked connectivity database. This file has an extension of .LNF and is used by PC Board Layout Tools.

Figure 9-3 shows these processes.

![Diagram of ILINK processes]

Figure 9-3. ILINK produces the intermediate netlist structure or the linked connectivity database.

△ 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. Other tool sets (such as PC Board Layout Tools) require a linked connectivity database for all of the design. In such instances, use ILINK to create a .LNF file.
### Intermediate netlist structure

The intermediate netlist structure consists of three binary files: the .INS, .RES, and .PIP files. 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 the parts in the .INF file.

#### 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 command is the |SPICE command used by the SPICE netlist formats.

### The linked connectivity database

OrCAD's PC Board Layout Tools uses the linked connectivity database (.LNF) directly.

#### The .LNF file
The .LNF file is a giant version of all the .INF files. It has a .LNF extension because a .INF already exists for the root sheet. Producing the .LNF file from ILINK is discussed in Chapter 10: Create Netlist.
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 9-4 shows the IFORM process.

![Diagram of IFORM process]

**Figure 9-4. IFORM produces a flattened netlist.**

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 9-5 shows the HFORM process.

![Diagram of HFORM process]

**Figure 9-5. HFORM produces a hierarchical netlist.**
Caveats

As previously mentioned, INET is incremental. It compiles one sheet at a time. This can lead to what appear to be anomalies, but aren't.

For instance, say you have a multi-sheet 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? Since 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, set configuration to force INET to compile all files. Chapter 10: Create Netlist and Chapter 11: Create Hierarchical Netlist discuss this.
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-fac ing 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.

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 consists of three processes to create linked and flattened netlists. 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

For example, if the design is called root and the format is contained in my_format, the following processes run:

1. INET root.sch
2. ILINK root.inf
3. IFORM root my_format

The 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.
Chapter 10: Create Netlist

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 11: 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 to 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.

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.
△ NOTE: If you are creating a netlist for either Digital Simulation Tools or PC Board Layout Tools, use the transfer buttons To Digital Simulation and To Layout, because they are already properly set up to perform the appropriate netlist processes when you transfer to the respective tool.

To duplicate the process of creating an incremental connectivity database to use with Digital Simulation Tools, turn both ILINK and IFORM off.

To duplicate the process of creating a linked connectivity database for PC Board Layout Tools, turn IFORM off.
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 10-1).

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.

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.
<table>
<thead>
<tr>
<th>Processing Options</th>
<th>You may select any combination of the following options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiet mode</td>
<td>Turns quiet mode on.</td>
</tr>
<tr>
<td>Descend into sheet path parts</td>
<td>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 not recommended for use except for FPGA, ASIC, or other designs that require a complex hierarchy.</td>
</tr>
<tr>
<td>Assign a net name to all pins</td>
<td>Tells INET to assign a net name to all pins, including unconnected ones.</td>
</tr>
<tr>
<td>Report off-grid parts</td>
<td>Tells INET to check the worksheet for parts, sheets, labels, module ports, and power objects that are off-grid.</td>
</tr>
<tr>
<td>Report all connected labels and parts</td>
<td>Tells INET to report all connected labels and module ports.</td>
</tr>
<tr>
<td>Report all unconnected wires, pins, module ports</td>
<td>Tells INET to report all unconnected wires, pins, and module ports.</td>
</tr>
<tr>
<td>Run ERC on all sheets processed</td>
<td>Runs an electrical rules check on all sheets that INET processes. This is the same process provided by the Check Electrical Rules tool.</td>
</tr>
</tbody>
</table>
Chapter 10: Create Netlist

- 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 all module port connections

Tells INET to check all module ports for correctness after tests and processing are completed.

- 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.

- **Ignore warnings**

  Causes INET to continue running when it encounters warnings, instead of halting in the middle of execution.

  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 10-2).

![Configure Netlist Linker](image)

**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.
**Processing Options**

Select any combination of the following:

- **Quiet mode**
  
  Turns quiet mode on.

- **Produce a linked connectivity database**
  
  Tells ILINK to produce a linked connectivity database (.LNF). This text file is read by PC Board Layout Tools. If a destination file is not supplied, the output is placed in a file with the same name as the source, except with a .LNF extension.

  A **Destination** may be supplied to override the default output file. This file may have any valid pathname.

  If this option is not selected, ILINK produces the intermediate netlist structure files (.INS, .RES, and .PIP). These binary files are used by IFORM.

- **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**
  
  Reports all nodes that have only a single pin. This report is used to identify nodes that still need to be connected in the design.

- **Ignore warnings**
  
  Causes ILINK to continue running when it encounters warnings, instead of halting.
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 10-3).

**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.

**Destination 1 & 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 *.CCF*, 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 database shows the current format is up to date.

- **Ignore warnings**
  
  Causes IFORM to continue running when it encounters warnings, instead of halting.
Chapter 10: Create Netlist

Format Specific Options

Schematic Design Tools 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 custom netlist formats explains this process.
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 10: Create Netlist.

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 done 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" viewpoint 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

Since there are various final formats for netlists, a method for rapidly producing different formats is required. Further, it would be nice if you could produce the desired format without too much effort. As a result, HFORM is actually an interpreter that uses a format specification file (my_format in item 2 above) to produce the final netlist.

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.
Chapter 11: Create Hierarchical Netlist

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.

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 prompts:

<table>
<thead>
<tr>
<th>Select Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure INET</td>
</tr>
<tr>
<td>Configure HFORM</td>
</tr>
<tr>
<td>INET on</td>
</tr>
<tr>
<td>HFORM on</td>
</tr>
</tbody>
</table>

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 10: Create Netlist.

Configure HFORM

HFORM functions exactly as IFORM, except it produces a hierarchical netlist. For information about HFORM, see Configure IFORM in Chapter 10: Create Netlist, replacing references to IFORM with HFORM.
Select Field View scan 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 12-1).

![Configure Select Field View](image)

Figure 12-1. 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:

- 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

Select one of the following options to define which visible attribute is to be changed:

- Reference
- Part Value
- Part Field 1
- Part Field 2
- Part Field 3
- Part Field 4
- Part Field 5
- Part Field 6
- Part Field 7
- Part Field 8

If desired, select the following:

- Quiet mode
  Turns quiet mode on.

Select an option to set the field’s visibility attribute:

- Set specified field to visible
- 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**
  
  Causes *Set Field View* to continue running when it encounters warnings, instead of halting.
Update Field Contents searches for and replaces text in part fields on schematics. Figure 13-1 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(s) 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 13-1. The Update Field Contents process.
Before you run Update Field Contents, be sure you:

- Specify which part field(s) 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 field(s) 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.
Chapter 13: Update Field Contents

- 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 13-2 shows a typical update file with match strings in the left column and their corresponding update strings in the right column. For example, 741s00 is a match string; its corresponding update string is 14DIP300.

<table>
<thead>
<tr>
<th>741s00</th>
<th>14DIP300</th>
</tr>
</thead>
<tbody>
<tr>
<td>741s138</td>
<td>16DIP300</td>
</tr>
<tr>
<td>741s163</td>
<td>16DIP300</td>
</tr>
<tr>
<td>8259a</td>
<td>28DIP600</td>
</tr>
</tbody>
</table>

Figure 13-2. Typical update file.

You can include single quotes in a string if you delimit the string with double quotes. For example:

| "741s00" | "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. You can, however, select the Convert update string to upper case option to force uppercase on all text added to the schematic.

- 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.
Chapter 13: Update Field Contents

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.
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 13-3).

![Figure 13-3. Local configuration screen for Update Field Contents.](image-url)
Chapter 13: 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:

○ 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.

○ 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:

○ Part Value  ○ Part Field 3  ○ Part Field 6
○ Part Field 1  ○ Part Field 4  ○ Part Field 7
○ Part Field 2  ○ Part Field 5  ○ Part Field 8

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:

- 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.

- Set specified field to visible
  
  Causes Update Field Contents to make all updated fields visible.

- Set specified field to invisible
  
  Causes Update Field Contents to make all updated fields invisible.

Select any combination of these options:

- Convert update string to uppercase
  
  Causes Update Field Contents to convert the update string to all upper case letters before placing the text in a field. The update file itself remains unchanged.

- Convert key field match string to upper case
  
  Causes Update Field Contents to be case-insensitive when matching update file match strings with part fields on the schematic.

- Ignore warnings
  
  Causes Update Field Contents to continue running when it encounters warnings instead of halting.
Included with Schematic Design Tools 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 14: 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 15: List Library describes how to list a library's contents with List Library.

Chapter 16: 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 17: Edit Library describes the Edit Library graphical part editor and its commands.

Chapter 18: Decompile Library describes how Decompile Library converts a compiled library file into a library source file.

Chapter 19: Creating a library source file with a text editor describes, in detail, how to create a library source file with a text editor.

Chapter 20: Symbol Description Language explains how to define a part in a custom library using OrCAD's Symbol Description Language (SDL).

Chapter 21: 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.

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 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 20: 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 21, 18, and 16, 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 17). 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, 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.

Δ NOTE: Libraries can be loaded into main system memory (RAM), EMS memory, 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 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 15.

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 17 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 19. Compile Library is described in chapter 21.

Figure 14-1 shows the development process using Compile Library to create a custom part 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 14-2 shows this process.

Figure 14-2. 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 14-3 shows examples of parts with block bodies. A block part can be up to 12.7 inches by 12.7 inches in size.

![Diagram of block parts](image)

*Figure 14-3. Parts with block bodies.*
Chapter 14: About libraries

**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.

If your part is larger than 1.2 inches by 1.2 inches, it must be an IEEE part (rather than a graphic part). However, if a graphic part’s X axis is less than 1.2 inches, its Y axis can be larger than 1.2 inches, and vice versa. Think of it as a rubber band . . . if you stretch it in one direction, it thins out in the other.

Figure 14-3 shows examples of parts with graphic bodies.

![Figure 14-3. Parts with graphic bodies.](image)

**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 may be larger. You must always create an IEEE part (rather than a graphic part) if the part will be larger than 1.2 by 1.2 inches. An IEEE part can be up to 12.7 inches by 12.7 inches in size.

---

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.

![PIN SHAPES](image)

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 14-3 and 14-4 show parts with pin numbers.

△ **NOTE:** *For parts with alphanumeric pin numbers, use Grid Array parts.*
Chapter 14: About libraries

**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 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 15-1 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.
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.

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 15-2).

Figure 15-2. List Library's local configuration screen.
### Chapter 15: List Library

**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.
You may select any combination of the following options:

- **Quiet mode**
  
  Turns quiet mode on.

- **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. String files are used to create libraries with **Compile Library**.

  String files can be used to create libraries with a subset of parts, or as the "left" column of a stuff file.

Below is an example of a short string file:

```
'GND'
'74LS00'
'RESISTOR'
'SW SPDT'
```
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.
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:

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 21: 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 16-1).

![Configure Library Archive](image)

Figure 16-1. Archive Parts in Schematic'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:

- 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.

- Source file is a single sheet

  Specifies that the source file is a single worksheet and you want to process the single sheet only.
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:

- 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).
Chapter 16: Archive Parts in Schematic

- **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**
  
  Causes LIBARCH to continue running when it encounters warnings, instead of halting in the middle of execution.
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 21: Compile Library, replacing references to Compile Library with COMPOSER.
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 bitmaps and 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.

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 the library.
- Modify an existing part in an existing library.

See Chapter 5: Creating a custom component in the Schematic Design Tools 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.

The body of a library part can be defined in vector form and bitmap form. Edit Library reads only the vector definition of a part. Before you use Edit Library, you should become familiar with vectors and bitmaps and how Edit Library uses them.

A vector is a set of numbers that describe the starting and ending coordinates, size, and so on, of a graphic object. For example, the following vector describes a straight line:

```
LINE +1.0 +2.0 +1.0 +4.0
```

The first two numbers (+1.0, +2.0) are the X and Y coordinates of the beginning of the line. The second two numbers (+1.0, +4.0) are the X and Y coordinates of the end of the line. Circles, arcs, and text can be specified in a similar way.

Although vectors are a precise way to specify graphic objects, bitmaps are a faster way to send the objects to a screen or printer. A bitmap is a set of bits in memory correlating to pixels on a screen or dots on a printer. A bitmap tells which pixels to “turn on” on a screen and which dots to print on a printer. Unlike vectors, bitmaps require little processing before being displayed or printed.
NOTE: A disadvantage of bitmaps is that a bitmap of a large object requires a lot of memory. This is true even if most of the pixels are not used.

IEEE parts are represented by vectors. Although the main purpose for IEEE type parts is to support drawing to the IEEE standard, IEEE type parts are also useful for handling parts that are not practical as bitmaps.

If you use Edit Library to create or edit a part and then write it out to a library, the part is stored in both vector and bitmap form. The bitmap description is used by Draft and Print Schematic to bypass the time-consuming process of converting vectors to bitmaps. The vector definition is used by Plot Schematic to produce high quality plots.

CAUTION: Although the parts in the OrCAD libraries use bitmap and vector definitions, parts in other libraries may not use vectors. Be careful when using Edit Library to edit a part without a vector definition. If you do, you will not see the part's shape on the screen. The bitmap defining the shape is still there, but Edit Library is not designed to read bitmaps. If you write the part back to the library with the LIBRARY Update Current command followed by QUIT Update or QUIT Write, you save the part as it appears on the screen—that is, without a body. You lose the bitmap.

If you're not sure whether a part has a vector definition, run Decompile Library on the library file containing the part. Then use a text editor to look at the library source file produced by Decompile Library. If the part has a vector definition, it always appears at the end of the part's definition.
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. 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.
Chapter 17: Edit Library

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.

Limit on a part's complexity

Using Edit Library, you construct a part from lines, arcs, circles, and so on. Each part has a maximum number of such elements allowed. The limit is for each part body; that is, the limit for a normal part is distinct from the limit for its converted form. Also, the limit does not depend on the number of parts per package.

For example, consider a part with four parts per package and a converted form. There are two limits pertinent for this part, one for the normal version and one for the converted form version. If you edit the part so it has only two parts per package, the limits do not change.

The limits for graphic parts are as follows:

- **Lines**: 127
- **Circles**: 127. The IEEE symbol \( \circ \) (negation) counts as a circle
- **Text**: 255 strings. Each string has a maximum of 10 characters
- **Arcs**: 63
- **Fills**: 31

IEEE parts are constructed in an area of 4096 bytes. The number of bytes required for each type of object are:

- **Lines**: 9
- **Circles**: 8
- **IEEE Symbols**: 8
- **Text**: 7 + length of text (8–17)
Hence, the total number of objects within an IEEE part is dependent on the mixture of the different type of objects in the part. At most, an IEEE part can have:

\[
\text{Lines} \quad \frac{4096}{9} = 455 \\
\text{Circles} \quad \frac{4096}{8} = 512 \\
\text{IEEE Symbols} \quad \frac{4096}{8} = 512 \\
\text{Text} \quad \frac{4096}{8...17} = 512-240
\]

**Limit of total library size**
Each library consists of several parts, each of which occupies its own block in memory. Each block can occupy up to 65536 bytes. As you add parts to the library, you may fill one of these blocks. At this point, the library is full. As you approach the memory limit, Edit Library displays a warning message so you do not lose any work. For information about checking the amount of remaining memory in these blocks, see the CONDITIONS command later in this chapter.

**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.
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 17-1).

**Local Configuration**

**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:

```
\ORCADES\SDT\LIBRARY\*.LIB
```

If you erase the entire field, the entry is restored to the prefix specified in the Library Prefix entry box on the Configure Schematic Design Tools screen.

**Figure 17-1. Edit Library's local configuration 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.
Chapter 17: Edit Library

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.
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 17–1 to quickly look up the command alphabetically or use table 17–2 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.

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 <Return> or clicking MENU. Move the pointer over the command you wish to select and click SELECT.

Press <Esc> to return to the previous menu.
<table>
<thead>
<tr>
<th>Command</th>
<th>Menu commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGAIN</td>
<td>none</td>
</tr>
<tr>
<td>BODY</td>
<td>Kind of Part? Block Graphic IEEE</td>
</tr>
<tr>
<td>BODY &lt;Block&gt;</td>
<td>Size of Body Kind of Part</td>
</tr>
<tr>
<td>BODY &lt;Graphic&gt;</td>
<td>Line Circle IEEE Symbol Arc Fill</td>
</tr>
<tr>
<td></td>
<td>Text IEEE Symbol Erase Body</td>
</tr>
<tr>
<td>BODY &lt;IEEE&gt;</td>
<td>Line Circle Text</td>
</tr>
<tr>
<td></td>
<td>IEEE Symbol Delete Erase Body</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>none</td>
</tr>
<tr>
<td>EXPORT</td>
<td>none</td>
</tr>
<tr>
<td>GET PART</td>
<td>none</td>
</tr>
<tr>
<td>IMPORT</td>
<td>none</td>
</tr>
<tr>
<td>JUMP</td>
<td>A-H tags X location Y location</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>Update Current Delete Part List List Directory Prefix Browse</td>
</tr>
<tr>
<td>MACRO</td>
<td>Capture List Delete Read Initialize Write</td>
</tr>
<tr>
<td>NAME</td>
<td>Add Delete Edit</td>
</tr>
<tr>
<td>ORIGIN</td>
<td>none</td>
</tr>
<tr>
<td>PIN</td>
<td>Add Delete Name Shape</td>
</tr>
<tr>
<td></td>
<td>Pin-Number Type Jump</td>
</tr>
<tr>
<td>QUIT</td>
<td>Update File Write to File Initialize Run User Commands</td>
</tr>
<tr>
<td>REFERENCE</td>
<td>none</td>
</tr>
<tr>
<td>SET</td>
<td>Auto Pan Backup File Error Bell</td>
</tr>
<tr>
<td></td>
<td>Left Button Macro Prompts Power Pins Visible</td>
</tr>
<tr>
<td></td>
<td>Show Body Outline Visible Grid Dots</td>
</tr>
<tr>
<td>TAG</td>
<td>A-H tags</td>
</tr>
<tr>
<td>ZOOM</td>
<td>Center Select In Out</td>
</tr>
</tbody>
</table>

Table 17-1. Edit Library menu commands.
<table>
<thead>
<tr>
<th>Category</th>
<th>Task</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entering and editing objects and data</td>
<td>Display a menu with commands you use to define a part’s type and draw the part.</td>
<td>BODY</td>
</tr>
<tr>
<td></td>
<td>Add, delete, or edit a part name, or activate or deactivate a part name prefix.</td>
<td>NAME</td>
</tr>
<tr>
<td></td>
<td>Add, delete, name, number, or edit pins on the current part.</td>
<td>PIN</td>
</tr>
<tr>
<td></td>
<td>Specify or edit a part’s reference designator.</td>
<td>REFERENCE</td>
</tr>
<tr>
<td>Accessing libraries</td>
<td>Write a definition of the current part to a file that you then add to a library using the IMPORT command.</td>
<td>EXPORT</td>
</tr>
<tr>
<td></td>
<td>Read a part from an export file created with the IMPORT command.</td>
<td>IMPORT</td>
</tr>
<tr>
<td></td>
<td>Retrieve a part from a library for editing.</td>
<td>GET PART</td>
</tr>
<tr>
<td></td>
<td>Update the current library, list the part names in the library, browse through the library, delete a part from the library, and define prefixes.</td>
<td>LIBRARY</td>
</tr>
<tr>
<td>Navigating on the screen</td>
<td>Move the pointer to a specified location on the screen.</td>
<td>JUMP</td>
</tr>
<tr>
<td></td>
<td>Change the amount of detail you see on the screen.</td>
<td>ZOOM</td>
</tr>
<tr>
<td>Repeating repetitive or complex tasks</td>
<td>Repeat the last main menu command.</td>
<td>AGAIN</td>
</tr>
<tr>
<td></td>
<td>Define and use macros.</td>
<td>MACRO</td>
</tr>
<tr>
<td>Setting locations and conditions</td>
<td>Change the origin (0,0) to the pointer’s current position.</td>
<td>ORIGIN</td>
</tr>
<tr>
<td></td>
<td>Set the status of Edit Library parameters.</td>
<td>SET</td>
</tr>
<tr>
<td></td>
<td>Assign tags to specific locations on the screen.</td>
<td>TAG</td>
</tr>
<tr>
<td>Showing status or other information</td>
<td>Display memory available for the macro buffer, system, library, and current part.</td>
<td>CONDITIONS</td>
</tr>
<tr>
<td>Leaving the program</td>
<td>Save parts and libraries, abandon edits without saving, initialize Edit Library, suspend to system, or exit the program.</td>
<td>QUIT</td>
</tr>
</tbody>
</table>

Table 17-2. 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 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 17-2).

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 12.7 inches by 12.7 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.

If your part is larger than 1.2 inches by 1.2 inches, it must be drawn as an IEEE part (rather than a graphic part). However, if a graphic part's X axis is less than 1.2 inches, its Y axis can be larger than 1.2 inches, and vice versa. Think of it as a rubber band ... if you stretch it in one direction, it thins out in the other.
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. You must always create an IEEE part (rather than a graphic part) if the part will be larger than 1.2 by 1.2 inches. An IEEE part can be up to 12.7 by 12.7 inches.

An example of an IEEE part

![Diagram 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.

<table>
<thead>
<tr>
<th>Kind of Part?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
</tr>
<tr>
<td>Graphic</td>
</tr>
<tr>
<td>IEEE</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Kind of Part?</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>A block part is one whose body is a square or rectangle. For example, a JK flip-flop is a block part.</td>
<td>If you choose Block, Edit Library requests the number of parts per package. Choose a number from 0 to 16. 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.</td>
</tr>
<tr>
<td>Graphic</td>
<td>A graphic part is one whose body contains curves or arcs. For example, an OR gate is a graphic part.</td>
<td>If you choose Graphic, Edit Library requests the number of parts per package. Choose a number from 0 to 16. 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).</td>
</tr>
<tr>
<td>IEEE</td>
<td>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.</td>
<td>If you choose IEEE, Edit Library displays the Place command (described on the next page).</td>
</tr>
</tbody>
</table>
Chapter 17: Edit Library

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 17-2.

If SET Show Body is turned on, a solid line represents vector-drawn parts (IEEE and block), and a dashed line indicates a bit-mapped part (graphic or IEEE).

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.
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.
Chapter 17: Edit Library

BODY <Graphic> commands

The BODY <Graphic> commands are shown at the right. Descriptions of these commands follow.

<table>
<thead>
<tr>
<th>BODY &lt;Graphic&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
</tr>
<tr>
<td>Circle</td>
</tr>
<tr>
<td>Arc</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>IEEE Symbol</td>
</tr>
<tr>
<td>Fill</td>
</tr>
<tr>
<td>Delete</td>
</tr>
<tr>
<td>Erase Body</td>
</tr>
<tr>
<td>Size of Body</td>
</tr>
<tr>
<td>Kind of Part</td>
</tr>
</tbody>
</table>

Use the Line command to draw lines. This command is similar to Draft’s PLACE Wire.

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 (if the line you are drawing makes a 90° 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.
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.

\[\text{NOTE: Circles may appear elliptical on your screen, depending on the type of monitor you have, because of variations between monitors. They will print or plot correctly, though.}\]
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 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°. Although you can move the pointer outside the current quadrant, the arc can only be drawn in the current quadrant.

For finer control when drawing an arc, zoom in to ½ or ¼ scale.
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 to make the text larger and Smaller to make it 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 <IEEE> Text to make the pin names visible on IEEE parts. If you do this, the text rotates and mirrors with the graphic image. Be aware, however, that Schematic Design Tools does not recognize the text as a pin definition.
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, its submenus, and the IEEE symbols placed by each command are shown in figure 17-3.

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 17-3. 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.
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:

```
IFill Jump Origin Tag Zoom
```

To darken an enclosed area, move the pointer inside the area and select **Fill**.

► **Helpful hints** . . .

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.

---

**CAUTION:** Be careful when you delete an item; you cannot undo the deletion.

Select **Delete**. The following command line displays:

```
IDelete Jump Origin Tag Zoom
```

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.
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.

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.

To draw a different type of part, select **Kind of Part**. Edit Library displays:

<table>
<thead>
<tr>
<th>Changes to Current Part may be lost. Continue?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Yes</strong>. The menu shown at right displays. Select the type of part to draw, as described previously in this section.</td>
</tr>
<tr>
<td>Kind of Part?</td>
</tr>
<tr>
<td>Block</td>
</tr>
<tr>
<td>Graphic</td>
</tr>
<tr>
<td>IEEE</td>
</tr>
</tbody>
</table>
Chapter 17: Edit Library

BODY <IEEE> commands

The BODY <IEEE> commands are shown at the right. Descriptions of each of the commands on this menu follow.

<table>
<thead>
<tr>
<th>Body &lt;IEEE&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
</tr>
<tr>
<td>Circle</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>IEEE Symbol</td>
</tr>
<tr>
<td>Delete</td>
</tr>
<tr>
<td>Erase Body</td>
</tr>
<tr>
<td>Size of Body</td>
</tr>
<tr>
<td>Kind of Part</td>
</tr>
</tbody>
</table>

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:

```
IBegin 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 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 (if the line you are drawing makes a 90° 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.
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.
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. If you do this, the text rotates and mirrors with the graphic image. Be aware, however, that Schematic Design Tools does not recognize the text as a pin definition.
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 its submenus, and the IEEE symbols placed by each command are shown in figure 17-3.

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**. When you are done, press `<Esc>`. The **IEEE Symbol** menu (figure 17-2) 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 <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.
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**.

—if 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.

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.

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.
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.

<table>
<thead>
<tr>
<th>Kind of Part?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
</tr>
<tr>
<td>Graphic</td>
</tr>
<tr>
<td>IEEE</td>
</tr>
</tbody>
</table>
CONDITIONS

CONDITIONS reports your computer’s memory, and the memory available for the worksheet, hierarchy buffer, and macro buffer. When you select CONDITIONS, a screen similar to the one shown in figure 17-4 displays. To return to the main menu level, press any key or mouse button.

You can print a copy of the CONDITIONS screen by pressing <Print Screen>. Make sure the Disable <Print Screen> key function option on Edit Library’s local configuration screen is not selected.

The CONDITIONS screen has two areas: The upper area describes the environment and the current library, and the lower area describes the current part. If no part is current, only the upper area displays. The lower area contains a variety of information, depending on the part type.

<table>
<thead>
<tr>
<th>CONDITIONS:</th>
<th>Allocated</th>
<th>Used</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Buffer</td>
<td>8192</td>
<td>0</td>
<td>8192</td>
</tr>
<tr>
<td>Free System Memory</td>
<td>---------</td>
<td>------</td>
<td>106352</td>
</tr>
<tr>
<td>Library</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objects</td>
<td>1023</td>
<td>634</td>
<td>389</td>
</tr>
<tr>
<td>General Control Information</td>
<td>65520</td>
<td>64435</td>
<td>1085</td>
</tr>
<tr>
<td>Bit Map Images</td>
<td>65520</td>
<td>11849</td>
<td>53671</td>
</tr>
<tr>
<td>Vectors (Circles, Lines, etc)</td>
<td>65520</td>
<td>1613</td>
<td>63907</td>
</tr>
<tr>
<td>Current Part (Type = Graphic + Convert)</td>
<td>Size X = 60 Y = 40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pins</td>
<td>255</td>
<td>7</td>
<td>248</td>
</tr>
<tr>
<td>Normal Image</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arcs</td>
<td>64</td>
<td>2</td>
<td>62</td>
</tr>
<tr>
<td>Circles</td>
<td>128</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>Lines</td>
<td>128</td>
<td>3</td>
<td>125</td>
</tr>
<tr>
<td>Texts</td>
<td>256</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>Fills</td>
<td>32</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>Convert Image</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arcs</td>
<td>64</td>
<td>4</td>
<td>60</td>
</tr>
<tr>
<td>Circles</td>
<td>128</td>
<td>2</td>
<td>126</td>
</tr>
<tr>
<td>Lines</td>
<td>128</td>
<td>2</td>
<td>126</td>
</tr>
<tr>
<td>Texts</td>
<td>256</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>Fills</td>
<td>32</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

Figure 17-4. CONDITIONS screen.
Macro Buffer
This shows the amount of memory available in the macro buffer. The buffer contains user-created macros—both those created on line and those loaded from a macro file. You can change the amount of memory allocated to the macro buffer on the Configure Schematic Design Tools screen. See Chapter 1: Configure Schematic Tools for instructions.

Free System Memory
This shows how much unused memory remains in your computer.

Library
This area describes the current library.

Objects
This tells how many names you can add to 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 14: About libraries in this guide.

General Control Information
This shows the number of bytes available for additional library symbols. General Control Information symbols include part names, pin names, pin positions, block symbols, and so on.

Bit Map Images
This shows the number of bytes available for additional library bitmaps. When you make a part with Edit Library, you use vectors. However, bitmaps are needed for screen work. For example, Draft requires a bitmap to display a graphic part. Consequently, when you add a graphic part to a library, a bitmap definition is stored as well as a vector definition.

Vectors
(Circles, Lines, etc.)
This shows the number of bytes available for additional library vectors. Edit Library vectors include lines, circles, arcs, IEEE symbols, and text.
Chapter 17: Edit Library

Current Part
This area describes the current part.

Type
Tells the type of the current part. The types are:
- Block
- Block/GRIDARRAY
- Graphic
- Graphic + Convert
- IEEE

Size
Tells the size of the current part.

Pins
Tells the number of pins allocated, used, and available.

Normal Image
Tells how many of these objects on graphic parts are allocated, used, and available:
- Arks
- Circles
- Lines
- Texts
- Fills

Convert Image
Tells how many objects on graphic parts with convert are available, used, and remaining. The objects are the same types reported in the Normal Image area.

Space
Tells how much space on IEEE parts is available, used, and remaining in bytes. This table lists the objects you can place and the number of bytes they use:

<table>
<thead>
<tr>
<th>Object</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>9</td>
</tr>
<tr>
<td>Circle</td>
<td>7</td>
</tr>
<tr>
<td>Text</td>
<td>7 plus 1 byte per character</td>
</tr>
<tr>
<td>IEEE Symbol</td>
<td>8 except Negation which is 7</td>
</tr>
</tbody>
</table>
**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.
Chapter 17: Edit Library

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 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:

<table>
<thead>
<tr>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the desired part name exactly as it appears in the library directory. Edit Library searches the configured and finds the part you requested. Once it finds the part, Edit Library displays it on the screen.</td>
<td>Press &lt;Enter&gt;. Edit Library displays a menu listing the library parts. Move the highlight to the part name you want, then press &lt;Enter&gt; to retrieve the part. Edit Library displays the part on the screen.</td>
</tr>
</tbody>
</table>

> Helpful hint . . .

To verify the spelling of a part name, use the LIBRARY Directory command.

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 14: 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 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:

Enter the name of the part 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.

<table>
<thead>
<tr>
<th>Jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>A tag</td>
</tr>
<tr>
<td>B tag</td>
</tr>
<tr>
<td>C tag</td>
</tr>
<tr>
<td>D tag</td>
</tr>
<tr>
<td>E tag</td>
</tr>
<tr>
<td>F tag</td>
</tr>
<tr>
<td>G tag</td>
</tr>
<tr>
<td>H tag</td>
</tr>
<tr>
<td>X location</td>
</tr>
<tr>
<td>Y location</td>
</tr>
</tbody>
</table>

**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 \( \frac{1}{100} \) (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 1 inch. If you enter 10, without a plus or minus sign, the pointer moves to the actual X grid coordinate 10.0.
Chapter 17: Edit Library

JUMP Y-Location

The Y-Location command moves the pointer a specific distance along the Y-axis. Each step represents $\frac{1}{100}$ (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 up, and a negative number (−10, −25, −30, and so on) down. If you enter +10, for example, the pointer jumps up 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.

<table>
<thead>
<tr>
<th>Library filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Current</td>
</tr>
<tr>
<td>List Directory</td>
</tr>
<tr>
<td>Browse</td>
</tr>
<tr>
<td>Delete Part</td>
</tr>
<tr>
<td>Prefix</td>
</tr>
</tbody>
</table>

**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).
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.

**Screen** Select Screen. The library directory displays on the screen.

**Printer** Select Printer. The library directory prints on the printer.

**File** Select File. The prompt "File?" displays. Enter the path and filename. The library directory is sent to a file.

### Directory of EXAMPLE.LIB

<table>
<thead>
<tr>
<th>Prefix -</th>
<th>74LS</th>
<th>74S</th>
<th>74ALS</th>
<th>74AS</th>
<th>74HCT</th>
<th>74HC</th>
<th>74ACT</th>
<th>74AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shorthand-</td>
<td>LS</td>
<td>S</td>
<td>ALS</td>
<td>AS</td>
<td>HCT</td>
<td>HC</td>
<td>ACT</td>
<td>AC</td>
</tr>
<tr>
<td>Prefix -</td>
<td>74AHCT</td>
<td>74FCT</td>
<td>74F</td>
<td>74C</td>
<td>74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shorthand-</td>
<td>AHCT</td>
<td>FCT</td>
<td>F</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
</tr>
<tr>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td>71</td>
</tr>
<tr>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
</tr>
<tr>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
</tr>
<tr>
<td>99</td>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
</tr>
</tbody>
</table>

**Figure 17-5.** 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 Quit 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:

<table>
<thead>
<tr>
<th>Select Index</th>
<th>ID</th>
<th>Prefix</th>
<th>Short Prefix</th>
<th>In Use By</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>74LS</td>
<td>LS</td>
<td>277</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>74S</td>
<td>S</td>
<td>83</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>74ALS</td>
<td>ALS</td>
<td>241</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>74AS</td>
<td>AS</td>
<td>162</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>74HCTLS</td>
<td>HCTLS</td>
<td>159</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>74HCT</td>
<td>HCT</td>
<td>121</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>74HC</td>
<td>HC</td>
<td>178</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>74ACT</td>
<td>ACT</td>
<td>89</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>74AC</td>
<td>AC</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>74HCT</td>
<td>AHCT</td>
<td>159</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>74FCT</td>
<td>FCT</td>
<td>49</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>74F</td>
<td>F</td>
<td>216</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>74C</td>
<td>C</td>
<td>71</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>74PCT</td>
<td>PCT</td>
<td>46</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>&amp;$BC</td>
<td>BC</td>
<td>30</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

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 14: About libraries.

Up to 16 prefixes can be defined in each library.
Chapter 17: Edit Library

0 through F

Once you select Prefix, the menu shown at right displays.

To add or change a prefix or short prefix, choose the number that corresponds to the ID of 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.

Enter (or edit) the prefix or short prefix. Edit Library makes the change in the table displayed in the window.
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 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.

➤ 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 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 sheet parts, see Editing sheets and PLACE Sheet in Chapter 2: Draft in this 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.
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.
Schematic Design Tools Reference Guide

PIN

Use the PIN command to add, delete, or edit pins.

Select PIN. Edit Library displays:

| Add | Delete | Name | Pin-Number | Type | Shape | Jump | Zoom |

**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:

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.

<table>
<thead>
<tr>
<th>Pin Type</th>
<th>Input</th>
<th>Output</th>
<th>Bidirectional</th>
<th>Power</th>
<th>Passive</th>
<th>3 State</th>
<th>Open Collector</th>
<th>Open Emitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>An output pin is one to which the part applies a signal. For example, pin 3 on the 74LS00 NAND gate is an output.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bidirectional</td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>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). Power pins are invisible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passive</td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 State</td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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-up. 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.
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.

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. The current library is removed from memory, and the following prompt displays:

Read library?

Enter the path and filename of the file you want to edit.

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 allow the initialize command to continue.

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 for an example. In addition, these types of parts are not shown in the **Bill of Materials** report.
Chapter 17: Edit Library

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

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

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 worksheet 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 back-up files created; select No if you don’t want back-up 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. |
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.

<table>
<thead>
<tr>
<th>Zoom (present scale=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center (1)</td>
</tr>
<tr>
<td>In (Half)</td>
</tr>
<tr>
<td>Out (1)</td>
</tr>
<tr>
<td>Select</td>
</tr>
</tbody>
</table>

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 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. You can 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 the library source file from the library file usable by Draft.

When Decompile Library completes its task, the Schematic Design Tools screen appears.
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 18-1).

![Configure Decompile Library](image)

**Figure 18-1. 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.
Chapter 18: Decompile Library

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 1.2 inches by 1.2 inches in size.

If your part is larger than 1.2 inches by 1.2 inches, it must be an IEEE part (rather than a graphic part). However, if a graphic part's X axis is less than 1.2 inches, its Y axis can be larger than 1.2 inches, and vice versa. Think of it as a rubber band . . . if you stretch it in one direction, it thins out in the other.
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 also be larger than a graphic part. An IEEE part can be up to 12.7 inches by 12.7 inches.

△ NOTE: For more information, 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.

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.

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 19-1.

<table>
<thead>
<tr>
<th>Get? 04</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS04</td>
</tr>
<tr>
<td>74S04</td>
</tr>
<tr>
<td>74ASL04</td>
</tr>
<tr>
<td>74AS04</td>
</tr>
<tr>
<td>74HCT04</td>
</tr>
<tr>
<td>74HC04</td>
</tr>
<tr>
<td>74F04</td>
</tr>
<tr>
<td>7404</td>
</tr>
</tbody>
</table>

Figure 19-1. 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
```

\[\triangle\] NOTE: To view the prefix definition of a library file, use Decompile Library to convert the file to a library source file. See Chapter 18: 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.
Chapter 19: Creating a library source file with a text editor

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 bitmap and vector definitions. When Compile Library sees a bitmap, it uses the bitmap to represent the part, rather than defaulting to a square or rectangle.

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.
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 number 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 19-2.
Chapter 19: Creating a library source file with a text editor

Figure 19-2. Pin definition line.

For specific details about defining a pin, see Chapter 20: Symbol Description Language.

- An optional bitmap definition. Use this if the symbol you want is not a square or rectangle.

- An optional vector definition. This describes a graphic part in vector format. Edit Library and Plot Schematic use vector definitions for screen display and plotting.

- An optional converted form bitmap definition. This only has meaning if you've defined a bitmap. The most common use for converted bitmaps is to specify the DeMorgan equivalent symbol of the defined part.

- An optional converted form vector definition. This describes the converted form of a graphic part in vector format.
Defining a block symbol

Figure 19-3 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 19-4 shows the symbol produced by this block definition.

<table>
<thead>
<tr>
<th>Reference</th>
<th>'LATCH'</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Pin Name</th>
<th>Location</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>3</td>
<td>11</td>
<td>SHORT IN 'J'</td>
</tr>
<tr>
<td>L5</td>
<td>1</td>
<td>13</td>
<td>DOT CLK IN 'CLK'</td>
</tr>
<tr>
<td>L9</td>
<td>2</td>
<td>12</td>
<td>SHORT IN 'K'</td>
</tr>
<tr>
<td>B3</td>
<td>15</td>
<td>14</td>
<td>DOT IN 'CL'</td>
</tr>
<tr>
<td>T3</td>
<td>4</td>
<td>10</td>
<td>DOT IN 'P'</td>
</tr>
<tr>
<td>R1</td>
<td>6</td>
<td>7</td>
<td>OUT 'Q'</td>
</tr>
<tr>
<td>R9</td>
<td>5</td>
<td>9</td>
<td>OUT 'Q'</td>
</tr>
<tr>
<td>T0</td>
<td>16</td>
<td>16</td>
<td>PWR 'VCC'</td>
</tr>
<tr>
<td>B0</td>
<td>8</td>
<td>8</td>
<td>PWR 'GND'</td>
</tr>
</tbody>
</table>

Figure 19-3. Block symbol definition for 74EXAMP.

Figure 19-4. 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.
Chapter 19: Creating a library source file with a text editor

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 19-5.

Figure 19-5. Block symbol definition with multiple part names.

Notice that comments are enclosed in curly braces, as in figure 19-5. The X and Y coordinates are preceded by comments, as is the number of parts per package.

Sheetpath keyword

The example in figure 19-3 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. Then, whenever you place a sheetpath part on your design, you actually place a copy of this schematic. 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.

335
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 19-3, the reference designator appears as "LATCH." When you run Annotate Schematic on a design containing the part, the question mark 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.

\[\text{NOTE: If, for some reason, you want a part without pin numbers, use a device with zero parts per package.}\]

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 19-1 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.

<table>
<thead>
<tr>
<th>Source File . . .</th>
<th>Number of Parts Per Package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Does not use “REFERENCE” keyword</td>
<td>Nothing is displayed</td>
</tr>
<tr>
<td>Uses “REFERENCE” keyword</td>
<td>Uses reference keyword you enter in source library</td>
</tr>
</tbody>
</table>

Table 19-1. Controlling display of reference designators.

Grid unit size and parts/package

The next line in figure 19-3 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 19-3 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 19-3 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 19-4 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 19-6 shows a grid representing the possible pin positions for the 74EXAMP library part.

Note the SHORT keyword in the L1 pin position in figure 19-3. 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 19-6. The grid of a 6X by 10Y block symbol.](image)

**Pin type**

One possible pin type is PWR for power. Power pins do not appear on the screen. 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.

\[\triangle\] 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 19-7 and 19-8 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.
Also note the definition of 74EXAMP was modified so the power pins only appear in the second part of the package. 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 19-7. Before annotation.
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 19-9 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 19-10 shows the resulting screen figure.
Chapter 19: Creating a library source file with a text editor

'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 19-9. Pin-grid array part definition.

Figure 19-10. The block symbol for the 68020 defined in figure 19-9.
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\' results in Q with a bar over it (\bar{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\'.

Chapter 19: Creating a library source file with a text editor

Defining a graphic symbol

A graphic symbol definition is composed of two parts: a bitmap definition and a vector definition. Both definitions are optional. However, Draft requires a bitmap, while Plot Schematic requires vectors. A bitmap definition specifies which screen pixels should be “turned on” when the part displays in Draft. A vector definition specifies the shape of the part in vector format. The vector definition is used by the Edit Library and the Plot Schematic tools.

Defining a bitmap

Creating a bitmap is an easy way to represent non-rectangular parts such as resistors, diodes, transistors, MOSFETs, relays, and many others. Draft draws complex parts by selectively turning on pixel bits representing the library part. Activating the correct pixel bit is controlled by a bitmap in the library source file you created.

To define a part with a bitmap, you define the part just as if it were a block symbol, but you include a bitmap after the last pin definition. You can either draw the bitmap with periods (.) and pound signs (#) (see figure 19-11 for an example), or you can reference a previously drawn bitmap. “Previously drawn” means the bitmap was defined earlier in the library source file.

Refer to a previously drawn bitmap if two parts have different pinouts, but the same symbol. For example, the 7439 and the 7400 have the same symbol, but different pinouts. Assume you’ve defined the 7400 and you’re now defining the 7439. Instead of drawing another bitmap, you can use the 7400’s for the 7439 by including the line:

```
BITMAP '7400'
```

If you don’t reference a previously drawn-out bitmap, you must specify the part’s own bitmap. The bitmap begins after the last pin definition. A pound sign (#) indicates the pixel bit is turned on, and a period (.) indicates the pixel bit is turned off.
Each . or # in the bitmap represents a screen pixel spacing of 0.01 inch in the X direction. Each line of the bitmap represents 0.01 inch in the Y direction. Remember, the X and Y sizes in the part definition are given in units of 0.1 inch. For example, if you specify X and Y to be 3 and 2, your bitmap actually is 31 characters in the X direction and 21 lines in the Y direction. The extra 1 results because the bitmap starts counting at zero. Figure 19-11 shows the bitmap for a graphic.

Defining a vector

Following the bitmap definition, you can specify a vector definition of the part. 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 +4.0 +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 20: Symbol Description language.
Chapter 19: Creating a library source file with a text editor

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 can 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. The maximum number of bits allowed in a bitmap is 16,384. However, bitmaps are allocated in blocks of eight. So consider a bitmap consisting of 21 rows of 31 bits. Rather than $21 \times 31 = 651$ bits, the space actually taken up by such a bitmap is $21 \times 32 = 672$ bits.

5. **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 bitmap and 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 19-11 and figure 19-12 show 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.

Note that the part definition in figure 19-11 contains both the bitmap and vector definition of the part.
| '7400' |  
|---|---|
| (X Size =) | 6  |
| (Y Size =) | 4  |
| (Parts per Package =) | 4  |
| L1 | 1 4 9 12 IN '10' |
| L3 | 2 5 10 13 IN '11' |
| R2 | 3 6 8 11 DOT OUT 'O' |
| T0 | 14 14 14 14 PWR 'VCC' |
| B0 | 7 7 7 7 PWR 'GND' |
| (0000000001111111111112222222222333333333333333333334444444444455555555556) |
| (0123456789012345678901234567890123456789012345678901234567890) |
| (0.0) | 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 |
| (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 |

Figure 19-11. The 7400 symbol.
Figure 19-12. The converted form of the 7400 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 below.

```
'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 19-13. IEEE symbol definition for 74114 parts.*
Chapter 19: Creating a library source file with a text editor

Size and type definitions

The line following '74S114' in figure 19-13 contains the numbers 8 and 12, followed by the letters IEEE. The comments inside the curly braces (see figure 19-13) show the X Size, Y Size, and the keyword IEEE.

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 19-14. Inputs placed to the left of the part body have a negative X offset in the vector definition. See figure 19-13.
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 19-3, 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 20: 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 12.7 x 12.7 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.
Chapter 19: Creating a library source file with a text editor

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 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 19-14 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 nicely between pins which are normally spaced ten units apart.

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 for plots or hardcopy that is 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 an 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.
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).

Figure 20-1 is an example of a syntax diagram. It represents the complete syntax for a library source file.

![Syntax diagram](image)

Figure 20-1. 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 20-1 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 20-1, 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 20-1 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:

  **BITMAP** Takes an argument (a text string representing a part name) and represents the bitmap of the identified part.

  **CLK** Represents the clock symbol in a pin definition.
CONVERT  Introduces a converted graphic symbol definition. With an argument, it refers to the converted bitmap and vector definitions 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.
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 required. You must enter what’s represented within the braces. Don’t type the braces.

, 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 20-2. Syntax diagram for a prefix definition.
### Example 1

| PREFIX  | '
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'74LS'</td>
<td>'LS'</td>
</tr>
<tr>
<td>'74S'</td>
<td>'S'</td>
</tr>
<tr>
<td>'74ALS'</td>
<td>'ALS'</td>
</tr>
<tr>
<td>'74AS'</td>
<td>'AS'</td>
</tr>
<tr>
<td>'74HCT'</td>
<td>'HCT'</td>
</tr>
<tr>
<td>'74HC'</td>
<td>'HC'</td>
</tr>
<tr>
<td>'74ACT'</td>
<td>'ACT'</td>
</tr>
<tr>
<td>'74AC'</td>
<td>'AC'</td>
</tr>
<tr>
<td>'74F'</td>
<td>'F'</td>
</tr>
<tr>
<td>'74'</td>
<td></td>
</tr>
</tbody>
</table>

END

### Example 2

| PREFIX  | '
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END
Part definition

'part name string'
[REFERENCE 'ref string']
[SHEETPATH 'path and filename']
{X-size Y-size parts-per-package}
[pin definition]

... 
... 

[bitmap definition]
[vector definition]
[converted form bitmap definition]
[converted form vector definition]

Figure 20-3. Syntax diagram of a part definition.

Figure 20-4. Syntax diagram of a part name string.
△ **NOTE:** To type an apostrophe, use two single quotes. For example, to type:

'John's'
type:
'John''s'

△ **NOTE:** To improve readability, you can include comments within a part definition. Comment text is enclosed within curly brackets. 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 the **View Reference Material** tool.

**⇒ 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 127. The horizontal size of the part as it appears on a printed worksheet. Each entry corresponds to one grid unit.
Chapter 20: Symbol Description Language

- **Y size**
  A numeric constant in the range 1 to 127. 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.

- **bitmap definition**
  This definition describes the part body in bitmap form. See the bitmap 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 bitmap definition**
  This definition describes the part body of the part’s converted form (for example, its DeMorgan equivalent) in bitmap form. See the converted form bitmap 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 show 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] pin name]
[OUT]
[I/O]
[OC]
[OE]
[PWR]
[PAS]
[HIZ]]

Figure 20-5. 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 20-6 shows the location of L3.

![Figure 20-6. The grid of a 6X by 10Y block symbol.](image)
Chapter 20: Symbol Description Language

![Diagram of Position Syntax](image)

**Figure 20-7. The Position syntax number diagram.**

- **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 (figure 20-8). 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.

![Diagram of Gridarray Pin Number Syntax](image)

**Figure 20-8. Gridarray pin number syntax diagram.**

- **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 20-9 shows the syntax for the pin name string.

Fig. 20-9. Pin name string syntax diagram.
### Example 1

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'2114'</td>
<td>'2148'</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>L1</td>
<td>5</td>
<td>IN</td>
</tr>
<tr>
<td>L2</td>
<td>6</td>
<td>IN</td>
</tr>
<tr>
<td>L3</td>
<td>7</td>
<td>IN</td>
</tr>
<tr>
<td>L4</td>
<td>4</td>
<td>IN</td>
</tr>
<tr>
<td>L5</td>
<td>3</td>
<td>IN</td>
</tr>
<tr>
<td>L6</td>
<td>2</td>
<td>IN</td>
</tr>
<tr>
<td>L7</td>
<td>1</td>
<td>IN</td>
</tr>
<tr>
<td>L8</td>
<td>17</td>
<td>IN</td>
</tr>
<tr>
<td>L9</td>
<td>16</td>
<td>IN</td>
</tr>
<tr>
<td>L10</td>
<td>15</td>
<td>IN</td>
</tr>
<tr>
<td>L12</td>
<td>8</td>
<td>IN</td>
</tr>
<tr>
<td>L13</td>
<td>10</td>
<td>IN</td>
</tr>
<tr>
<td>R1</td>
<td>14</td>
<td>HIZ</td>
</tr>
<tr>
<td>R2</td>
<td>13</td>
<td>HIZ</td>
</tr>
<tr>
<td>R3</td>
<td>12</td>
<td>HIZ</td>
</tr>
<tr>
<td>R4</td>
<td>11</td>
<td>HIZ</td>
</tr>
<tr>
<td>T0</td>
<td>18</td>
<td>PWR</td>
</tr>
<tr>
<td>B0</td>
<td>9</td>
<td>PWR</td>
</tr>
</tbody>
</table>

### Example 2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'68020'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>66</td>
<td>GRIDARRAY</td>
</tr>
<tr>
<td>L1</td>
<td>'C2'</td>
<td>CLK IN 'CLK'</td>
</tr>
</tbody>
</table>

*and so on*

### Example 3

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'7474'</td>
<td>'74ALS74'</td>
<td>'74AS74'</td>
</tr>
<tr>
<td>'74S74'</td>
<td>'74HC74'</td>
<td>'74AC74'</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>L2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>L4</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>B3</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

*and so on*
Chapter 20: Symbol Description Language

Bitmap definition

\[
[\{.,#\}...] \text{BITMAP 'part name'} \text{,CONVERT 'part name'}]
\]

Figure 20-10. The bitmap syntax diagram.

Figure 20-11. The bitmap line.

- . A . represents a cleared pixel bit not displayed on the screen.

- # A # represents a set pixel bit.
A previously-defined part name with a bitmap. If you specify part name with the BITMAP keyword, the bitmap and vector definitions of the normal form of the previously-defined part are used. If you specify part name with the CONVERT keyword, the definitions of the converted form of the previously defined part are used.

Keep these limitations in mind:

- Maximum bitmap size is 16384 pixels.
- Each pixel represents 0.1 of a grid unit. Therefore, it takes 10 bitmap pixels to represent one grid unit. Remember, pin placement is determined according to grid units.
  
  For example, L0 identifies a pin on the left side of the bitmap line in the zero position. L1 identifies a pin on the left side of the tenth bitmap line. Pin placements are at lines 0, 10, 20, 30, etc.

- The lines and character positions of a bitmap start numbering at 0. Hence, a symbol with X=2 and Y=3 has 21 lines, each with 31 character positions.
- If a line contains only periods (a clear line), you need only place a period in the first column.
- You need not include clear lines (lines containing only periods) at the end of a bitmap.
- Periods are not required after the last # in a line.
- If you use the BITMAP 'part name' option, be sure the part name you refer to is previously defined.
Example 1

'CAPACITOR'
Reference 'C'
{X Size =} 2 {Y Size =} 1 {Parts per Package =} 0
T1 SHORT PAS '1'
B1 SHORT PAS '2'
{000000001111111111}
{012345678901234567890}
{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 +1.0 +0.4 +1.0 +1.0
ARC +1.0 +2.0 +1.0 -1.3 +0.0 -1.6 +1.6
ARC +1.0 +2.0 +0.0 -1.6 -1.0 -1.3 +1.6
END

Example 2

BITMAP '7400'
This uses the bitmap and vector definitions for the normal form of the 7400 device.

Example 3

CONVERT '7400'
This uses the bitmap and vector definitions for the converted form of the 7400 device. The 7400 part must have a converted form if the library source file is to compile without errors.
Vector definition

VECTOR
ARC X center Y center X edge1 Y edge1 X edge2 Y edge2 radius
CIRCLE X center Y center
FILL X start Y start
LINE X start Y start X end Y end
TEXT X start Y start height {'text'/keyword}
END

\[\text{△} \quad \text{NOTE: If you use the BITMAP 'part name' option (as described in the "Bitmap Definition" section), you don't need to specify a separate vector definition. The bitmap and vector definitions of the part referred to by part name are used.}\]

- **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.

- **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.
Chapter 20: Symbol Description Language

- **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.

- **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.

  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

[\textit{pin definitions}]

\[\cdot\]

\[\cdot\]

\[[\text{BITMAP 'part name'}, \text{CONVERT 'part name'}],\]

[\textit{bitmap definition}]

[\textit{vector definition}]}

\emph{Figure 20-12. Converted form syntax diagram.}

\begin{itemize}
  \item \textbf{part name} The part name of a previously-defined part. If you specify part name with the BITMAP keyword, the bitmap and vector definitions of the normal form of the previously-defined part are used. If you specify part name with the CONVERT keyword, the definitions of the converted form of the previously defined part are used.
  \item \textbf{pin definition} See the pin definition description earlier in this section. Note the pin definition for a converted bitmap has the same value for parts-per-package as the normal bitmap.
  \item \textbf{bitmap definition} This specifies a new bitmap for the converted form of the part. See the bitmap definition description earlier in this section. If you use bitmap definition, do not use the BITMAP or CONVERT keywords.
\end{itemize}
This specifies a new vector for the converted form of the part. See the vector definition description earlier in this section. If you use vector definition, do not use the BITMAP or CONVERT keywords.

\( \triangle \) \textbf{NOTE: If you use the CONVERT 'part name' option, be sure the part name you refer to is previously defined.}
Compile Library takes a library source file and produces a compiled library file used by the schematic editor, processors, reporters, and transfers.

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 20).

   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.

   \(\text{NOTE: The maximum length for a part name is 127 characters. Use much smaller part names if you possibly can, however. On a 640 x 480 screen at ZOOM level 1, a name longer than 78 characters will be clipped. Also, many netlist formats place severe restraints on the length of names. Check the name length requirements for the netlist format you are using. This information is available using the View Reference Material tool.}\)
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.

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 21-1).

![Configure Compile Library](image)

Figure 21-1. Compile Library's local configuration screen.

File Options defines the library source file and the resulting library file.
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:

`\ORCADES\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.

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.

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>.

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).
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 22:  *Check Electrical Rules* describes how *Check Electrical Rules* checks a design for conformity to basic electrical rules.

Chapter 23:  *Cross Reference Parts* describes how *Cross Reference Parts* scans through the design, gathers information for all parts used in the design, and creates a cross reference listing telling you where each part is located.

Chapter 24:  *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 25:  *Plot Schematic* describes how *Plot Schematic* plots designs.

Chapter 26:  *Print Schematic* describes how *Print Schematic* prints designs.

Chapter 27:  *Create Bill of Materials* describes how *Create Bill of Materials* creates a summary list of all parts used in a design.

Chapter 28:  *Show Design Structure* describes how *Show Design Structure* scans the worksheets in a design and displays the sheet names and their associated worksheet filenames.
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.

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 the figure below. Note that the wire coming from pin 5 on U1B and the wire coming from pin 3 of U1A have circles on them.

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 22-1 on the next page 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 22-1 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 is either 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 22-1. The decision matrix used by Check Electrical Rules to check a connection.
### Table 22-1. Error messages created by Check Electrical Rules and possible solutions.

<table>
<thead>
<tr>
<th>Message</th>
<th>Check for</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WARNING:</strong> Unconnected MODULE PORT “....” at X=... at Y=...</td>
<td>A bus may not be labeled properly. It 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 9: Creating a netlist.</td>
</tr>
<tr>
<td><strong>WARNING:</strong> POWER Supplies are CONNECTED .... &lt;-&gt; ....</td>
<td>If you connected two power supplies together, this warning appears. This condition may be acceptable in your design. If you did not mean to connect two power supplies together, this indicates a problem may exist. If you intentionally connected two power supplies together, you may ignore this warning.</td>
</tr>
<tr>
<td><strong>WARNING:</strong> INPUT has NO Driving Source ....</td>
<td>If you did not connect wires to the input pins of a library part, this warning appears. Again, this condition may be acceptable in your design. 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. If you intentionally did not connect wires to the input pins of a library part, you may ignore this warning.</td>
</tr>
<tr>
<td><strong>ERROR:</strong> Module Port on a bus does not have a proper format... can not process...</td>
<td>When a module port is connected to a bus, it must be in the format: BUSNAME[m..n].</td>
</tr>
<tr>
<td><strong>ERROR:</strong> Bus Label does not have a proper format... can not process...</td>
<td>When a label is placed on a bus, it must be in the format: BUSNAME[m..n].</td>
</tr>
<tr>
<td><strong>ERROR:</strong> Sheet Net on a bus does not have a proper format... can not process...</td>
<td>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].</td>
</tr>
</tbody>
</table>

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.
For example, if you have an output pin connected to an input pin, find the Check Electrical Rules value in the matrix by starting in the OUT column (the third column) and going down to the IN row (the first row). The box is empty, which 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.

Example

The following example shows the type of errors Check Electrical Rules looks for in schematic sheets. Figure 22-2 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.
Suppose you add a 74LS00 part (also from TTL.LIB) to the sheet shown in figure 22-3. 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,0
Warning, Possible conflict I/O connected to OUTPUT U2A,0
WARNING - INPUT has NO Driving Source U2B,10
WARNING - INPUT has NO Driving Source U2B,11
WARNING - INPUT has NO Driving Source U2A,10
WARNING - INPUT has NO Driving Source U2A,11

Figure 22-3. Example circuit for Check Electrical Rules (with 74LS00).
Chapter 22: Check Electrical Rules

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 22-4).

![Configure Check Electrical Rules](image)

Figure 22-4. 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:

- **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.

**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.
Chapter 22: Check Electrical Rules

- Unconditionally process all sheets in design
  Forcing 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.

- Check module port connections
  Causing Check Electrical Rules to check all module ports and make sure they have a corresponding connection.

- Do not report warnings
  Causing 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
  Causing Check Electrical Rules to continue running when it encounters warnings, instead of halting in the middle of execution.
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 23-1 shows a sample report created by Cross Reference Parts.

<table>
<thead>
<tr>
<th>Item</th>
<th>Reference</th>
<th>Part</th>
<th>SheetName</th>
<th>Sheet</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R1</td>
<td>1K</td>
<td>&lt;&lt;&lt;root&gt;&gt;&gt;</td>
<td>1</td>
<td>EX1.SCH</td>
</tr>
<tr>
<td>2</td>
<td>R2</td>
<td>1K</td>
<td>&lt;&lt;&lt;root&gt;&gt;&gt;</td>
<td>1</td>
<td>EX1.SCH</td>
</tr>
<tr>
<td>3</td>
<td>S1</td>
<td>SW SPDT</td>
<td>&lt;&lt;&lt;root&gt;&gt;&gt;</td>
<td>1</td>
<td>EX1.SCH</td>
</tr>
<tr>
<td>4</td>
<td>U1A</td>
<td>74LS00</td>
<td>&lt;&lt;&lt;root&gt;&gt;&gt;</td>
<td>1</td>
<td>EX1.SCH</td>
</tr>
<tr>
<td>5</td>
<td>U1B</td>
<td>74LS00</td>
<td>&lt;&lt;&lt;root&gt;&gt;&gt;</td>
<td>1</td>
<td>EX1.SCH</td>
</tr>
</tbody>
</table>

Figure 23-1. 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 23-2).

Figure 23-2. Local configuration screen for Cross Reference Parts.
<table>
<thead>
<tr>
<th><strong>File Options</strong></th>
<th>File Options defines the source file, its type, and the destination file.</th>
</tr>
</thead>
</table>
| **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 \texttt{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. |
<p>| <strong>Destination</strong> | 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. |</p>
<table>
<thead>
<tr>
<th>Processing Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiet mode</td>
<td>Turns quiet mode on.</td>
</tr>
<tr>
<td>Descend into sheetpath parts</td>
<td>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.</td>
</tr>
<tr>
<td>Report identical part reference designators</td>
<td>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.</td>
</tr>
<tr>
<td>Report unused parts in multiple-part packages</td>
<td>Tells Cross Reference Parts to list all unused parts in multiple-part packages in addition to its usual report.</td>
</tr>
<tr>
<td>Report the X, Y coordinates of all parts</td>
<td>Tells Cross Reference Parts to list the X,Y coordinates for all parts.</td>
</tr>
<tr>
<td>Place each part entry on a separate line</td>
<td>Tells Cross Reference Parts to place each part entry on a separate line.</td>
</tr>
</tbody>
</table>
Report type mismatch parts

Tells Cross Reference Parts to report type mismatches in addition to the usual report. Two kinds of mis-matches 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.

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 or not to place a header 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
- single-spaced
- double-spaced

If desired, select this option:
- Ignore warnings

Causes Cross Reference Parts to continue running when it encounters warnings, instead of halting in the middle of execution.
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 25: 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 24-1 shows a sample report created by Convert Plot to IGES.

```
IGES file generated from OrCAD Schematic Design Tools
,,21HorCAD IGES Translator,,5HorCAD,4HIGES,16,11,53,11,53,,1.0,3,4HINCTH,G
30,0.46785,6H000001,,,8HIGESTRAN,5HorCAD,6,0;

110 1 1 1 1
110 1 1 1
110 2 1 1 1
110 3 1 1 1
110 1 1 1
110 4 1 1 1
110 1 1 1
110 0.77,0.583,.0,0.77,0.517,.0; 1P
110,0.77,0.583,.0,0.93,0.517,.0; 3P
110,0.93,0.517,.0,0.93,0.583,.0; 5P
110,0.77,0.517,.0,0.93,0.517,.0; 7P
S 1G 2D 8P 4
```

Figure 24-1. 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 24-2).

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 plots schematic sheets.

There are two types of output devices that can be used with OrCAD Schematic Design Tools: 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 there is more mathematical calculation needed). For these reasons, use Plot Schematic to plot the schematic when design work is complete. Use Print Schematic (described in chapter 26) to print working copies of your schematic during the design process.

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.
Chapter 25: Plot Schematic

Sample output

Figure 25-1 shows a sample plot created by Plot Schematic.

Figure 25-1. 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 25-2).

![Plot Schematic local configuration screen](image)

**Figure 25-2. The Plot Schematic local configuration screen.**

**File Options**

File Options defines the source file and its type and the destination of the plot.

**Source**

The Source can be the root sheet name of the design, or the filename of a single sheet, with any valid pathname.
Chapter 25: Plot Schematic

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 three options:

- **Send output to plotter**
  
  Use this option only if your sheet dimensions are 32 inches by 32 inches or less. 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.

- **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.

\[ \text{NOTE: When plotting to a printer, make the pen width for buses small enough to make the lines thin. This makes the print neater and more readable. For example, if you have a printer with a resolution of 180 DPI, set the pen width to be:} \]

\[ \left( \frac{1}{180} \right) \frac{2}{2} = 0.00277 \text{ inches} \]

Round the result up to 0.003 inches for best results. The same value is also used for the part body pen width and during FILL commands in the vector stream of the part definition.
○ 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:

○ Create a PRINT file

File

○ 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 large file containing print information for every sheet in the design, separated by page breaks. Enter a name for the file in the File entry box.

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. It is a good idea to use this switch when you are reducing a plot or 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 25-1 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.

<table>
<thead>
<tr>
<th>Worksheet Size</th>
<th>Plotter Paper Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.000</td>
</tr>
<tr>
<td>B</td>
<td>0.638</td>
</tr>
<tr>
<td>C</td>
<td>0.474</td>
</tr>
<tr>
<td>D</td>
<td>0.301</td>
</tr>
<tr>
<td>E</td>
<td>0.224</td>
</tr>
</tbody>
</table>

Table 25-1. Plot Schematic scale factors.

\(\text{NOTE: Table 25-1 assumes you are using the default Template Table values for Horizontal and Vertical paper dimensions. These values are set on the Template Table section of the Configure Schematic Design Tools screen. For more information, see Chapter 1: Configure Schematic Tools.}\)
To scale the size of **Plot Schematic**'s plot, find the scale factor in Table 25-1 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*  

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. For example, when X is -3.600, it means the origin is 3.6 inches to the left of the plot’s center.  

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 a D 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 \(-\frac{1}{2}\) 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.

Select one of the following if Send output to Plotter or Create a PLOT file is selected:

- Plotter uses single sheet paper
- 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 one of the following if Send output to Printer or Create a PRINT file is selected to specify whether wide (13-inch) paper or narrow (8-inch) paper is used to plot to a printer:

- Printer has narrow paper
- Printer has wide paper
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.

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: 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.

**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 26-1 shows a sample printout created by Print Schematic.

Figure 26-1. Output created by Print Schematic.
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 26-2).

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:

- Printer has narrow paper
- 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 27-1 shows a sample parts list created by Create Bill of Materials.

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Reference</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>R2, R1</td>
<td>1K</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>S1</td>
<td>SW SPDT</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>U1</td>
<td>74LS00</td>
</tr>
</tbody>
</table>

Figure 27-1. Bill of Materials created by Create Bill of Materials.
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
   Part Value Combine
   Include File Combine

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
   Part Value Combine
   Include File Combine

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.
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 27-2).

![Configuration Create Bill of Materials](image)

Figure 27-2. 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.
Chapter 27: Create Bill of Materials

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 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 27-3.

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.

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>Part Order Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1K' 1/4 Watt 5%</td>
<td>1000111003</td>
</tr>
<tr>
<td>'4.7K' 1/4 Watt 5%</td>
<td>10001114703</td>
</tr>
<tr>
<td>'22K' 1/4 Watt 5%</td>
<td>10001112204</td>
</tr>
<tr>
<td>'1uf' Capacitor Ceramic Disk</td>
<td>10000211006</td>
</tr>
<tr>
<td>'.1uf' Capacitor Ceramic Disk</td>
<td>10000211007</td>
</tr>
<tr>
<td>'.01uf' Capacitor Ceramic Disk</td>
<td>10000211008</td>
</tr>
<tr>
<td>'.001uf' Capacitor Ceramic Disk</td>
<td>10000211009</td>
</tr>
<tr>
<td>'7400' TTL Quad Two Input NAND Gate</td>
<td>10001040000</td>
</tr>
<tr>
<td>'74LS00' TTL Quad Two Input NAND Gate</td>
<td>10002040000</td>
</tr>
<tr>
<td>'74S00' TTL Quad Two Input NAND Gate</td>
<td>10003040000</td>
</tr>
<tr>
<td>'74ALS00' TTL Quad Two Input NAND Gate</td>
<td>10004040000</td>
</tr>
<tr>
<td>'74AS00' TTL Quad Two Input NAND Gate</td>
<td>10005040000</td>
</tr>
<tr>
<td>'7402' TTL Quad Two Input NAND Gate</td>
<td>10001040002</td>
</tr>
<tr>
<td>'74LS02' TTL Quad Two Input NAND Gate</td>
<td>10002040002</td>
</tr>
<tr>
<td>'74S02' TTL Quad Two Input NAND Gate</td>
<td>10003040002</td>
</tr>
<tr>
<td>'74ALS02' TTL Quad Two Input NAND Gate</td>
<td>10004040002</td>
</tr>
<tr>
<td>'74AS02' TTL Quad Two Input NAND Gate</td>
<td>10005040002</td>
</tr>
</tbody>
</table>

Figure 27-3. Include file format.
Chapter 27: Create Bill of Materials

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 produce a more detailed report. This report includes the title block information.

Select either 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 either of the following options:

**Report is**

- **single-spaced**

- **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 keep track of which strings in an include file don't have a corresponding match string in the design. This report can be used to find match strings in an include file that were accidentally duplicated.

- Ignore warnings
  
  Causes Create Bill of Materials to continue running when it encounters warnings, instead of halting in the middle of execution.
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 28-1 shows a sample report created by Show Design Structure.

```
<<<root>>>
  halfadd_A
[ex6b.sch] October 11, 1990
  halfadd_B
[ex6b.sch] October 11, 1990
```

*Figure 28-1. Report created by Show Design Structure.*
With the Schematic Design Tools screen displayed, select Show Design Structure. Select Local Configuration from the menu that displays.

Select Configure TREELIST. A configuration screen appears (figure 28-2).

![Figure 28-2. Show Design Structure's local configuration screen.](image)

### File Options

File Options defines the source file and its destination file.

**Source**

The Source is the name of the root sheet of the design.

**Destination**

A pathname where the output of Show Design Structure is to be placed. The default file extension is .TWG. If a path is not specified, the output displays 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
  
  Tells Show Design Structure to descend into any sheetpath parts. Without this option selected, Show Design Structure treats the sheetpath itself as a part and does not descend into the sheetpath.

- Ignore warnings
  
  Causes Show Design Structure to continue running when it encounters warnings, instead of halting in the middle of execution.
Schematic Design Tools 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 to other OrCAD tool set screens.

Part VI describes Transfer tools and provides instructions for their use.

Chapter 29: To PLD describes the transfer to the Programmable Logic Design Tools screen.

Chapter 30: To Digital Simulation describes the transfer to the Digital Simulation Tools screen.

Chapter 31: To Layout describes the transfer to the PC Board Layout Tools screen.

Chapter 32: To Main describes the transfer to the main ESP design environment screen.
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 13: 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 PLD 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 PLD 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 29-1).

The Edit Schematic Logic and Compile Schematic Logic buttons are preconfigured at OrCAD with the correct libraries for logic development and the correct settings for the PLD compiler.
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.

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 displays:

A menu with the options on and off displays. Select on to turn the process on.
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 13: Update Field Contents, substituting references to Update Field Contents with FLDSTUFF.

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 29-2).

![Configure Extract PLD](image)

Figure 29-2. 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 |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 either or both 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.

Select one of the following:

- Extract all PLD information
  
  Causes EXTRACT to extract all PLD source code from the schematic.

- Extract information on device
  
  Causes EXTRACT to extract the PLD 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.
Chapter 29: To PLD

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:

```
Extract PLD
  PLD Part Combine
  PLD Type Combine
```

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 29-3 and 29-4 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 non-programmable 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 Reference 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 29-7). 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.
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 29-4. 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**.
Figure 29-4. 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  
|test CLK  
|clear RESET  
|clear MIL  
|test CLK 30(0,1)  
|end }  

Figure 29-4. Sample EXTRACT output (continued from previous page).
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 30-1).

![Digital Simulation Tools](image)

*Figure 30-1. 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.

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 displays:

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 10: Create Netlist.
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 30-2).

![Configure IBUILD](image)

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.

![Configuration Compile Simulation Specification File](image)

*Figure 30-3. 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.
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 30-1.

<table>
<thead>
<tr>
<th>Option</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source is a stimulus file</td>
<td>FILE.AST</td>
<td>FILE.STM</td>
</tr>
<tr>
<td>Source is a trace file</td>
<td>FILE.ATR</td>
<td>FILE.TRC</td>
</tr>
<tr>
<td>Source is a breakpoint file</td>
<td>FILE.ABR</td>
<td>FILE.BRK</td>
</tr>
</tbody>
</table>

Table 30-1. Source and Destination automatically defined on Compile Simulation Specification File’s local configuration screen.
The To Layout transfer consists of four processes that are used to update the connectivity database and the layout directives given in the schematics. These processes are:

- **FLDSTUFF** loads user-defined information into part fields 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 13: Update Field Contents.

- **ANNOTATE** scans a design and 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.

- **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.
To Layout updates the schematic database, the connectivity database, and the layout directives used in PC Board Layout Tools. 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 FLDSTUFF, ANNOTATE, INET, and ILINK are each configured individually, To Layout has four 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 INET and ILINK turned on and the other processes turned off.

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 displays:

A menu with the options on and off displays. Select on to turn the process on.
Chapter 31: To Layout

Configure FLDSTUFF

With the Schematic Design Tools screen displayed, select To Layout. 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 13: Update Field Contents, substituting references to Update Field Contents with FLDSTUFF.

Configure ANNOTATE

With the Schematic Design Tools screen displayed, select To Layout. 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 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 10: 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 10: Create Netlist.
The To Main tool transfers from the Schematic Design Tools screen to the design environment’s main screen. This tool does not process or create any files and has nothing for you to configure.

With the Schematic Design Tools screen displayed, select To Main. Select Execute from the menu that displays. The design environment’s main screen (figure 32-1) displays.

Figure 32-1. Design environment main screen.
The Appendices 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.

Appendix C: Interpreting connectivity databases describes the format of the incremental connectivity database.

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.

Appendix F: File extensions gives descriptions of the file extensions used by Schematic Design Tools.
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 Release IV does not support the /F switch. Many version 3 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. Release IV uses the \LINK command to list the files. For more information, see The \LINK command in Chapter 9: Creating a netlist.

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.
## ANNOTATE source [switches]

Corresponding tool: Annotate Schematic

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/H</td>
<td>Reset reference numbers to begin with 1 for each sheet of the hierarchy.</td>
<td>□ Reset reference numbers to begin with 1 on each sheet of the hierarchy</td>
</tr>
<tr>
<td>/L</td>
<td>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.</td>
<td>□ Report the last assigned reference values</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>○ Source file is the root of the design ○ Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>Unannotate the schematic. All reference numbers are set to &quot;?&quot; and all reference letters (for multiple parts-per-package) are set to &quot;A.&quot;</td>
<td>□ Unannotate Schematic</td>
</tr>
<tr>
<td>/S</td>
<td>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.</td>
<td>□ Do NOT change the sheet number</td>
</tr>
<tr>
<td>/U</td>
<td>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.</td>
<td>○ Incremental annotation (only update reference designators shown as ?) ○ Unconditional annotation (update all reference designators)</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warning messages to be ignored.</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>
**Appendix A: Command line controls**

**BACKANNO source was/is [switches]**

Corresponding tool: Back Annotate

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>○ Source file is the root of the design ○ Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warning messages to be ignored</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>

**CLEANUP source [destination] [switches]**

Corresponding tool: Cleanup Schematic

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/E</td>
<td>Removes error markers placed on the schematic by Check Electrical Rules (INET /W Run ERC on all sheets processed).</td>
<td>□ Remove error objects from schematic sheet(s)</td>
</tr>
<tr>
<td>/G</td>
<td>Report items found to be off grid.</td>
<td>□ Report off-grid items</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>○ Source file is the root of the design ○ Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>Repeat CLEANUP if it was too large to complete in one pass.</td>
<td>□ Repeat CLEANUP if sheet is too large to complete in one pass</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>
**COMPOSER source destination [switches]**

Corresponding tool: Compile Library

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/N</td>
<td>Do not sort library as it is compiled. Instead save devices in the order in which they appear in the source file.</td>
<td>□ Do not sort the library</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
</tbody>
</table>

**CROSSREF**

Corresponding tool: Cross Reference Parts

CROSSREF no longer exists as a separate program. It is now a part of PARTLIST. See PARTLIST in this chapter for details.

**DECOMP library source [/Q]**

Corresponding tool: Decompile Library

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
</tbody>
</table>
**DRAFT filename [switches]**

Corresponding tool: Draft

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/M</td>
<td>Disable the mouse.</td>
<td>□ Disable mouse</td>
</tr>
<tr>
<td>/P</td>
<td>Disable Draft's &lt;Print Screen&gt; key function.</td>
<td>□ Disable &lt;Print Screen&gt; key function</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode without echoing tracking information on the screen.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/S</td>
<td>Slow the mouse. Used for mice that are too sensitive.</td>
<td>□ Decrease mouse sensitivity</td>
</tr>
<tr>
<td>/Y</td>
<td>Reverse the “Y” axis operation of the mouse.</td>
<td>□ Reverse “Y” axis operation of the mouse</td>
</tr>
</tbody>
</table>

**ERC source [destination] [switches]**

Corresponding tool: Check Electrical Rules

ERC no longer exists as a separate program. It is now a part of INET. See INET in this chapter for details.
**EXTRACT source [switches]**

Corresponding tool: Extract PLD, in Programmable Logic Design Tools

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into sheetpath parts.</td>
<td>✅ Descend into sheetpath parts</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>✅ Source file is the root of the design ✅ Source file is single sheet</td>
</tr>
<tr>
<td>/P</td>
<td>Extract PLD information.</td>
<td>✅ Extract all PLD information</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in the “quiet” mode,</td>
<td>✅ Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>Create a flat file listing of extracted files.</td>
<td>✅ Produce a flat-file listing of extracted files</td>
</tr>
<tr>
<td>/S &lt;arg&gt;</td>
<td>Extract information about the device specified by &lt;arg&gt;.</td>
<td>✅ Extract information on device Extract part value</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>✅ Ignore warnings</td>
</tr>
</tbody>
</table>

**FLDATTRB source partField [switches]**

Corresponding tool: Select Field View

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/I</td>
<td>Set specified field(s) to invisible.</td>
<td>✅ Set the specified field to invisible</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>✅ Source file is the root of the design ✅ Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode without echoing tracking information on the screen.</td>
<td>✅ Quiet mode</td>
</tr>
<tr>
<td>/U</td>
<td>Unconditionally set visibility parameter, even if nothing is in the field.</td>
<td>✅ Unconditionally set attribute</td>
</tr>
<tr>
<td>/V</td>
<td>Set specified field(s) to visible.</td>
<td>✅ Set the specified field to visible</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>✅ Ignore warnings</td>
</tr>
</tbody>
</table>
**Appendix A: Command line controls**

**FLDSTUFF source updatePartField updateFile [reportFile] [switches]**

Corresponding tool: Update Field Contents

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/F</td>
<td>Convert update string to uppercase.</td>
<td>Convert update string to uppercase</td>
</tr>
<tr>
<td>/I</td>
<td>Specify that this field will be invisible.</td>
<td>Set the specified field to invisible</td>
</tr>
<tr>
<td>/K</td>
<td>Keep visibility the same for all fields.</td>
<td>Leave visibility of specified field unaltered</td>
</tr>
<tr>
<td>/N</td>
<td>Convert key field match string to uppercase.</td>
<td>Convert key field match string to uppercase</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>Source file is the root of the design, Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode.</td>
<td>Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>Create a report of all FLDSTUFF activity. Enter the name of the report file to use after entering the name of the stuff file.</td>
<td>Create an update report, Destination</td>
</tr>
<tr>
<td>/U</td>
<td>Unconditionally change this field. Normally it is only changed if it is empty.</td>
<td>Unconditionally update field (Normally stuffed only if empty)</td>
</tr>
<tr>
<td>/V</td>
<td>Specify that this field will be visible.</td>
<td>Set the specified field to visible</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>Ignore warnings</td>
</tr>
</tbody>
</table>
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. The format file is the path and filename of the netlist format file. OrCAD hierarchical netlist format files have an extension of .CH. A destination must be specified. If the netlist format requires two output files, destination 2 must be present on the command line.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/2</td>
<td>Make two output files. Valid for SPICE only.</td>
<td>None</td>
</tr>
<tr>
<td>/B</td>
<td>Bypass the .INX check and force a link even if it is not required</td>
<td>□ Force HFORM to always create a formatted netlist</td>
</tr>
<tr>
<td>/I</td>
<td>Assign all unconnected pins a unique net. Valid for SPICE only.</td>
<td>□ Include unconnected pins</td>
</tr>
<tr>
<td>/L</td>
<td>Does not add the sheet number to label descriptions.</td>
<td>□ Do not append sheet number to labels</td>
</tr>
<tr>
<td>/N</td>
<td>Use node names. Valid for SPICE only.</td>
<td>□ Use node names</td>
</tr>
<tr>
<td>/O</td>
<td>Use a compiled netlist format.</td>
<td>None</td>
</tr>
<tr>
<td>/P</td>
<td>Output pin numbers instead of pin names. Valid for EDIF only.</td>
<td>□ Output pin numbers (instead of pin names)</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>
**Appendix A: Command line controls**

**IFORM source format [destination] [destination2] [switches]**

Corresponding tool: Create Netlist

IFORM is Create Netlist's incremental netlist formatter. An extension of .INS is assumed for the source. The format file is the path and filename of the netlist format file. OrCAD flat netlist format files have an extension of .CF. A destination must be specified. If the netlist format requires two output files (for example, RacalRedac), destination2 must be present on the command line.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/2</td>
<td>Make two output files. Valid for these netlist formats: Calay Mentor, PADSASC, RACALRED, SPICE, and Vectron.</td>
<td>None</td>
</tr>
<tr>
<td>/A</td>
<td>Abbreviate label descriptions Valid for Wirelist format only.</td>
<td>Abbreviate label descriptions</td>
</tr>
<tr>
<td>/B</td>
<td>Bypass the .INX check and force a link even if it is not required.</td>
<td>Force IFORM to always create a formatted netlist</td>
</tr>
<tr>
<td>/I</td>
<td>Assign all unconnected pins a unique net. Valid for these netlist formats: SPICE AlteraAD, IntelADF, Case, Hilo, PCAD.</td>
<td>Include unconnected pins</td>
</tr>
<tr>
<td>/K</td>
<td>Create CON* symbols for module ports. Valid for FutureNet format only.</td>
<td>Create CON* symbols for module ports</td>
</tr>
<tr>
<td>/L</td>
<td>Does not append the sheet number to labels. Valid for all formats except: DUMP, EEDesign, PDUMP, and VSTModel.</td>
<td>Do not append sheet number to labels</td>
</tr>
<tr>
<td>/M</td>
<td>Assign SIG* attributes to module ports. Valid for FutureNet format only.</td>
<td>Assign SIG* attributes to module ports</td>
</tr>
<tr>
<td>/N</td>
<td>FutureNet format only: Create a netlist instead of a pinlist.</td>
<td>Create a netlist (instead of a pinlist)</td>
</tr>
<tr>
<td>/N</td>
<td>AlteraAD, IntelADF, and VSTModel format only: Suppress comments.</td>
<td>Suppress comments in the netlist file</td>
</tr>
<tr>
<td>/N</td>
<td>SPICE format only: Use node names.</td>
<td>Use node names</td>
</tr>
<tr>
<td>/N</td>
<td>VSTMODEL format only: omit comments from the netlist file. Comments in the VSTMODEL format begin with a semicolon.</td>
<td>Suppress comments</td>
</tr>
<tr>
<td>/O</td>
<td>Use a compiled netlist format.</td>
<td>None</td>
</tr>
</tbody>
</table>

*continued on next page*
**IFORM source format [destination] [destination2] [switches]**

(continued from previous page)

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/P</td>
<td>EDIF and FutureNet formats: Output pin numbers instead of pin names.</td>
<td>□ Output pin numbers (instead of pin names)</td>
</tr>
<tr>
<td>/P</td>
<td>Wirelist format only: Skip any non-numerical pin numbers.</td>
<td>□ Do not output pin numbers for Grid Array parts</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/V</td>
<td>Assign FutureNet power attributes to power objects. Valid for FutureNet format only.</td>
<td>□ Assign FutureNet power attributes to power objects</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B</td>
<td>Bypass the .INX check and force a link even if it is not required.</td>
<td>□ Force ILINK to always link the database</td>
</tr>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/F</td>
<td>Produce a .LNF file (linked incremental netlist format). The .LNF file is used by PCB.</td>
<td>□ Produce a linked connectivity database Destination</td>
</tr>
<tr>
<td>/I</td>
<td>Include unconnected pins.</td>
<td>□ Report single net nodes</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode.</td>
<td>□ Quiet mode</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warning messages to be ignored.</td>
<td>□ Ignore warnings</td>
</tr>
</tbody>
</table>
**INET source [destination] [switches]**

Corresponding tool: Create Netlist and Create Hierarchical Netlist.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/A</td>
<td>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.</td>
<td>Do not report warnings</td>
</tr>
<tr>
<td>/B</td>
<td>Build the file stack from schematics rather than the .INX file.</td>
<td>Rebuild file stack</td>
</tr>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into sheetpath parts.</td>
<td>Descend into sheetpath parts</td>
</tr>
<tr>
<td>/G</td>
<td>Check worksheet for parts, sheets, labels, module ports, and power objects placed off grid. Report is placed in a .GRD file.</td>
<td>Report off-grid parts</td>
</tr>
<tr>
<td>/I</td>
<td>Assign a net name to all pins, even unconnected ones.</td>
<td>Assign a net name to all pins</td>
</tr>
<tr>
<td>/L</td>
<td>Report all connected labels and module ports. The report is placed in a .LAB file.</td>
<td>Report all connected labels and ports</td>
</tr>
<tr>
<td>/N</td>
<td>Do not rebuild the .INF file.</td>
<td>Do NOT create .INF files (Report only)</td>
</tr>
<tr>
<td>/O</td>
<td>Process one sheet only.</td>
<td>Process one sheet only (This forces Rebuild file stack on next run)</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode.</td>
<td>Quiet mode</td>
</tr>
<tr>
<td>/T</td>
<td>Rebuild the entire database, recreate all connecting database files.</td>
<td>Unconditionally process all sheets in design</td>
</tr>
<tr>
<td>/U</td>
<td>Report all unconnected wires and pins. The report is placed in a .NC file. This switch also checks for pins, module ports, and power objects that might be overlapping and reports them.</td>
<td>Report unconnected wires, pins, module ports</td>
</tr>
</tbody>
</table>

*continued on next page*
### INET source [destination] [switches]
(continued from previous page)

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/w</td>
<td>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.</td>
<td>☐ Run ERC on all sheets processed</td>
</tr>
<tr>
<td>/x</td>
<td>Check module ports against sheet nets for correctness. This is only done after all database and electrical rules checking is complete.</td>
<td>☐ Check module port connections</td>
</tr>
<tr>
<td>/z</td>
<td>Cause warnings to be ignored.</td>
<td>☐ Ignore warnings</td>
</tr>
</tbody>
</table>

### LIBARCH source [destination] [switches]

Corresponding tool: Archive Parts in Schematic

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into any parts defined as sheetpath parts.</td>
<td>☐ Descend into sheetpath parts</td>
</tr>
<tr>
<td>/L</td>
<td>Make an ASCII file consisting of the names of all parts used in a schematic. This file is called a &quot;string file.&quot; The names are delimited with single quotes, and each exists on a separate line.</td>
<td>☐ Output is a library source file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>☐ Output is a string file</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>☐ Source file is the root of the design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>☐ Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode without echoing tracking information on the screen.</td>
<td>☐ Quiet mode</td>
</tr>
<tr>
<td>/S</td>
<td>Treat the source file as a “string file.” LIBARCH constructs a library source file, ready for use by COMPOSER. IEEE parts are not supported.</td>
<td>☐ Source file is a string file</td>
</tr>
<tr>
<td>/z</td>
<td>Cause warnings to be ignored.</td>
<td>☐ Ignore warnings</td>
</tr>
</tbody>
</table>
**LIBEDIT filename.LIB [switches]**

Corresponding tool: Edit Library

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/M</td>
<td>Disable the mouse.</td>
<td>☐ Disable mouse</td>
</tr>
<tr>
<td>/P</td>
<td>Disable LIBEDIT's &lt;Print Screen&gt; key function.</td>
<td>☐ Disable &lt;Print Screen&gt; key function</td>
</tr>
<tr>
<td>/S</td>
<td>Slow the mouse. Used for mice that are too sensitive.</td>
<td>☐ Decrease mouse sensitivity</td>
</tr>
<tr>
<td>/Y</td>
<td>Reverse the &quot;Y&quot; axis operation of the mouse.</td>
<td>☐ Reverse &quot;Y&quot; axis operation of mouse</td>
</tr>
</tbody>
</table>

**LIBLIST filename.LIB [destination] [switches]**

Corresponding tool: List Libraries

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/L</td>
<td>Create a string file as the report from the library.</td>
<td>☐ Output is a string file</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>☐ Quiet mode</td>
</tr>
<tr>
<td>/S</td>
<td>Print a report showing total number of devices in a library.</td>
<td>☐ Report the total number of devices in the library</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Switch</th>
<th>BOM</th>
<th>XRF</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>✓</td>
<td>✓</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>✓</td>
<td>✓</td>
<td>Descend into sheetpath parts.</td>
<td>Descend into sheetpath parts</td>
</tr>
<tr>
<td>/E</td>
<td>✓</td>
<td>✓</td>
<td>Report unused match strings in an include file.</td>
<td>None</td>
</tr>
<tr>
<td>/I</td>
<td>✓</td>
<td></td>
<td>Specify an include file to be added to the part list.</td>
<td>Merge an include file with report Include</td>
</tr>
<tr>
<td>/L</td>
<td>✓</td>
<td>✓</td>
<td>Specifies that each part starts on a new line.</td>
<td>Place each part entry on a separate line</td>
</tr>
<tr>
<td>/N</td>
<td></td>
<td>✓</td>
<td>Sort output by name, then reference designator. If neither /R or /N is present, then both reports are done (one sorted by name and the other sorted by reference).</td>
<td>Sort output by part value, then by reference designator</td>
</tr>
<tr>
<td>/O</td>
<td>✓</td>
<td>✓</td>
<td>Treat this file as a single sheet.</td>
<td>Source file is the root of the design Source file is a single sheet</td>
</tr>
<tr>
<td>/P</td>
<td></td>
<td>✓</td>
<td>Output the X,Y coordinates for all parts.</td>
<td>Report the X and Y grid coordinates for all parts</td>
</tr>
<tr>
<td>/Q</td>
<td>✓</td>
<td>✓</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>✓</td>
<td></td>
<td>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).</td>
<td>Sort output by reference designator</td>
</tr>
<tr>
<td>/S</td>
<td>✓</td>
<td>✓</td>
<td>Produce a single-spaced report rather than a double.</td>
<td>Report is single-spaced double-spaced</td>
</tr>
</tbody>
</table>

*continued on next page*
PARTLIST source [destination] [include] [switches]
(continued from previous page)

<table>
<thead>
<tr>
<th>Switch</th>
<th>BOM</th>
<th>XRF</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/T</td>
<td>✓</td>
<td>✓</td>
<td>Report type mismatches for parts, such as mixing parts with single and multiple parts per package (U1 and U1A).</td>
<td>✗ Report type mismatch parts</td>
</tr>
<tr>
<td>/U</td>
<td>✓</td>
<td>✓</td>
<td>List all unused parts in multiple-part packages.</td>
<td>✗ Report unused parts in multiple-part packages</td>
</tr>
<tr>
<td>/V</td>
<td>✓</td>
<td>✓</td>
<td>Output report in a verbose format.</td>
<td>✗ Verbose report</td>
</tr>
<tr>
<td>/W</td>
<td>✓</td>
<td>✓</td>
<td>Check for identical part references. There should be none.</td>
<td>✗ Report identical part reference designators</td>
</tr>
<tr>
<td>/X</td>
<td>✓</td>
<td>✓</td>
<td>Print cross reference report.</td>
<td>None</td>
</tr>
</tbody>
</table>
| /Y     | ✓   | ✓   | Turns off the header at the beginning of each page of the report. | ✗ Insert a header for each page
|        |     |     |                          | ✗ Do not insert a header for each page |
| /Z     | ✓   | ✓   | Cause warnings to be ignored. | ✗ Ignore warnings |
**PLOTALL source [destination] [switches]**

**Corresponding tool: Plot Schematic**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into sheet path parts.</td>
<td>✗ Descend into sheetpath parts</td>
</tr>
<tr>
<td>/G</td>
<td>Plot grid references around the border of output.</td>
<td>✗ Plot grid references around the worksheet border</td>
</tr>
<tr>
<td>/N</td>
<td>Instruct PLOTALL to ignore “fill” commands.</td>
<td>✗ Ignore “fill” commands</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>✗ Source file is the root of the design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Source file is a single sheet</td>
</tr>
<tr>
<td>/P</td>
<td>Direct output to printer instead of plotter.</td>
<td>✗ Send output to printer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Send output to plotter</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in “quiet” mode without echoing tracking information on the screen.</td>
<td>✗ Quiet mode</td>
</tr>
<tr>
<td>/R</td>
<td>Specify plotter using roll feed paper.</td>
<td>✗ Plotter uses roll feed paper</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Plotter uses single sheet paper</td>
</tr>
<tr>
<td>/S</td>
<td>Specify a scale factor for plot output “###.###”.</td>
<td>✗ Manually set scale factor and/or X,Y offsets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Set Scale factor</td>
</tr>
<tr>
<td>/U</td>
<td>Use offsets for another paper size (for example, /UA means to use A-size paper).</td>
<td>✗ Produce 1:1 scale plot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Automatically scale and set X, Y offsets for specified sheet size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ A O B C O D O E</td>
</tr>
<tr>
<td>/W</td>
<td>Specify wide paper in printer.</td>
<td>✗ Printer has wide paper</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Printer has narrow paper</td>
</tr>
<tr>
<td>/X</td>
<td>Specify an “X” offset for plot output “###.###”.</td>
<td>✗ Manually set scale factor and/or X,Y offsets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Set X, Y offsets X</td>
</tr>
<tr>
<td>/Y</td>
<td>Specify a “Y” offset for plot output “###.###”.</td>
<td>✗ Manually set scale factor and/or X,Y offsets</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✗ Set X, Y offsets Y</td>
</tr>
</tbody>
</table>
PRINTALL source [destination] [switches]

Corresponding tool: Print Schematic

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into sheetpath parts.</td>
<td>Descend into sheetpath parts</td>
</tr>
<tr>
<td>/G</td>
<td>Print grid references around the border of the output.</td>
<td>Print grid references around the worksheet border</td>
</tr>
<tr>
<td>/N</td>
<td>Suppress pin numbers.</td>
<td>Suppress pin numbers on print</td>
</tr>
<tr>
<td>/O</td>
<td>Treat this file as a single sheet.</td>
<td>Source file is the root of the design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source file is a single sheet</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>Quiet mode</td>
</tr>
<tr>
<td>/W</td>
<td>Specify wide paper in printer.</td>
<td>Printer has narrow paper</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Printer has wide paper</td>
</tr>
</tbody>
</table>
### SIMPLE source destination [switches]

Corresponding tool: Complex to Simple in Design Management Tools

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Corresponding options (no local configuration)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display the SDT configuration screen.</td>
<td>None</td>
</tr>
<tr>
<td>/N</td>
<td>Do not descend into sheetpath parts.</td>
<td>None</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>None</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>None</td>
</tr>
</tbody>
</table>

### TREELIST source destination [switches]

Corresponding tool: Show Schematic Structure

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Local Configuration Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>/C</td>
<td>Display SDT configuration information.</td>
<td>None</td>
</tr>
<tr>
<td>/D</td>
<td>Descend into sheetpath parts.</td>
<td>☐ Descend into sheetpath parts</td>
</tr>
<tr>
<td>/Q</td>
<td>Run in &quot;quiet&quot; mode without echoing tracking information on the screen.</td>
<td>☐ Quiet mode</td>
</tr>
<tr>
<td>/Z</td>
<td>Cause warnings to be ignored.</td>
<td>☐ Ignore warnings</td>
</tr>
</tbody>
</table>
This chapter discusses the various netlist format files that OrCAD provides. These format files are selected on the local configurations of Create Netlist and Create Hierarchical Netlist. See Chapter 10: Create Netlist and Chapter 11: 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.

The following sections contain descriptions and examples of the flat and hierarchical netlist format files supplied with Schematic Design Tools. 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 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.

Two versions of each netlist format file are included: a compiled version and an uncompiled version written in a high-level programming language. Using the compiled version reduces the time IFORM and HFORM take to create netlists. You can view and edit uncompiled netlist format files with Edit File.
The uncompiled versions are useful because you can customize the netlist format file. These netlist format files are written in a small, interpreted language whose syntax is similar to the programming language "C." Appendix C explains the syntax of the language, and all of the built-in functions and symbols.

\[\checkmark\] NOTE: You may want to create a new directory called \ORCADESP\SDT\NETFORMS\SOURCE and copy the .CF and .CH files into that directory.

Each uncompiled netlist format file includes a comment section at the top that lists the restrictions imposed by that particular netlist format. The same information is included in this appendix.

This table lists the extensions used for netlist format files:

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type of netlist format file</th>
</tr>
</thead>
<tbody>
<tr>
<td>.CCF</td>
<td>Compiled flat</td>
</tr>
<tr>
<td>.CCH</td>
<td>Compiled hierarchical</td>
</tr>
<tr>
<td>.CF</td>
<td>Uncompiled flat</td>
</tr>
<tr>
<td>.CH</td>
<td>Uncompiled hierarchical</td>
</tr>
</tbody>
</table>

*Table B-1. Netlist format file types and extensions.*
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:

<table>
<thead>
<tr>
<th>Create Netlist</th>
<th>Part Value Combine</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Value Combine</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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 (1st part field, represented by "1" in the example above) is shown in bold italics.

For more information about configuring key fields, see the section Key Fields in chapter 1.

Flat netlists

To create a netlist in one of these flat formats, it is assumed that your schematic has already been simplified (if necessary), netlisted, and linked. These steps are fully explained in Chapter 10: Create Netlist.

Flat netlist format files are distinguished by a .CCF file extension. 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.
Figure B.1. Used by most of the flat netlist formats.
Figure B-2. Used by the Alter-ADF and Intel-ADF formats.
Figure B.4. Used by the flat SPICE format.

```
VCC DC -12
RC1 10K
RC2 10K
CLOAD1 5PF
RS2 1K
RS1 1K
VIN 10 5MEG
VIN AC 1 SIN(0 0.1 5MEG)
Q2 QNL
Q3 QNL
Q4 QNL
Q1 QNL
VCC DC -12
VCC
RBIAS1 20K
VEE DC -12

.SPICE
.OPTIONS ACCT LIST NODE OPTS NOPAGE RELTOL=.001
.WIDTH OUT=.80
.DC VIN -0.25 0.25 0.005
.AC DEC 10 1 100Hz
.TRAN OP SNS 500ns
.MODEL QNL NPN (BF=80 VAF=0.3PF)
.PRINT DC V(4) V(5)
.PLOT DC IC(V2)
.PROBE
```

OrCAD
3175 NW Aloclek Drive
Hillsboro, OR 97124-7135
(503) 690-9881 Sales & Administration
(503) 690-9722 Technical Support
Title
SPICE Example
Size
Document Number
A
OrCAD-04
A
Date: October 30, 1990
Sheet 1 of 1
Figure B-5. Used by the OrCAD/PLD netlist format.
### Appendix B: Netlist formats

<table>
<thead>
<tr>
<th>Format Specific Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Do not append sheet number to labels</td>
</tr>
</tbody>
</table>

**Algorex (ALGOREX.CCF)**

The ALGOREX.CCF 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 six digits including the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.
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-6 was created—with no options selected—from the schematic in figure B-1.

<table>
<thead>
<tr>
<th>Net</th>
<th>Description</th>
<th>Sheet Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>N00001</td>
<td>U1 (14DIP300)-8, U2 (14DIP300)-1</td>
<td></td>
</tr>
<tr>
<td>Q-1</td>
<td>U1 (14DIP300)-6, U2 (14DIP300)-2, U1 (14DIP300)-1</td>
<td></td>
</tr>
<tr>
<td>N00003</td>
<td>U1 (14DIP300)-5, U1 (14DIP300)-3</td>
<td></td>
</tr>
<tr>
<td>VCC</td>
<td>U2 (14DIP300)-</td>
<td></td>
</tr>
<tr>
<td>14,</td>
<td>U1 (14DIP300)-14</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>U1 (14DIP300)-9, U1 (14DIP300)-10</td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>U2 (14DIP300)-7, U1 (14DIP300)-7</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>U1 (14DIP300)-4</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>U2 (14DIP300)-3</td>
<td></td>
</tr>
<tr>
<td>CLOCK</td>
<td>U1 (14DIP300)-2</td>
<td></td>
</tr>
</tbody>
</table>

Figure B-6. Example netlist in the Algorex format.
Appendix B: Netlist formats

Allegro (ALLEGRO.CCF)  The ALLEGRO.CCF 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 six digits including 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-7 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-7. Example netlist in the Allegro format.
### Schematic Design Tools Reference Guide

**AlteraADF**  
(ALTERAAD.CCF)  
The ALTERAAD.CCF 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, pin names, node numbers, and pin numbers are not checked for length.

- All ASCII characters are legal.

**Format Specific Options**

- **Suppress comments in the netlist file**
  
  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.

- **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.

**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.
### Table B-2. Title block information in AlteraADF netlists.

<table>
<thead>
<tr>
<th>Line</th>
<th>Example Header</th>
<th>Title Block Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADF Example</td>
<td>Title of sheet</td>
</tr>
<tr>
<td>1</td>
<td>October 11, 1990</td>
<td>Date</td>
</tr>
<tr>
<td>2</td>
<td>OrCAD-02</td>
<td>Document Number</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Revision Code</td>
</tr>
<tr>
<td>3</td>
<td>OrCAD</td>
<td>Organization Name</td>
</tr>
<tr>
<td>4</td>
<td>3175 NW Aloclek Drive</td>
<td>1st Address Line</td>
</tr>
<tr>
<td>6</td>
<td>Turbo = ON</td>
<td>3rd Address Line</td>
</tr>
<tr>
<td>7</td>
<td>5C031</td>
<td>4th Address Line</td>
</tr>
</tbody>
</table>

**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 (`|`). On many PC keyboards, the pipe character appears as a broken vertical bar. 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 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 and the Altera software. Schematic Design Tools 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 pre-defined levels in the Altera software. Because Schematic Design Tools 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-8 was created—with no options selected—from the schematic in figure B-2.
Figure B-8. Example netlist in the AlteraADF format.
AppliconBRAVO
(APPLBRAV.CCF)
The APPLBRAV.CCF format file is used to produce netlists in the AppliconBRAV format.

AppliconBRAV 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 six digits including the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

<table>
<thead>
<tr>
<th>Format Specific Options</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not append sheet number to labels</td>
<td>The netlist in figure B-9 was created—with no options selected—from the schematic in figure B-1.</td>
</tr>
</tbody>
</table>

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

```
*** Design 14DIP300
U1
*** Design 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-9. Example netlist in the AppliconBRAVO format.**
The APPLLEAP.CCF 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 (plus the leading "N").
- 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.

![Figure B-10. Example netlist in the AppliconLEAP format.](image-url)
| **Cadnetix (CADNETIX.CCF)** | The CADNETIX.CCF 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-11 was created—with no options selected—from the schematic in figure B-1. |
**Figure B-11. Example netlist in the Cadnetix format.**
Calay (CALAY.CCF)  The CALAY.CCF 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 six digits including 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  The Calay format creates two files. In addition to the netlist file, it also creates a component file. 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-12 and its corresponding component file in figure B-13 were created—with no options selected—from the schematic in figure B-1.
Appendix B: Netlist formats

Figure B-12. Example netlist in the Calay format.

```
/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 component file in the Calay format.

```
74L800   U1    14DIP300  000 000 0
74L832   U2    14DIP300  000 000 0
```
The CALAY.CCF 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-14 and its corresponding component file in figure B-15 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-14. Example netlist in the new Calay format.

<table>
<thead>
<tr>
<th>74LS00</th>
<th>U1</th>
<th>14DIP300</th>
<th>000</th>
<th>000</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS32</td>
<td>U2</td>
<td>14DIP300</td>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure B-15. Example component file in the new Calay format.
Case (CASE.CCF) The CASE.CCF 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 (plus the leading "N").
- 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-16 was created—with no options selected—from the schematic in figure B-1.
Figure B-16. Example netlist in the Case format.
CBDS (CBDS.CCF) The CBDS.CCF format file is used to produce netlists in the CBDS format.

CBDS netlists have the following characteristics:
- Part names, module names, and reference strings are not checked for length.
- 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 six digits including 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-17 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-17. Example netlist in the CBDS format.
The COMPVISN.CCF 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 (plus the leading "N").
- Pin names are not used.
- All ASCII characters are legal.

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.

The netlist in figure B-18 was created—with no options selected—from the schematic in figure B-1.

<table>
<thead>
<tr>
<th>Number</th>
<th>Part Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>N00001</td>
<td>U1-8 U2-1</td>
</tr>
<tr>
<td>0002</td>
<td>Q/1</td>
<td>U1-6 U2-2 U1-1</td>
</tr>
<tr>
<td>0003</td>
<td>N00003</td>
<td>U1-5 U1-3</td>
</tr>
<tr>
<td>0004</td>
<td>VCC</td>
<td>U2-14 U1-14</td>
</tr>
<tr>
<td>0005</td>
<td>A</td>
<td>U1-9 U1-10</td>
</tr>
<tr>
<td>0006</td>
<td>GND</td>
<td>U2-7 U1-7</td>
</tr>
<tr>
<td>0007</td>
<td>B</td>
<td>U1-4</td>
</tr>
<tr>
<td>0008</td>
<td>OUT</td>
<td>U2-3</td>
</tr>
<tr>
<td>0009</td>
<td>CLOCK</td>
<td>U1-2</td>
</tr>
</tbody>
</table>

Figure B-18. Example netlist in the ComputerVision format.
**DUMP (DUMP.CCF)**  The DUMP.CCF 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 (EDIF.CCF)  Two EDIF (Version 2.0.0) formats are supplied with Schematic Design Tools: EDIF.CCF, used to create flat EDIF netlists (described here) and EDIF.CCH, 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 (plus the leading "N").
- Legal characters are:
  
  0 .. 9  a .. z  A .. Z  _ (underscore)

  Case is not significant. When IFORM encounters an illegal character, it issues a warning and makes the following changes:

  - Changes "-" to "MINUS"
  - Changes "+" to "PLUS"
  - Changes "\" to "BAR"
  - Changes all other illegal characters to "_"

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.

- Output pin numbers (instead of pin names)
  Select this option if you want the netlist to contain pin numbers instead of pin names. If IFORM encounters a pin without a number, it still outputs the pin name. If you require a netlist consisting entirely of pin numbers, you may need to modify the library part.
Example The netlist in figure B-19 was created—with no options selected—from the schematic in figure B-1.

```plaintext
(edif &EX1
(edifVersion 2 0 0)
(edifLevel 0)
(keywordMap (keywordLevel 0))
(status
(written
(timeStamp 0 0 0 0 0)
(comment "Original data from OrCAD/SDT schematic")
)
(comment "Original data from OrCAD/SDT schematic")
(comment "Generic Netlist Example")
(comment * November 7, 1990*)
(comment "OrCAD-01")
(comment "A")
(comment "OrCAD")
(comment "3175 NW Alcolek Drive")
(comment "Hillsboro, OR 97124-7135")
(comment "(503) 690-9881 Sales & Administration")
(comment "(503) 690-9722 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))))
(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)))
```

Figure B-19. Example netlist in the EDIF format (continued).
(port &II_B (direction INPUT))
(port &II_C (direction INPUT))
(port &II_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 &Ul
(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 &Ul))
(portRef &I0_A (instanceRef &U2)))
(net &Q_1
(joined
(portRef &O_B (instanceRef &Ul))
(portRef &I0_A (instanceRef &U1))
(portRef &II_A (instanceRef &U2)))
(net N00003
(joined
(portRef &II_B (instanceRef &Ul))
(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 &II_C (instanceRef &U1)))
(net &GND
(joined

Figure B-19. Example netlist in the EDIF format (continued).
(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-19. Example netlist in the EDIF format.
EEDesigner (EEDESIGN.CCF)

The EEDESIGN.CCF 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 four digits including the "N" 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-20 was created from the schematic in figure B-1.

```plaintext
{ 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-20. Example netlist in the EEDesigner format.
The FUTURE.CCF 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, pin numbers, and node names are not checked for length.
- Reference strings are limited to six characters.
- Node numbers are limited to six digits (plus the leading "****").
- Characters are not checked for legality.

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 power objects. The pins on the following Schematic Design Tools power objects (which are matched by name) are assigned FutureNet power attributes.

<table>
<thead>
<tr>
<th>OrCAD Pin Value</th>
<th>FutureNet Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>100</td>
</tr>
<tr>
<td>+5V</td>
<td>101</td>
</tr>
<tr>
<td>+12V</td>
<td>105</td>
</tr>
<tr>
<td>-12V</td>
<td>106</td>
</tr>
<tr>
<td>VEE</td>
<td>107</td>
</tr>
</tbody>
</table>

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-21 was created—with no options selected—from the schematic in figure B-1. The netlist in figure B-22 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, 74LS80
    DATA, 4, 14DIP300
    PIN, Q_1, 1-1, 1-5, 23, 10_A
    PIN, CLOCK, 1-1, 1-5, 23, 11_A
    PIN, ***000003, 1-1, 1-5, 21, 0_A
    PIN, B, 1-1, 1-5, 23, 10_B
    PIN, ***000003, 1-1, 1-5, 23, 11_B
    PIN, Q_1, 1-1, 1-5, 21, 0_B
    PIN, GND, 1-1, 1-5, 23, GND
    PIN, ***000001, 1-1, 1-5, 21, 0_C
    PIN, A, 1-1, 1-5, 23, 10_C
    PIN, A, 1-1, 1-5, 23, 11_C
    PIN, UN000001, 1-1, 1-5, 21, 0_D
    PIN, UN000002, 1-1, 1-5, 23, 10_D
    PIN, UN000003, 1-1, 1-5, 23, 11_D
    PIN, VCC, 1-1, 1-5, 23, VCC
    )
    (SYM, 2
    DATA, 2, U2
    DATA, 3, 74LS32
    DATA, 4, 14DIP300
    PIN, ***000001, 1-1, 1-5, 23, 10_A
    PIN, Q_1, 1-1, 1-5, 23, 11_A
    PIN, OUT, 1-1, 1-5, 21, 0_A
    PIN, UN000004, 1-1, 1-5, 23, 10_B
    PIN, UN000005, 1-1, 1-5, 23, 11_B
    PIN, UN000006, 1-1, 1-5, 21, 0_B
    PIN, GND, 1-1, 1-5, 23, GND
    PIN, UN000007, 1-1, 1-5, 21, 0_C
    PIN, UN000008, 1-1, 1-5, 23, 10_C
    PIN, UN000009, 1-1, 1-5, 23, 11_C
    PIN, UN000010, 1-1, 1-5, 21, 0_D
    PIN, UN000011, 1-1, 1-5, 23, 10_D
    PIN, UN000012, 1-1, 1-5, 23, 11_D
    PIN, VCC, 1-1, 1-5, 23, VCC
    )
    SIG, Q_1, 1-1, 1-5, Q_1
    SIG, VCC, 1-1, 1-5, VCC
    SIG, A, 1-1, 1-5, A
    SIG, GND, 1-1, 1-5, GND
    SIG, B, 1-1, 1-5, B
    SIG, OUT, 1-1, 1-5, OUT
    SIG, CLOCK, 1-1, 1-5, CLOCK
    )
```

Figure B-21. Example pinlist in the FutureNet netlist format.
Figure B-22. Example netlist in the FutureNet netlist format (continued).
Figure B-22. Example netlist in the FutureNet netlist format.
HiLo (HILO.CCF)  The HILO.CCF format file is used to produce netlists in the HiLo format.

HiLo netlists have the following characteristics:

- Part names, module names, pin numbers, and reference strings are not checked for length.
- Node names are limited to 14 characters.
- Node numbers are limited to 6 digits, including 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-23 was created—with no options selected—from the schematic in figure B-1.
** Generic Netlist Example Revised:  October 30, 1990
** OrCAD-01
** OrCAD
** 3175 NW Aloclek Drive
** Hillsboro, OR 97124-7135
** (503) 690-9881 Sales & Administration
** (503) 690-9722 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-23. Example netlist in the HiLo format.
The INTELADF.CCF 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**

- 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.

- 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.

- 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.
### 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

When you create an IntelADF netlist, you must configure Schematic Design Tools to use the OrCAD-supplied ADF.LIB library. You can only use the parts provided in the ADF library to create the schematic.

Inputs and outputs are handled differently in OrCAD's Schematic Design Tools and the Intel software. Schematic Design Tools 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 does not default unconnected pins to any particular level, you must tie all unused pins to the appropriate level.

---

**Table B-4. Title block information in IntelADF netlists.**

<table>
<thead>
<tr>
<th>Line</th>
<th>Example</th>
<th>Header</th>
<th>Title Block Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADF</td>
<td>Example</td>
<td>Title of sheet</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>October 11, 1990</td>
<td>Date</td>
</tr>
<tr>
<td>2</td>
<td>OrCAD</td>
<td>-02</td>
<td>Document Number</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td></td>
<td>Revision Code</td>
</tr>
<tr>
<td>3</td>
<td>OrCAD</td>
<td></td>
<td>Organization Name</td>
</tr>
<tr>
<td>4</td>
<td>3175 NW</td>
<td>Aloclek Drive</td>
<td>1st Address Line</td>
</tr>
<tr>
<td>6</td>
<td>Turbo</td>
<td>= ON</td>
<td>3rd Address Line</td>
</tr>
<tr>
<td>7</td>
<td>5C031</td>
<td></td>
<td>4th Address Line</td>
</tr>
</tbody>
</table>
Example

The netlist in figure B-24 was created—with no options selected—from the schematic in figure B-2.

```
ADF Example
30, 1990
OrCAD-02
OrCAD-3175 NW Alcove Drive
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=INF(COINDROP) % SYM 7 %
D.1=INF(CLOCK) % SYM 8 %
N.1=INF(CUPFULL) % SYM 9 %
I.1=INF(RESET) % SYM 10 %
J.1=INF(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-24. Example netlist in the IntelADF format.
The INTERGRA.CCF 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 six digits, including 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-25 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-25. Example netlist in the Intergraph format.*
Mentor (MENTOR.CCF)

The MENTOR.CCF format file is used to produce netlists in the Mentor format.

Mentor netlists have the following characteristics:

- Node names and pin numbers are not checked for length.
- Part names, module names, and reference strings are limited to nineteen characters.
- Node numbers are limited to six digits including the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

File Options

IFORM creates two files: a netlist file and a component file. 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-26 and the corresponding component file in figure B-27 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-26. Example netlist in the Mentor format.
Appendix B: Netlist formats

<table>
<thead>
<tr>
<th>Field</th>
<th>Reference</th>
<th>Value Field</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>PART 74L800</td>
<td>14DIP300</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>PART 74L832</td>
<td>14DIP300</td>
<td></td>
</tr>
</tbody>
</table>

Figure B-27. Example component file in the Mentor format.
MultiWire
(MULTIWIR.CCF)
The MULTIWIR.CCF format file is used to produce netlists in the MultiWire format.

MultiWire netlists have the following characteristics:

- Part 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 six digits including 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-28 was created—with no options selected—from the schematic in figure B-1.

<table>
<thead>
<tr>
<th>Node</th>
<th>Part</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N00001</td>
<td>U1</td>
<td>8</td>
</tr>
<tr>
<td>N00001</td>
<td>U2</td>
<td>1</td>
</tr>
<tr>
<td>Q-1</td>
<td>U1</td>
<td>6</td>
</tr>
<tr>
<td>Q-1</td>
<td>U2</td>
<td>2</td>
</tr>
<tr>
<td>Q-1</td>
<td>U1</td>
<td>1</td>
</tr>
<tr>
<td>N00003</td>
<td>U1</td>
<td>5</td>
</tr>
<tr>
<td>N00003</td>
<td>U2</td>
<td>3</td>
</tr>
<tr>
<td>VCC</td>
<td>U1</td>
<td>14</td>
</tr>
<tr>
<td>VCC</td>
<td>U2</td>
<td>14</td>
</tr>
<tr>
<td>A</td>
<td>U1</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>U2</td>
<td>10</td>
</tr>
<tr>
<td>GND</td>
<td>U1</td>
<td>7</td>
</tr>
<tr>
<td>GND</td>
<td>U2</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>U1</td>
<td>4</td>
</tr>
<tr>
<td>OUT</td>
<td>U2</td>
<td>3</td>
</tr>
<tr>
<td>CLOCK</td>
<td>U1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure B-28. Example netlist in the MultiWire format.
The PCBII.CCF 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 31: 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 pin numbers are not checked for length.
- Node names are limited to eight characters.
- Node numbers are limited to six digits including the "N" prefix.
- Pin names are not used.
- 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-29 was created—with no options selected—from the schematic in figure B-1.
Figure B-29. Example netlist in the OrCAD/PCB II format.
The PLDNET.CCF 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 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.
- Node numbers are limited to six characters including 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

When you create a PLD netlist, you must configure Schematic Design Tools 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-30 was created—with no options selected—from the schematic in figure B-5.
Figure B-30. Example netlist in the PLD Netlist format.
The VSTMODEL.CCF 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.
- 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-29).

**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 (|). (On many PC keyboards, the pipe character displays as a broken vertical bar.) 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 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-31 was created—with no options selected—from the schematic in figure B-3.

```
; TO 1 MULTIPLEXOR
; 2MUX MY_LIB 4
LINV(P3;L1);M1
LAND(P2,P1;L2);M2
LMOR(L1,P1;L3);M3
SET(RHIGH)
OR(L2,L3;P4;10,50,15,50);M4
%
```

Figure B-31. Example netlist in the VSTModel format.
The PADSASC.CCF 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 twelve characters.
- Node names are limited to six characters.
- Node numbers are limited to six digits including the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal except:

  Node names can use only these characters:
  
  ~ ! # $ % _ = + | / . : ; < >
  A...Z a...z 0..9

  Reference strings can use only these characters:

  A...Z a...z 0..9

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.

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-32 and the corresponding component file in figure B-33 were created—with no options selected—from the schematic in figure B-1.

```
*PADS-PCB
*Net
*Sig N1
U1.8 U2.1
*Sig Q-l
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-32. Example connection list in the PADS ASCII format.

```
*PADS-PCB
*Part*
U1  14DIP300
U2  14DIP300
*End
```

Figure B-33. Example part list in the PADS ASCII format.
Appendix B: Netlist formats

PADS ASCII (PADSASCØ.CCF) The PADSASCØ.CCF format file is used to produce netlists in the PADS ASCII format. The resulting netlist is identical to the netlist IFORM creates using PADSASC.CCF. The only difference is that node names may have up to twelve characters instead of six as described in the previous section.
PCAD (PCAD.CCF)  The PCAD.CCF 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 eight characters including 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-34 was created—with no options selected—from the schematic in figure B-1.
Figure B-34. Example netlist in the PCAD format.
**PCADnlt (PCADNLT.CCF)**

The EEDESIGN.CCF format file is used to produce netlists in the EEDesigner 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:
  - A-Z a-z 0-9 $ - + (underscore)
- Node numbers are limited to six digits including 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-35 was created—with no options selected—from the schematic in figure B-1.
% Generic Netlist Example Revised: October 30, 1990
% OrCAD-01 Revision: A
% OrCAD
% 3175 NW Alocleak Drive
% Hillsboro, OR 97124-7135
% (503) 690-9881 Sales & Administration
% (503) 690-9722 Technical Support
BOARD = ORCAD.PCB;

<table>
<thead>
<tr>
<th>PARTS</th>
<th>NETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>14DIP300</td>
<td>74LS00</td>
</tr>
</tbody>
</table>

BOARD = ORCAD.PCB;

<table>
<thead>
<tr>
<th>NET</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_1</td>
<td>U1/6</td>
</tr>
<tr>
<td>N00003</td>
<td>U1/5</td>
</tr>
<tr>
<td>N00001</td>
<td>U1/8</td>
</tr>
<tr>
<td>VCC</td>
<td>U2/14</td>
</tr>
<tr>
<td>A</td>
<td>U1/9</td>
</tr>
<tr>
<td>GND</td>
<td>U2/7</td>
</tr>
<tr>
<td>B</td>
<td>U1/4</td>
</tr>
<tr>
<td>OUT</td>
<td>U2/3</td>
</tr>
<tr>
<td>CLOCK</td>
<td>U1/2</td>
</tr>
</tbody>
</table>

Figure B-35. Example netlist in the PCADnlt format.
Schematic Design Tools Reference Guide

PDUMP (PDUMP.CCF)  The PDUMP.CCF 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.
The RACALRED.CCF 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 six characters including the “N” prefix.
- Pin names are not used.
- All ASCII characters are legal.

IFORM creates two files. In addition to the netlist file, it also creates a component file. You must enter a second filename in the Destination 2 entry box on the Configure Netlist Format screen.

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.

The netlist in figure B-36 and the corresponding component file in figure B-37 were created—with no options selected—from the schematic in figure B-1.
Figure B-36. Example netlist in the RacalRedac format.

```
.PCB
.REM Generic Netlist Example Revised: October 30, 1990
.REM OrCAD-01 Revision: A
.REM OrCAD
.REM 3175 NW Aloclek Drive
.REM Hillsboro, OR 97124-7135
.REM (503) 690-9881 Sales & Administration
.REM (503) 690-9722 Technical Support
.COM
.REF 2
.REM NOOOOI
U1 8 U2 1
.REM Q_1 (local to sheet 1)
U1 6 U2 2 U1 1
.REM NO0003
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-37. Example component list in the RacalRedac format.

```
.PCB
.REM Generic Netlist Example Revised: October 30, 1990
.REM OrCAD-01 Revision: A
.REM OrCAD
.REM 3175 NW Aloclek Drive
.REM Hillsboro, OR 97124-7135
.REM (503) 690-9881 Sales & Administration
.REM (503) 690-9722 Technical Support
.COM
.REF
.REM 74LS00
U1 14DIP300
.REM 74LS32
U2 14DIP300
.EOD
```
### Scicards (SCICARDS.CCF)

The SCICARDS.CCF format file is used to produce netlists in the Scicards format.

Scicards netlists have the following characteristics:
- Node numbers are not checked for length.
- 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 3 characters.
- Node names are limited to sixteen characters.
- 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-38 was created—with no options selected—from the schematic in figure B-1.
### Parts List

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>U1</th>
<th>U2</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS00</td>
<td></td>
<td>14DIP300</td>
<td>U1</td>
</tr>
<tr>
<td>74LS32</td>
<td></td>
<td>14DIP300</td>
<td>U2</td>
</tr>
</tbody>
</table>

### Net List

<table>
<thead>
<tr>
<th>Node</th>
<th>Node Name</th>
<th>U1</th>
<th>U2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>U1</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>U1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>U2</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>U1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>U2</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>U1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Figure B-38. Example netlist in the Scicards format.**
The SPICE.CCF 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 characters from this subset of ASCII characters if the option Use node names is selected:
  
  0..9 A..Z a..z  $  _ (underscore)

- All ASCII characters are legal except as noted for node numbers.

**File Options**

IFORM creates two files. In addition to the netlist file, it also creates a map file. 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.

\[\Delta\] **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.
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.

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.

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 (|). (On many PC keyboards, the pipe character appears as a broken vertical bar.) 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 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). Therefore, the library source file for an NPN transistor that is compatible with the SPICE pin numbering convention is as follows:

```
'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-39. Library source file for the NPN transistor.
Examples

The netlist in figure B-40 and the corresponding map file in figure B-41 were created—with no options selected—from the schematic in figure B-4.

```
* SPICE Example Revised: October 30, 1990
* OrCAD-04 Revision: A
* OrCAD
* 3175 NW Aloncle Drive
* Hillsboro, OR 97124-7135
* (503) 690-9881 Sales & Administration
* (503) 690-9722 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 .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-40. Example netlist in the SPICE format.

```
10001 4
10002 5
10005 6
10006 7
10007 VCC DC 12
10008 VIN AC 1 SIN(0 .1 5MEG)
 0 GND
10010 VEE DC -12
```

Figure B-41. Example map file in the SPICE format.
Tango (TANGO.CCF)  The TANGO.CCF format file is used to produce netlists in the Tango format.

Tango netlists have the following characteristics:

- Pin numbers are not checked for length.
- Part names, module names, reference strings, and node names are limited to 16 characters.
- Node numbers are limited to six digits including the "N" prefix.
- Pin names are not used.
- Reference strings and module names must be only upper case 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-42 was created—with no options selected—from the schematic in figure B-1.
Figure B-42. Example netlist in the Tango format.
Telesis (TELESIS.CCF) The TELESIS.CCF 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 six digits including 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-43 was created—with no options selected—from the schematic in figure B-1.

```plaintext
$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-43. Example netlist in the Telesis format.
The VECTRON.CCF 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 six digits including the "N" prefix.
- Pin names are not used.
- All ASCII characters are legal.

IFORM creates two files. In addition to the netlist file, it also creates a part list file. You must enter a second filename in the Destination 2 entry box on the Configure Netlist Format screen.

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.

The netlist in figure B-44 and the corresponding part file in figure B-45 were created—with no options selected—from the schematic in figure B-1.
Appendix B: Netlist formats

<table>
<thead>
<tr>
<th>Netlist entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*NO0001</td>
<td>U1 8 U2 1</td>
</tr>
<tr>
<td>*Q.1</td>
<td>U1 6 U2 2 U1 1</td>
</tr>
<tr>
<td>*NO0003</td>
<td>U1 5 U1 3</td>
</tr>
<tr>
<td>*VCC</td>
<td>U2 14 U1 14</td>
</tr>
<tr>
<td>*A</td>
<td>U1 9 U1 10</td>
</tr>
<tr>
<td>*GND</td>
<td>U2 7 U1 7</td>
</tr>
<tr>
<td>*B</td>
<td>U1 4</td>
</tr>
<tr>
<td>*OUT</td>
<td>U2 3</td>
</tr>
<tr>
<td>*CLOCK</td>
<td>U1 2</td>
</tr>
</tbody>
</table>

**Figure B-44.** Example netlist in the Vectron format.

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>14DIP300</td>
</tr>
<tr>
<td>U2</td>
<td>14DIP300</td>
</tr>
</tbody>
</table>

**Figure B-45.** Example part list file in the Vectron format.
The WIRELIST.CCF format file is used to produce netlists in the WireList format.

WireList netlists have the following characteristics:
- Part 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 six digits including 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 option Do not output numbers for Grid Array parts 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.

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 do not select this option, IFORM does not output pin numbers for grid array parts.

Abbreviate label descriptions

If you select this option, IFORM shortens label descriptions.
Appendix B: Netlist formats

Example

The netlist in figure B-46 was created—with no options selected—from the schematic in figure B-1.

Wire List

Generic Netlist Example
Revised: October 30, 1990
OrCAD-01
Revision: A
OrCAD
3175 NW Aloclek Drive
Hillsboro, OR 97124-7135
(503) 690-9881 Sales & Administration
(503) 690-9722 Technical Support

<<< Component List >>>

<table>
<thead>
<tr>
<th>PART</th>
<th>VALUE</th>
<th>PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS00</td>
<td>U1</td>
<td>14DIP300</td>
</tr>
<tr>
<td>74LS32</td>
<td>U2</td>
<td>14DIP300</td>
</tr>
</tbody>
</table>

<<< Wire List >>>

<table>
<thead>
<tr>
<th>NODE</th>
<th>VALUE</th>
<th>REFERENCE</th>
<th>PIN #</th>
<th>PIN NAME</th>
<th>PIN TYPE</th>
<th>PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>[00001]</td>
<td>N00001</td>
<td>U1</td>
<td>8</td>
<td>O_C</td>
<td>Output</td>
<td>74LS00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U2</td>
<td>1</td>
<td>I0_A</td>
<td>Input</td>
<td>74LS32</td>
</tr>
<tr>
<td>[00002]</td>
<td>O_1 (local to sheet 1)</td>
<td>U1</td>
<td>6</td>
<td>O_B</td>
<td>Output</td>
<td>74LS00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U2</td>
<td>2</td>
<td>I1_A</td>
<td>Input</td>
<td>74LS32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U1</td>
<td>1</td>
<td>I0_A</td>
<td>Input</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00003]</td>
<td>N00003</td>
<td>U1</td>
<td>5</td>
<td>I1_B</td>
<td>Input</td>
<td>74LS00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U1</td>
<td>3</td>
<td>O_A</td>
<td>Output</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00004]</td>
<td>VCC</td>
<td>U2</td>
<td>14</td>
<td>VCC</td>
<td>Power</td>
<td>74LS32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U1</td>
<td>14</td>
<td>VCC</td>
<td>Power</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00005]</td>
<td>A</td>
<td>U1</td>
<td>9</td>
<td>I0_C</td>
<td>Input</td>
<td>74LS00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U1</td>
<td>10</td>
<td>I1_C</td>
<td>Input</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00006]</td>
<td>GND</td>
<td>U2</td>
<td>7</td>
<td>GND</td>
<td>Power</td>
<td>74LS32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U1</td>
<td>7</td>
<td>GND</td>
<td>Power</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00007]</td>
<td>B</td>
<td>U1</td>
<td>4</td>
<td>I0_B</td>
<td>Input</td>
<td>74LS00</td>
</tr>
<tr>
<td>[00008]</td>
<td>OUT</td>
<td>U2</td>
<td>3</td>
<td>O_A</td>
<td>Output</td>
<td>74LS32</td>
</tr>
<tr>
<td>[00009]</td>
<td>CLOCK</td>
<td>U1</td>
<td>2</td>
<td>I1_A</td>
<td>Input</td>
<td>74LS00</td>
</tr>
</tbody>
</table>

Figure B-46. 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 11: Create Hierarchical Netlist.

Hierarchical netlist format files are distinguished by a .CCH file extension. They cannot be used within the Create Netlist processor (the two interpreter 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-47 and B-48 are used by the hierarchical EDIF and SPICE netlists.
Figure B.47. Root sheet used by the hierarchical EDIF format.

![Diagram of a half adder circuit with carry inputs and outputs.]

**Appendix B: Netlist formats**

<table>
<thead>
<tr>
<th>Title</th>
<th>Hierarchy (Complex) Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size Document Number</td>
<td>OrCAD-06</td>
</tr>
<tr>
<td>Date</td>
<td>October 30, 1990</td>
</tr>
</tbody>
</table>

OrCAD
3175 NW Aloclak Drive
Hillsboro, OR 97124-7135
(503) 690-9881 Sales & Administration
(503) 690-9722 Technical Support

REV A
Sheet 1 of 3
Figure B-49. Root sheet used by the hierarchical SPICE format.
Figure B-50. Lead sheet used by the hierarchical SPICE format.
EDIF (EDIF.CCH)  This is one of two EDIF (Version 2.0.0) netlist format files supplied with OrCAD's Schematic Design Tools. 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. All of the options that apply to the flat EDIF format apply to the hierarchical version as well. See the section EDIF (EDIF.CCF) for information about 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 requires 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-51 was created—with no options selected—from the schematic in figure B-46.
(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 "November 7, 1990")
(comment "OrCAD-06")
(comment "A")
(comment "OrCAD")
(comment "3175 NW Alcolek Drive")
(comment "Hillsboro, OR 97124-7135")
(comment "/(503) 690-9881 Sales & Administration")
(comment "(503) 690-9722 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))
(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))

Figure B-51. Example netlist in the hierarchical EDIF format (continued).
Appendix B: Netlist formats

Figure B-51. Example netlist in the hierarchical EDIF format (continued).
```plaintext
(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 &IIB (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 &I0D (instanceRef &U3))))
(net &GND
  (joined
    (portRef &GND (instanceRef &U4))
    (portRef &GND (instanceRef &U1))
    (portRef &GND (instanceRef &U3))
    (portRef &GND (instanceRef &U2))))
(net &SUM
  (joined
    (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

Figure B-51. Example netlist in the hierarchical EDIF format (continued).

562
Appendix B: Netlist formats

```
(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)
                (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-51. Example netlist in the hierarchical EDIF format.
The HDUMP.CCH 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.

The SPICE.CCH 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 (SPICE.CCF) for information about netlist characteristics and configuration options.

HFORM uses SPICE.CCH 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.

The netlist in figure B-52 and the corresponding map file in figure B-53 were created—with no options selected—from the schematic in figure B-47.
Appendix B: Netlist formats

Figure B-52. 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-53. Example map file in the hierarchical SPICE format.
This appendix covers the following topics:

- Introduces connectivity databases
- Describes in detail the format of incremental and linked connectivity databases that are used with OrCAD's tools
- Includes 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.

Two of the databases, the *incremental connectivity database* and the *linked connectivity database*, organize information about a design's connectivity. Basic information about these databases is contained in *Chapter 9: Creating a netlist*.

The information in these two databases 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.

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 each database.
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.

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.
About the linked connectivity database

Using the incremental connectivity database as input, ILINK, depending on the local configuration options selected, creates one or two databases: an intermediate netlist structure or an intermediate netlist structure and a linked connectivity database.

The linked connectivity database is in the same format as the .INF files, but is a single ASCII file with the .LNF extension. OrCAD's PC Board Layout Tools Release IV uses the linked connectivity database.

Typographical conventions

This appendix contains several typographical, or notational, conventions that are in addition to or used differently than the rest of this reference guide. The conventions are:

\textit{Italic text} 

This italic font shows a parameter representing specific data.

\ldots Ellipses shows the continuation of a series.

\texttt{"parameter"} 

Double quotes surrounding a parameter indicate the parameter is a quoted token. The double quotes are part of the syntax.

\texttt{?parameter?} 

Question marks surrounding a parameter indicate the parameter can be either a number or a quoted token. The question marks are \textit{not} part of the syntax.

\texttt{parameter} 

Parameters without double quotes or question marks are either numbers or strings. Parameter definitions specify the type of token.

Terminology

This section includes definitions for the various parts of .INF and .LNF 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.
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.

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.

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:

<table>
<thead>
<tr>
<th>Quoted token</th>
<th>Corresponds to</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OUTPUT2&quot;</td>
<td>OUTPUT2</td>
</tr>
<tr>
<td>&quot;&quot;&quot;SEC&quot;&quot; Carry&quot;&quot;</td>
<td>&quot;SEC&quot; Carry</td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>(empty field)</td>
</tr>
</tbody>
</table>

Examples of quoted tokens.

A string is an unquoted token that may contain only these characters:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1 2 3 4 5 6 7 8 9 0
= - ! # $ % ' , . : ; = ? @ \ _ * { } ! ~ /
```

Strings may not contain spaces. For example:

Clock
42_22V10
@17#sec?

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:

```
BEFHIJKLMNOPSTVW
```

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:

```
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef
```

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:

```
`WS "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 and .LNF 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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Name</th>
<th>Frequency in .INF files</th>
<th>Frequency in .LNF files</th>
</tr>
</thead>
<tbody>
<tr>
<td>'H or 'F</td>
<td>Header</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>'B</td>
<td>Title block</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>'L</td>
<td>Link</td>
<td>Zero or more</td>
<td>None</td>
</tr>
<tr>
<td>'P</td>
<td>Port</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'S</td>
<td>Signal</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'E</td>
<td>External</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'I</td>
<td>Instance</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'J</td>
<td>Joined</td>
<td>Zero or more</td>
<td>One or more</td>
</tr>
<tr>
<td>'K</td>
<td>Layout</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'T</td>
<td>Trace</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'V</td>
<td>Vector</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'W</td>
<td>Stimulus</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
<tr>
<td>'l</td>
<td>Pipe</td>
<td>Zero or more</td>
<td>Zero or more</td>
</tr>
</tbody>
</table>

Order and frequency of statements in .INF and .LNF 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 or .LNF 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 LP"
"3175 NW Alolelek Drive" "Hillsboro, Oregon 97124"
"(503) 690-9881" "`

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; .LNF files contain no link statements. 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.
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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Module port type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional, IO</td>
</tr>
<tr>
<td>U</td>
<td>Unspecified</td>
</tr>
<tr>
<td>S</td>
<td>Supply</td>
</tr>
</tbody>
</table>

*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"
```
Signal statements define signal names and the worksheets containing them. The syntax is:

\`
'S "signal_name" sheet_number
``

\`S \nBegins a signal definition.

"signal_name" \nTells the name of the signal.

sheet_number \nEquals the number of the worksheet containing the signal. A whole decimal number.

**Example**

\`
'S "TRYIN1" 2
``

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 \nBegins the external statement.

library_name \nTells the name of a library. A string.

**Example**

\`
'E TTL.LIB
``

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:
Appendix C: Interpreting connectivity databases

`IR "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 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.
Appendix C: Interpreting connectivity databases

?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.

<table>
<thead>
<tr>
<th>Character</th>
<th>Pin type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional, IO</td>
</tr>
<tr>
<td>S</td>
<td>Supply, power</td>
</tr>
<tr>
<td>P</td>
<td>Passive</td>
</tr>
<tr>
<td>T</td>
<td>Three-state</td>
</tr>
<tr>
<td>C</td>
<td>Open collector</td>
</tr>
<tr>
<td>E</td>
<td>Open emitter</td>
</tr>
</tbody>
</table>

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 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.

"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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Sheet net type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional, IO</td>
</tr>
<tr>
<td>S</td>
<td>Supply, power</td>
</tr>
<tr>
<td>P</td>
<td>Passive</td>
</tr>
<tr>
<td>T</td>
<td>Three-state</td>
</tr>
<tr>
<td>C</td>
<td>Open collector</td>
</tr>
<tr>
<td>E</td>
<td>Open emitter</td>
</tr>
<tr>
<td>U</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>

Characters for sheet net types.
Appendix C: Interpreting connectivity databases

Join

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

`I C "SECOND.SCH" 54AF3C22 "SECOND_1" ( "A0" I ) ( "A1" B ) ( "A2" U )`
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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Module port types</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional</td>
</tr>
<tr>
<td>U</td>
<td>Unspecified</td>
</tr>
<tr>
<td>S</td>
<td>Supply, power</td>
</tr>
</tbody>
</table>

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.
Appendix C: Interpreting connectivity databases

?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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Pin type</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional, IO</td>
</tr>
<tr>
<td>S</td>
<td>Supply, power</td>
</tr>
<tr>
<td>P</td>
<td>Passive</td>
</tr>
<tr>
<td>T</td>
<td>Three-state</td>
</tr>
<tr>
<td>C</td>
<td>Open collector</td>
</tr>
<tr>
<td>E</td>
<td>Open emitter</td>
</tr>
</tbody>
</table>

Characters for pin types.

) Ends a pin description.

Example

( R COAX1 "CENTER 1" 0 ) ( 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.
<table>
<thead>
<tr>
<th>Character</th>
<th>Sheet net types</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Input</td>
</tr>
<tr>
<td>O</td>
<td>Output</td>
</tr>
<tr>
<td>B</td>
<td>Bidirectional</td>
</tr>
<tr>
<td>S</td>
<td>Supply, power</td>
</tr>
<tr>
<td>P</td>
<td>Passive</td>
</tr>
<tr>
<td>T</td>
<td>Three-state</td>
</tr>
<tr>
<td>C</td>
<td>Open collector</td>
</tr>
<tr>
<td>E</td>
<td>Open emitter</td>
</tr>
<tr>
<td>U</td>
<td>Unspecified</td>
</tr>
</tbody>
</table>

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.
Appendix C: Interpreting connectivity databases

"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)"
Appendix C: Interpreting connectivity databases

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.
Appendix C: Interpreting connectivity databases

"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.

<table>
<thead>
<tr>
<th>Character</th>
<th>Display type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Binary bus</td>
</tr>
<tr>
<td>D</td>
<td>Decimal bus</td>
</tr>
<tr>
<td>H</td>
<td>Hexadecimal bus</td>
</tr>
<tr>
<td>O</td>
<td>Octal bus</td>
</tr>
</tbody>
</table>

Characters for bus types.

"bus_display_name"

Tells the name of the bus to display during simulation.

△ NOTE: If no value for display_type is provided, INET assigns hexadecimal.

Example

'T B "AD[0..7]" 4 H "ADDRESS"
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.
Appendix C: Interpreting connectivity databases

`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 text vector. A whole decimal number.

*Example*

`'V B "AD[0..7]" 4 14`
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.

<table>
<thead>
<tr>
<th>Character</th>
<th>State function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Set to value 0</td>
</tr>
<tr>
<td>1</td>
<td>Set to value 1</td>
</tr>
<tr>
<td>U</td>
<td>Set to undefined state</td>
</tr>
<tr>
<td>Z</td>
<td>Set to high impedance state</td>
</tr>
<tr>
<td>T</td>
<td>Toggle the signal state</td>
</tr>
</tbody>
</table>

*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.
\( \text{\textbf{Signal}} \)  

The syntax of a signal stimulus is:

`W S "signal_name" sheet_number (stimuli)

`W \hspace{1cm} \text{Begins parameters describing stimulus information.}

S \hspace{1cm} \text{Means the following parameters are for signals.}

"signal_name" \hspace{1cm} \text{Tells the name of the signal.}

\( \text{\textbf{NOTE: Digital Simulation Tools does not recognize signal names containing periods (.) anywhere in the signal name or colons (:)} \text{ at the end of the signal name.} \)

sheet_number \hspace{1cm} \text{Tells the number of the worksheet containing the signal. A whole decimal number.}

( \hspace{1cm} \text{Begins a list of stimuli.}

stimuli \hspace{1cm} \text{Lists one or more set or branch parameters.}

) \hspace{1cm} \text{Ends a list of stimuli.}

Example

`W S "INPUT" 3 ( 0:0 50:1 100:T 200:G:50 )

\( \text{\textbf{Pin}} \)  

The syntax of a pin stimulus is:

`W R part_reference ?pin_number? (stimuli)
Begins a stimulus description.

R Means the following parameters describe a pin.

\textit{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.

\textit{stimuli} Lists one or more set or branch parameters.

) Ends a list of stimuli.

\textit{Example}

```
`W R U1 3 ( 0:1 50:T )
```

\textbf{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.

\textit{"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.
Appendix C: Interpreting connectivity databases

△ **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 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 or .LNF 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 should begin in a different column.

**Example**

```
| PLDX1 in:(A1, A2, B1, B2),
| io:(Y1, Y2),
| Y1 = A1 & B1,
| Y2 = A2 # B2
```
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.
<table>
<thead>
<tr>
<th>Comments</th>
<th>Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST_INF.SCH is a hierarchy Title block information</td>
<td>'H 1.00 TEST-INF</td>
</tr>
<tr>
<td></td>
<td>'B 1&quot; * 2&quot; * A&quot; * February 19, 1991&quot; * &quot;1.0&quot;</td>
</tr>
<tr>
<td></td>
<td><em>SAMPLE SCHEMATIC FOR .inf FORMAT</em> &quot;OrCAD L.P.&quot;</td>
</tr>
<tr>
<td></td>
<td><em>3175 NW ALOCLEK DRIVE</em> &quot;HILLSBORO, OREGON 97124&quot;</td>
</tr>
<tr>
<td></td>
<td><em>U.S.A.</em> <em>(503)690-9881</em></td>
</tr>
<tr>
<td></td>
<td>'L inf#2</td>
</tr>
<tr>
<td></td>
<td>'P S &quot;VCC&quot;</td>
</tr>
<tr>
<td></td>
<td>'P I &quot;INPUT_PORT&quot;</td>
</tr>
<tr>
<td></td>
<td>'P S &quot;GND&quot;</td>
</tr>
<tr>
<td></td>
<td>'P I &quot;BUS0&quot;</td>
</tr>
<tr>
<td></td>
<td>'P I &quot;BUS1&quot;</td>
</tr>
<tr>
<td></td>
<td>'S &quot;TO_LED&quot; 1</td>
</tr>
<tr>
<td></td>
<td>'E TTL.LIB</td>
</tr>
<tr>
<td></td>
<td>'E DEVICE.LIB</td>
</tr>
<tr>
<td></td>
<td>'I R &quot;LED&quot; DEVICE.LIB &quot;LED&quot; 7B0738E9 D1 [1] ** ** ** **</td>
</tr>
<tr>
<td></td>
<td>** &quot;7TH PF&quot; &quot;8TH PF&quot; ** &quot;ANODE&quot; ( &quot;ANODE&quot; P )</td>
</tr>
<tr>
<td></td>
<td>( &quot;CATHODE&quot; &quot;CATHODE&quot; P )</td>
</tr>
<tr>
<td></td>
<td>'I R &quot;R330&quot; DEVICE.LIB &quot;R&quot; 7B0738E8 R1 [1] ** ** ** &quot;4TH PF&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;5TH PF&quot; &quot;6TH PF&quot; ** ** ** ( &quot;1&quot; &quot;1&quot; P ) ( &quot;2&quot; &quot;2&quot; P )</td>
</tr>
<tr>
<td></td>
<td>'I R &quot;74F352&quot; TTL.LIB &quot;74F352&quot; 7B07BEA3 U1 [1]</td>
</tr>
<tr>
<td></td>
<td>&quot;1ST PF&quot; &quot;2ND PF&quot; &quot;3RD PF&quot; ** ** ** **</td>
</tr>
<tr>
<td></td>
<td>( &quot;1CO&quot; 6 I ) ( &quot;1C1&quot; 5 I ) ( &quot;1C2&quot; 4 I ) ( &quot;1C3&quot; 3 I )</td>
</tr>
<tr>
<td></td>
<td>( &quot;2CO&quot; 10 I ) ( &quot;2C1&quot; 11 I ) ( &quot;2C2&quot; 12 I ) ( &quot;2C3&quot; 13 I )</td>
</tr>
<tr>
<td></td>
<td>( &quot;1G&quot; 1 I ) ( &quot;2G&quot; 15 I ) ( &quot;A&quot; 14 I ) ( &quot;B&quot; 2 I )</td>
</tr>
<tr>
<td></td>
<td>( &quot;1Y&quot; 7 O ) ( &quot;2Y&quot; 9 O ) ( &quot;VCC&quot; 16 S ) ( &quot;GND&quot; 8 S )</td>
</tr>
<tr>
<td></td>
<td>'I C &quot;914NFIW01.sch&quot; 7B5427A5 &quot;SHEET&quot; (&quot;FROM_CHILD&quot; O )</td>
</tr>
<tr>
<td></td>
<td>'J ( P S &quot;VCC&quot; ) ( R U1 16 S )</td>
</tr>
<tr>
<td></td>
<td>'J ( P I &quot;BUS0&quot; ) ( R U1 6 I )</td>
</tr>
<tr>
<td></td>
<td>'J ( P I &quot;BUS1&quot; ) ( R U1 5 I )</td>
</tr>
<tr>
<td></td>
<td>'J ( P I &quot;INPUT_PORT&quot; ) ( R U1 10 I )</td>
</tr>
<tr>
<td></td>
<td>'J ( R U1 9 O ) ( S &quot;TO_LED&quot; 1 ) ( R R1 &quot;1&quot; P )</td>
</tr>
<tr>
<td></td>
<td>'J ( R U1 1 I )</td>
</tr>
<tr>
<td></td>
<td>'J ( C &quot;SHEET&quot; &quot;FROM_CHILD&quot; O ) ( R U1 2 I )</td>
</tr>
<tr>
<td></td>
<td>'J ( R D1 &quot;CATHODE&quot; P ) ( P S &quot;GND&quot; ) ( R U1 8 S )</td>
</tr>
<tr>
<td></td>
<td>'J ( R R1 2&quot; P ) ( R D1 &quot;ANODE&quot; P )</td>
</tr>
<tr>
<td></td>
<td>'T B &quot;BUS[0..10]&quot; 1 H &quot;TRACENAME&quot;</td>
</tr>
<tr>
<td></td>
<td>'V B &quot;BUS[0..10]&quot; 1 5</td>
</tr>
<tr>
<td></td>
<td>'K B &quot;BUS[0..10]&quot; 1 &quot;PATTERN(TREE)&quot;</td>
</tr>
<tr>
<td></td>
<td>'W B &quot;BUS[0..10]&quot; 1 ( 2:200 )</td>
</tr>
<tr>
<td></td>
<td>'W S &quot;INPUT_PORT&quot; 1 ( 0:0 0:G:0 0:G:10 )</td>
</tr>
<tr>
<td></td>
<td>'V S &quot;INPUT_PORT&quot; 1 10</td>
</tr>
<tr>
<td></td>
<td>'T S &quot;INPUT_PORT&quot; 1 &quot;INPUT_PORT&quot;</td>
</tr>
<tr>
<td></td>
<td>'K S &quot;INPUT_PORT&quot; 1 &quot;WIDTH(.010)&quot;</td>
</tr>
</tbody>
</table>

Sample .INF file (continued).
**Vector, stimulus, trace, and layout information from a part**

Pipe text from the worksheet

<table>
<thead>
<tr>
<th>Pipe Text</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>V R U 1 2 6</code></td>
<td><code>V R U 1 2 (STIM_CHILD)</code></td>
</tr>
<tr>
<td><code>W R U 1 2 (STIM_CHILD)</code></td>
<td><code>T R U 1 2 &quot;TRACE_CHILD&quot;</code></td>
</tr>
<tr>
<td><code>T R U 1 2 &quot;TRACE_CHILD&quot;</code></td>
<td><code>K R U 1 2 &quot;STRATEGY(EXTENSIVE)&quot;</code></td>
</tr>
<tr>
<td><code>K R U 1 2 &quot;STRATEGY(EXTENSIVE)&quot;</code></td>
<td>`I TEXT FOR ANOTHER PROGRAM OR OrCAD TOOL</td>
</tr>
<tr>
<td><code>I TEXT FOR ANOTHER PROGRAM OR OrCAD TOOL&quot;</code></td>
<td><code>) RESIDES ON THIS SCHEMATIC&quot;</code></td>
</tr>
</tbody>
</table>

Sample .INF file.
Both .INF and .LNF files are in the .INF format described in this appendix. However, when ILINK links a series of .INF files to create a .LNF file, it uses the following guidelines during the process.

It is possible that sub-parts from a single package reside on separate schematics and have conflicting information in the part fields. When this happens, ILINK merges the instance commands for the sub-parts, using the first parameter containing information. ILINK combines the pin number parameters, discarding duplicates such as power and ground pins.

For example, if the following two instance commands are in separate .INF files processed by ILINK, the .LNF file contains the third instance command.

The instance command below is from the first .INF file.

```
'I R 74LS21 TTL.LIB *74LS21* 34D55A33 U1 [10] ** **
*14PDIP* ** ** ** ** **14PDIP* ( "IOA" 1 I )
( "I1A" 2 I ) ( "I2A" 4 I ) ( "I3A" 5 I )
( "OA" 6 0 ) ( "GND" 7 S ) ( "VCC" 14 S )
```

This instance command is from the second .INF file:

```
'I R 74LS21 TTL.LIB *74LS21* 34D55A16 U1 [01] ** **
*16PDIP* ** "JOE" ** ** "16PDIP" ( "IOB" 9 I )
( "I1B" 10 I ) ( "I2B" 12 I ) ( "I3B" 13 I )
( "OB" 8 0 ) ( "GND" 7 S ) ( "VCC" 14 S )
```

This is the resulting instance command in the .LNF file:

```
'I R 74LS21 TTL.LIB *74LS21* 34D55A33 U1 [11] ** **
*14PDIP* ** "JOE" ** ** "14PDIP" ( "IOA" 1 I )
( "I1A" 2 I ) ( "I2A" 4 I ) ( "I3A" 5 I ) ( "OA" 6 0 )
( "GND" 7 S ) ( "VCC" 14 S ) ( "IOB" 9 I )
( "I1B" 10 I ) ( "I2B" 12 I ) ( "I3B" 13 I )
( "OB" 8 0 )
```

Notice that ILINK used the latest absolute identifier, 34D55A33, and the first occurrences of the part fields and module field.
Creating a custom netlist format

This appendix describes the process of creating a custom netlist format to use with OrCAD's Schematic Design Tools. 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. The figure below shows both processes.

![Diagram showing the processes of IFORM and HFORM](image)

Figure D-1. Source and destination files for IFORM and HFORM.

To create a custom netlist format file, you should be familiar with programming, preferably in the C language.
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 9, 10, and 11 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 predefined 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. 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.
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 in the Schematic Design Tools Reference Guide 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 .CF or .CH files and compare them to the netlists they create.

Examine the .CF or .CH 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 .CF or .CH file. When necessary, refer to the syntax and reference information later in this appendix.

5. Make the changes.

Edit a copy of the .CF or .CH file that is nearest to what you want using a text editor or Edit File. Be sure to name it something different.

6. 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.
7. 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 matches your specification.

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 the extension .CF for flat netlists IFORM creates and .CH for hierarchical netlists HFORM creates.

Language

The language used in format files is a subset of the C language. Local and global variables are supported, as well as function calls and recursion. The following list contains exceptions, restrictions, and conventions:

- Int and char are the only data types supported.
- All functions are assumed to return int.
- Argument passing to user-defined functions is not supported.
- If, if/else, while, do-while, and for statements are supported.
- All blocks must begin with an opening curly brace ( { ) and end with a closing curly brace ( } ).
- Nesting is supported. Curly braces surround nested statements.
- These operations are supported:
  - +  -  /  *  unary +
  - =  ==  !=  %  unary 1
  - >  <  >=  <=
- The ASCII character set is supported except that symbol names may not contain double quotes.
Appendix D: Creating custom netlist formats

- Bit and logical operations are not supported.
- Arrays and pointers are not supported.
- Quoted strings are supported.
- Quoted strings can contain these escapes:

  \n  new line

  \b  backspace

  \t  tab

  \f  form feed

  \r  return

- Quoted strings can contain backslashes (\) to indicate the next character is to appear within the string. For example, this represents a double quote (" ) in a string:

  \\

  This represents a backslash in a string:

  \\n
Functions

IFORM and HFORM recognize both a subset of 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 pre-defined 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 .CF 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)
        {
            case '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 (continued).
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 from previous page).
/* 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 .CH files, see the section Required functions.

Following the pseudocode is a “Hello World” example.

```c
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.
Schematic Design Tools Reference Guide

/* 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:

<table>
<thead>
<tr>
<th>Required functions for IFORM</th>
<th>Required functions for HFORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>HandleNodeName();</td>
<td>Initialize();</td>
</tr>
<tr>
<td>Initialize();</td>
<td>PreFile();</td>
</tr>
<tr>
<td>ProcessFieldStrings();</td>
<td>PostFile();</td>
</tr>
<tr>
<td>WriteHeader();</td>
<td>PostProcess();</td>
</tr>
<tr>
<td>WriteNet();</td>
<td></td>
</tr>
<tr>
<td>WriteNetEnding();</td>
<td></td>
</tr>
<tr>
<td>WriteNetListEnd();</td>
<td></td>
</tr>
</tbody>
</table>
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 .CH 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 rolls 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 .CH format files for HFOM are:

```c
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:

```c
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 are OrCAD-defined functions. Some access data structures and some do not. The general functions are:

- AddSymbol\( (\text{string}\_\text{constant}) \)
- ClearSymbolicStrings();
- int CompareSymbol(\( \text{symbol}, \text{symbol} \));
- ConcatFile (file_index, file_index);
- CopySymbol(\( \text{symbol}, \text{user}\_\text{symbol} \));
- ExceptionsFor(\( \text{string}\_\text{constant}, \text{user}\_\text{symbol} \));
- int FindSymbolChar(\( \text{variable}, \text{user}\_\text{symbol} \));
- int GetStandardSymbol(\( \text{standard}\_\text{symbol} \));
- int GetSymbolChar(\( \text{variable}, \text{user}\_\text{symbol} \));
- MakeLocalSignal(\( \text{string}\_\text{constant} \));
- int PackString(\( \text{integer}\_\text{constant}, \text{integer}\_\text{constant}, \text{user}\_\text{symbol}, \text{user}\_\text{symbol} \));
- int PackString(\( \text{variable}, \text{variable}, \text{user}\_\text{symbol}, \text{user}\_\text{symbol} \));
- PadSpaces(\( \text{variable}, \text{variable}, \text{user}\_\text{symbol}, \text{user}\_\text{symbol} \));
- PadSpaces(\( \text{user}\_\text{symbol}, \text{variable} \));
- PutSymbolChar(\( \text{variable}, \text{integer}\_\text{constant}, \text{user}\_\text{symbol} \));
- PutSymbolChar(\( \text{variable}, \text{variable}, \text{user}\_\text{symbol} \));
- SetAccessType(\( \text{user}\_\text{symbol} \));
- SetCharSet(\( \text{string}\_\text{constant} \));
- SetNumberWidth(\( \text{integer}\_\text{constant} \));
- SetNumberWidth(\( \text{variable} \));
- SetPinMap(\( \text{integer}\_\text{constant}, \text{string}\_\text{constant} \));
- SetPinMap(\( \text{variable}, \text{string}\_\text{constant} \));
- SetSymbol(\( \text{user}\_\text{symbol}, \text{string}\_\text{constant} \));
- StripPath(\( \text{string}\_\text{symbol} \));
- int SwitchIsSet(\( \text{string}\_\text{constant} \));
- int SymbolInCharSet(\( \text{user}\_\text{symbol} \));
- int SymbolLength(\( \text{symbol} \));
- ToUpper(\( \text{user}\_\text{symbol} \));
- WriteCrLf(\( \text{file}\_\text{index} \));
- WriteInteger(file_index, integer_constant);
- WriteInteger(file_index, variable)
- WriteMap(\( \text{file}\_\text{index}, \text{integer}\_\text{constant} \));
- WriteMap(\( \text{file}\_\text{index}, \text{variable} \));
- WriteString(\( \text{file}\_\text{index}, \text{string}\_\text{constant} \));
- WriteSymbol(\( \text{file}\_\text{index}, \text{user}\_\text{symbol} \));
- WriteStdSymbol(\( \text{file}\_\text{index}, \text{standard}\_\text{symbol} \));

For more about these functions and symbols, see the reference portion of this appendix.
You can use these C-language functions in netlist format files:

```c
int getche();
int getnum();
int print( variable );
int print( string_constant );
int putch( variable )
int puts( string_constant );
```

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 .CF or .CH 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.

\[ \text{**NOTE:** Do not use these switches that IFORM or HFORM already use: } /B, /Q, \text{ and } /Z. \]
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.
<table>
<thead>
<tr>
<th>EXIT TYPE</th>
<th>IGNORE WARNINGS OPTION</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>ON</td>
<td>Exits</td>
</tr>
<tr>
<td>W</td>
<td>OFF</td>
<td>Exits reporting a warning</td>
</tr>
<tr>
<td>E</td>
<td>ON or OFF</td>
<td>Exits reporting an error</td>
</tr>
<tr>
<td>nothing</td>
<td>ON or OFF</td>
<td>Exits</td>
</tr>
</tbody>
</table>

Results of combinations of EXIT TYPE 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 EXIT TYPE; it is already defined.

The function ClearSymbolicStrings() does not clear EXIT TYPE.

**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.
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookupNameString</td>
<td><code>symbol</code></td>
<td>String. Contains the name used to look up the current part in the part library indicated by LibraryNameString.</td>
</tr>
<tr>
<td>ModuleName</td>
<td><code>symbol</code></td>
<td>String. This is the module name obtained from the module value field.</td>
</tr>
</tbody>
</table>
| 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         | `symbol`      | Unsigned integer. The current net number. IFORM and HFORM assign unique net numbers to all nodes in a net.                                    |
| **NetType** | symbol for | Character. Contains the signal type. This may be one of:
|            | HFORM      | 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 0xFFFFH 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 0xFFFFH or not 0xFFFFH. |
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>Field</th>
<th>Standard Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PinNameString</td>
<td>standard symbol</td>
<td>String. Contains the pin name string for the current instance of a node.</td>
</tr>
<tr>
<td>PinNumberString</td>
<td>standard symbol</td>
<td>String. Contains the string representation of the pin number for the current instance of a node.</td>
</tr>
<tr>
<td>PinType</td>
<td>standard symbol</td>
<td>Character. Contains one of these types for pins, ports and signals, or sheets:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>Pin</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BiDirectional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Open Collector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: Passive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Hi-Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6: Open Emitter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7: Power</td>
</tr>
<tr>
<td>Port &amp; signal</td>
<td></td>
<td>- <strong>Unspecified</strong></td>
</tr>
<tr>
<td>Sheet</td>
<td></td>
<td>- <strong>Unspecified</strong></td>
</tr>
<tr>
<td>PipeLine</td>
<td>standard symbol</td>
<td>String. This contains the current line in the pipe file.</td>
</tr>
<tr>
<td>ReferenceString</td>
<td>standard symbol</td>
<td>String. Contains the reference designator for either the instance of a part or of a node.</td>
</tr>
<tr>
<td>Revision</td>
<td>title block symbol</td>
<td>String. The revision number from the worksheet.</td>
</tr>
<tr>
<td><strong>SheetNumber</strong></td>
<td><strong>title block symbol</strong></td>
<td>Unsigned integer. Contains the current worksheet number. In IFORM, if the current TypeCode is &quot;L&quot;, <strong>SheetNumber</strong> reflects the number of the sheet containing the signal. If TypeCode is any other value, <strong>SheetNumber</strong> may not be accurate because it is updated only if the current signal net has a label. In HFORM, <strong>SheetNumber</strong> is always correct.</td>
</tr>
<tr>
<td><strong>SheetSize</strong></td>
<td><strong>title block symbol</strong></td>
<td>Character. Contains the size of the worksheet.</td>
</tr>
<tr>
<td><strong>SignalNameString</strong></td>
<td><strong>standard symbol</strong></td>
<td>String. May contain a signal name. If TypeCode is &quot;L&quot;, &quot;P&quot;, or &quot;S&quot;, then <strong>SignalNameString</strong> contains the signal name. The <strong>SignalNameString</strong> is empty otherwise.</td>
</tr>
<tr>
<td><strong>SignalType</strong></td>
<td><strong>standard symbol</strong></td>
<td>Character. Contains the signal type. The signal types are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  Unspecified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1  Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2  Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3  Bidirectional</td>
</tr>
<tr>
<td><strong>TimeStamp</strong></td>
<td><strong>standard symbol</strong></td>
<td>String. Contains the hexadecimal time stamp for the part. The hexadecimal time stamp has no suffix to indicate the number is hexadecimal.</td>
</tr>
<tr>
<td><strong>TitleString</strong></td>
<td><strong>title block symbol</strong></td>
<td>String. Contains the title string from the worksheet.</td>
</tr>
<tr>
<td><strong>TotalSheets</strong></td>
<td><strong>title block symbol</strong></td>
<td>Unsigned integer. Contains the total number of sheets as shown on the worksheet.</td>
</tr>
</tbody>
</table>
Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>TypeCode</th>
<th>standard symbol</th>
<th>Character. Contains the current net node type.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The net node types are:</td>
</tr>
<tr>
<td>L</td>
<td>Label node</td>
<td>L</td>
</tr>
<tr>
<td>S</td>
<td>Power node</td>
<td>S</td>
</tr>
<tr>
<td>P</td>
<td>Port node</td>
<td>P</td>
</tr>
<tr>
<td>N</td>
<td>Node containing no symbolic name</td>
<td>N</td>
</tr>
<tr>
<td>U</td>
<td>Unconnected node</td>
<td>U</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>access_constant</th>
<th>type definition for general functions</th>
<th>A subset of string_constant, this must be one of these four access fields:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&quot;PARTVALUE&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LIBRARY&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LOOKUP&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;REFERENCE&quot;</td>
</tr>
</tbody>
</table>

See SetAccessType() in the section Function reference.

<table>
<thead>
<tr>
<th>file_index</th>
<th>type definition for general functions</th>
<th>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.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>format_constant</th>
<th>type definition</th>
<th>One of these three values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ADF A standard known as both the Altera Design Format and Intel's Advanced Design File.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOD OrCAD's Programmable Logic Device Modeling Tools.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EDIF Electronic Design Intercange Format</td>
</tr>
</tbody>
</table>

See ExceptionsFor() in the section Function reference.
<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer_constant</td>
<td>A whole number in the range -32,678 to 32,767.</td>
<td>A whole number in the range -32,678 to 32,767.</td>
</tr>
<tr>
<td>part_symbol</td>
<td>A subset of user_symbol for use with functions for part-oriented netlists.</td>
<td>A subset of user_symbol for use with functions for part-oriented netlists.</td>
</tr>
<tr>
<td>SIGNALS</td>
<td>The table of all signal names and types.</td>
<td>The table of all signal names and types. The net contains one signal name and type for each connected node.</td>
</tr>
<tr>
<td>PARTS</td>
<td>The table of all parts found in the instance file.</td>
<td>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.</td>
</tr>
<tr>
<td>NETS</td>
<td>The table of all nets.</td>
<td>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.</td>
</tr>
<tr>
<td>NODES</td>
<td>The list of all nodes on a given net.</td>
<td>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.</td>
</tr>
<tr>
<td>ALL</td>
<td>The list of all signals, parts, nets, and nodes.</td>
<td>The list of all signals, parts, nets, and nodes.</td>
</tr>
</tbody>
</table>

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.
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th><strong>standard_symbol</strong></th>
<th><strong>type</strong></th>
<th>A string enclosed in quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>Either a standard_symbol or a user_symbol.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>string_constant</strong></th>
<th><strong>type</strong></th>
<th>A string enclosed in quotation marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>Either a standard_symbol or a user_symbol.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>symbol</strong></th>
<th><strong>type</strong></th>
<th>Either a standard_symbol or a user_symbol.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>Either a standard_symbol or a user_symbol.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>traversal_constant</strong></th>
<th><strong>type</strong></th>
<th>A subset of string_constant, this must be either &quot;ROOT&quot; or &quot;LEAF&quot;. See SetTraversal() in the section Function reference.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>A subset of string_constant, this must be either &quot;ROOT&quot; or &quot;LEAF&quot;. See SetTraversal() in the section Function reference.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td>A subset of string_constant, this must be either &quot;ROOT&quot; or &quot;LEAF&quot;. See SetTraversal() in the section Function reference.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>user_symbol</strong></th>
<th><strong>type</strong></th>
<th>A user-defined symbol.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>A user-defined symbol.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td>A user-defined symbol.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>variable</strong></th>
<th><strong>type</strong></th>
<th>An integer defined with an int statement in the range -32,768 to 32,767.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>definition</strong></td>
<td>An integer defined with an int statement in the range -32,768 to 32,767.</td>
</tr>
<tr>
<td></td>
<td><strong>for general functions</strong></td>
<td>An integer defined with an int statement in the range -32,768 to 32,767.</td>
</tr>
</tbody>
</table>

#### 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.

```c
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.
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int AccessKeyWord</td>
<td>Functions</td>
</tr>
<tr>
<td>(string_constant;)</td>
<td>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.</td>
</tr>
<tr>
<td>int AccessKeyWord</td>
<td>Functions</td>
</tr>
<tr>
<td>(standard_symbol;)</td>
<td>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.</td>
</tr>
<tr>
<td>int AccessKeyWord</td>
<td>Functions</td>
</tr>
<tr>
<td>(user_symbol;)</td>
<td>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.</td>
</tr>
<tr>
<td>AddSymbol</td>
<td>Functions</td>
</tr>
<tr>
<td>(string_constant;)</td>
<td>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.</td>
</tr>
<tr>
<td>int ChildPinCount()</td>
<td>Functions</td>
</tr>
<tr>
<td></td>
<td>Returns the total number of sheet pins on the current child.</td>
</tr>
<tr>
<td>ClearSymbolic Strings</td>
<td>Functions</td>
</tr>
<tr>
<td></td>
<td>Clears the list of available user symbols. All user symbols and their contents are erased. See also AddSymbol() in the section Function reference.</td>
</tr>
</tbody>
</table>
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>int CompareSymbol</strong> (symbol, symbol);</td>
<td>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.</td>
</tr>
<tr>
<td><strong>ConcatFile</strong> (file_index_1, file_index_2);</td>
<td>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.</td>
</tr>
<tr>
<td><strong>CopySymbol</strong> (symbol, user_symbol);</td>
<td>Copies the contents of the symbolic string indicated by symbol into the indicated user_symbol.</td>
</tr>
<tr>
<td><strong>CreatePartDataBase();</strong></td>
<td>Uses the instance file to create a part-oriented database IFORM can use. You call this function within the required function WriteHeader().</td>
</tr>
<tr>
<td><strong>int EndNode();</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>ExceptionsFor</strong> (format_constant, user_symbol);</td>
<td>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(&quot;EDIF&quot;, TempStr )</td>
</tr>
</tbody>
</table>
### int FindSymbolChar

- **Function**: General

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 `0xFFFFH (-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()

- **Function**: Child Data

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()

- **Function**: Child Data

Makes the first sheet net current and loads the associated standard symbols.

### FirstNet()

- **Function**: Net-oriented Data

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()

- **Function**: Net-oriented Data

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.
<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int FirstPart();</code></td>
<td><em>part data structure function</em></td>
<td>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, <code>FirstPart()</code> and <code>FirstChildO</code> return 0. Use this function only when creating format files for hierarchical netlists.</td>
</tr>
<tr>
<td><code>FirstPin();</code></td>
<td><em>part data structure function</em></td>
<td>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.</td>
</tr>
<tr>
<td><code>int FirstPipe();</code></td>
<td><em>pipe file function</em></td>
<td>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.</td>
</tr>
<tr>
<td><code>int getch();</code></td>
<td><em>C-language function</em></td>
<td>Returns a character from the keyboard.</td>
</tr>
<tr>
<td><code>int GetIndex( part_symbol );</code></td>
<td><em>part-oriented netlist function</em></td>
<td>Returns the current index for the table indicated by part_symbol. For example: <code>GetIndex(SIGNALS);</code>&lt;br&gt;For a discussion of the tables, see the type definition part_symbol.</td>
</tr>
<tr>
<td><code>int getnum();</code></td>
<td><em>C-language function</em></td>
<td>Returns an integer that is input from the keyboard. This function does not check the input to make sure it is valid.</td>
</tr>
</tbody>
</table>
int GetStandardSymbol(standard_symbol);

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 );

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();

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:

<table>
<thead>
<tr>
<th>Net type</th>
<th>Symbols accessed by HandleNodeName()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label (L)</td>
<td>TypeCode, NetNumber, SignalType, SignalNameString</td>
</tr>
<tr>
<td>Port (P)</td>
<td>TypeCode, NetNumber, SignalType, SignalNameString</td>
</tr>
<tr>
<td>Power (S)</td>
<td>TypeCode, NetNumber, SignalType, SignalNameString</td>
</tr>
<tr>
<td>Node (N)</td>
<td>TypeCode, NetNumber</td>
</tr>
<tr>
<td>Unconnected (U)</td>
<td>TypeCode, NetNumber</td>
</tr>
</tbody>
</table>

Initialize();

IFORM and HFORM call this function first. Here you can initialize symbols and variables, define character sets, write headers, and set number widths.
Appendix D: Creating custom netlist formats

```plaintext
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 *.CH 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*. 
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int NextKeyWord()</td>
<td>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.</td>
</tr>
<tr>
<td>int NextNet();</td>
<td>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.</td>
</tr>
<tr>
<td>int NextNode();</td>
<td>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.</td>
</tr>
<tr>
<td>int NextPart();</td>
<td>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.</td>
</tr>
<tr>
<td>int NextPin();</td>
<td>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.</td>
</tr>
<tr>
<td>int NextPipe();</td>
<td>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().</td>
</tr>
</tbody>
</table>
int PackString
  ( integer_constant,
    integer_constant,
    user_symbol,
    user_symbol );

int PackString
  ( variable, variable,
    user_symbol, 
    user_symbol );

def deleted 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 );

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();

Returns the number of pins on the current part.
Use this function only when creating format files for hierarchical netlists.

PostFile();

HFORM calls this function after parsing the current worksheet. All the data structures are loaded and available for use.

PostProcess();

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.
Appendix D: Creating custom netlist formats

**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);**  
C-language functions  
Writes the contents of string_constant, standard_symbol, user_symbol, or variable to the screen. The function returns 0.

**int print(standard_symbol);**

**int print(user_symbol);**

**int print(variable);**

**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);

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.

int puts (user_symbol);

int puts (standard_symbol)

PutSymbolChar (variable, integer_constant, 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.
Appendix D: Creating custom netlist formats

**SetAccessType**

*(access_constant)*;  
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:

<table>
<thead>
<tr>
<th>Value of access_constant or user_symbol</th>
<th>Symbol accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTVALUE</td>
<td>PartName</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>LibraryNameString</td>
</tr>
<tr>
<td>LOOKUP</td>
<td>LookupNameString</td>
</tr>
<tr>
<td>REFERENCE</td>
<td>ReferenceString</td>
</tr>
</tbody>
</table>

**SetCharSet**

*(string_constant)*;  
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)*;  
Makes the first entry in the table indicated by part_symbol current. For example:

SetFirst(SIGNALS);

**int SetIndexByRef**

*(user_symbol)*;  
Accesses the PARTS table, making the part with the reference specified by user_symbol current. This function returns 1 on success or 0 otherwise.
### Schematic Design Tools Reference Guide

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int SetNext (part_symbol);</td>
<td>Makes the next index in the table indicated by <code>part_symbol</code> current. This function returns 1 on success or 0 otherwise. This function appears in loop constructs. For example: SetNext (ALL);</td>
</tr>
<tr>
<td>SetNumberWidth (integer_constant);</td>
<td>Sets the width of fields containing output number to the value specified by <code>integer_constant</code> or <code>variable</code>. Numbers are padded with zeros on the left up to this width when <code>IFORM</code> or <code>HFORM</code> write them to a file or to the screen. Except for the symbol <code>LocalSignal</code>, 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.</td>
</tr>
<tr>
<td>SetNumberWidth (variable);</td>
<td></td>
</tr>
<tr>
<td>int SetPartIndex (integer_constant);</td>
<td>Makes the part specified by <code>integer_constant</code> or <code>variable</code> 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.</td>
</tr>
<tr>
<td>int SetPartIndex (variable);</td>
<td></td>
</tr>
<tr>
<td>SetPinMap (integer_constant, string_constant);</td>
<td>Sets the index for the input pin map specified by <code>integer_constant</code> or <code>variable</code> to the input <code>string_constant</code>. 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 <code>WriteMap()</code>;</td>
</tr>
<tr>
<td>SetPinMap (variable, string_constant);</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>int SetPrevious</td>
<td>Makes the previous index in the table indicated by <code>part_symbol</code> current. This function returns 1 on success or 0 otherwise. Loop constructs may include this function. For example: <code>SetPrevious(PARTS);</code> Use this function only when creating format files for hierarchical netlists.</td>
</tr>
<tr>
<td>int SetSignal()</td>
<td>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.</td>
</tr>
<tr>
<td>SetSymbol</td>
<td>Assigns the contents of <code>strins-constant</code> to <code>user_symbol</code>.</td>
</tr>
<tr>
<td>int SetToIndex</td>
<td>Sets the index of the table specified by <code>part_symbol</code> to the input variable. Returns 1 on success or 0 otherwise. For example: <code>SetToIndex(NETS, 5);</code></td>
</tr>
<tr>
<td>SetTraversal</td>
<td>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 <code>traversal_constant</code> is either &quot;ROOT&quot; or &quot;LEAF&quot;, and <code>user_symbol</code> should evaluate to either ROOT or LEAF. For example: <code>SetTraversal(&quot;ROOT&quot;);</code></td>
</tr>
<tr>
<td>Function Name</td>
<td>Category</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>SortByNumber();</td>
<td>instance</td>
</tr>
<tr>
<td>StripPath (string_symbol);</td>
<td>general</td>
</tr>
<tr>
<td>int SwitchIsSet (string_constant);</td>
<td>general</td>
</tr>
<tr>
<td>int SymbolInCharSet (user_symbol);</td>
<td>general</td>
</tr>
<tr>
<td>int SymbolLength (symbol);</td>
<td>general</td>
</tr>
<tr>
<td>ToUpper (user_symbol);</td>
<td>general</td>
</tr>
<tr>
<td>WriteCrLf (file_index);</td>
<td>general</td>
</tr>
</tbody>
</table>
### Appendix D: Creating custom netlist formats

<table>
<thead>
<tr>
<th>Function</th>
<th>Required Function for IFORM Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteHeader();</td>
<td>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().</td>
</tr>
<tr>
<td>WriteInteger(file_index, integer_constant);</td>
<td>Writes the specified integer_constant or variable to the file given in file_index.</td>
</tr>
<tr>
<td>WriteInteger(file_index, variable);</td>
<td></td>
</tr>
<tr>
<td>WriteMap(file_index, integer_constant);</td>
<td>Writes the pin map string at integer_constant or variable to the file given by file_index. See also the SetPinMap() functions.</td>
</tr>
<tr>
<td>WriteMap(file_index, variable);</td>
<td></td>
</tr>
<tr>
<td>WriteNet();</td>
<td>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().</td>
</tr>
<tr>
<td>WriteNetEnding();</td>
<td>IFORM calls this function when it reaches the end of a net. Then this function outputs whatever you specified to terminate nets.</td>
</tr>
<tr>
<td>WriteNetListEnd();</td>
<td>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().</td>
</tr>
<tr>
<td>Function Name</td>
<td>General Function</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>WriteStdSymbol</td>
<td>function</td>
</tr>
<tr>
<td>WriteString</td>
<td>function</td>
</tr>
<tr>
<td>WriteSymbol</td>
<td>function</td>
</tr>
</tbody>
</table>
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 ESP, 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.
This appendix contains additional information you may need to setup, configure, and use your plotter with Schematic Design Tools. The sections in this appendix are:

- Plotter cable wiring
- Plotter problems
- Plotting to a printer
- General plotter tips
- HP plotters
- HI plotters
- Calcomp plotters
- Notes on plotter and printer drivers
- Postscript plotter 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 PC's 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
Appendix E: Plotter information

Figure E-2. 9-pin cable wiring diagram

Figure E-3. 25-pin cable to IOLine plotter
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 follows:

    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, see your DOS user's guide for Asynchronous Communications.

After the serial channel has been configured, send the plot file to the plotter using the DOS COPY command as follows:

    COPY whatfile COM1:

Where *whatfile* is the name of the plot file.

If the plotter works, this indicates the problem may be in the plotter cable (incorrectly wired), or the hardware handshaking is incorrectly set (check the Printer/Plotter Output Options section of the Configure Schematic Design Tools screen).

If the plotter does not work, this indicates that there is a hardware problem. Check the following: serial card, incorrect serial channel configuration, plotter hardware, or a cable problem.

7. If you're using an IOline plotter, be sure you have PROM version 114 or greater.
When plotting to a printer, make the pen width for buses small enough to make the lines thin. This will make the print neater and more readable. For example, if you have a printer with a resolution of 180 dots per inch (dpi) you would set the pen width to be:

\[
\frac{1}{180} \times \frac{1}{2} = 0.00277 \text{ inches}
\]

for the best results. The same value is also used for the part body pen width. This pen width value is used during FILL commands in the vector stream of the part definition.
**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 affects 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 pen(s) 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.
The HP plotter family has a facility to set the corner points of the plot and automatically scale the plot to be within these points. These points are called P1 and P2. Plot Schematic ignores the preset P1 and P2 values and draws in 0.001 inch resolution.

Plot Schematic assumes the origin of the plot is the lower left corner of the page (when the finished plot is viewed). Rotation and paper size must be set before you run a plot.

If the plotter’s origin (0,0) is not the lower left corner, it may be moved via the Template Table section of the Configure Schematic Design Tools screen. To move the origin, configure the Plot X Offset and Plot Y Offset (value -32.76 inches to +32.76 inches) in the Template Table. Note that this origin offset can be used on any plotter to adjust where the plot is made on the page.

For large HP plotters (HP 7580, 7585, 7586), the origin is the center of the page and the template offset values must be set. For these HP plotters, the offset values are negative. The offset values are added to the location values and then sent to the plotter.

For example, suppose you set the Plot X Offset to -10 and the Plot Y Offset to -5. If your plot contains the logical point X=1 and Y=2, that point is actually sent to the plotter as X=-9 and Y=-3 (-9 = 1 + -10; -3 = 2 + -5).
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.

<table>
<thead>
<tr>
<th>PC AT</th>
<th>PC XT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>Plotter</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>22</td>
</tr>
</tbody>
</table>

Table E-1. Pinouts for Calcomp plotter cables.

Table E-2 lists intelligent and non-intelligent Calcomp plotters supported by Schematic Design Tools and which plotter driver to use for each supported plotter. Following the table is a key to codes used in the table.
## Table E-2. Supported Calcomp plotters and corresponding OrCAD plotter drivers.

| Supported Plotters | Controllers | No Controller | | | | | |
|--------------------|-------------|---------------|---|---|---|---|
|                    | 906         | 907           | 907 Rev G | 951/953 | PCI | 960 Format | Ext 960 Format |
| **Non-Intelligent** |             |               |          |          |     |            |                |
| 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.
2. Use CALCOMP2.DRV for Schematic Design Tools.
3. Plotter and configuration supported by Calcomp Company. No driver for Schematic Design Tools.
4. The 906 controller and 960 format lack commands to draw circles, arcs, and dashed lines.
5. The extended 960 format is not supported.
6. The remaining unsupported configurations result from differences in plotter step size.
7. OrCAD's CALCOMP plotter drivers support only a step size of 2032.
<table>
<thead>
<tr>
<th>NO.</th>
<th>Selection</th>
<th>AUTOCAD</th>
<th>OrCAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Bits</td>
<td>2 - Even</td>
<td>0 - None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>Stop bits</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Clock</td>
<td>0 - INT</td>
<td>0 - INT</td>
</tr>
<tr>
<td>2</td>
<td>Interface</td>
<td>0 - Serial</td>
<td>0 - Serial</td>
</tr>
<tr>
<td>3</td>
<td>Host baud rate</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>PCI</td>
<td>PCI</td>
</tr>
<tr>
<td>4</td>
<td>Term muting</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Checksum enable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Isochronous</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>EOM</td>
<td>13</td>
<td>03</td>
</tr>
<tr>
<td>6</td>
<td>Direct control</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>XON/XOFF</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Term baud rate</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td></td>
<td>Duplex</td>
<td>0 - Full</td>
<td>0 - Full</td>
</tr>
<tr>
<td>8</td>
<td>Sync codes</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Sync code value</td>
<td>022</td>
<td>002</td>
</tr>
<tr>
<td>16</td>
<td>Enable optimization</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*Table E-3. Calcomp communications and plot management settings.*

<table>
<thead>
<tr>
<th>0—&gt; 1</th>
<th>AUTOCAD</th>
<th>OrCAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS - 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS - 2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>POS - 3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS - 4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>POS - 5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>POS - 6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS - 7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POS - 8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*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 did not set the Plot X Offset and Plot Y Offset on the Configure Schematic Design Tools screen. This is because the values that the HP plotter uses are plotter units, not inches. A plotter unit is 0.00098 inch. The plotter units must be a 16-bit integer value (-32768 to +32767). Plot Schematic computes all dimensions with much greater range.

The plotter driver is passed word values (0.000 to +65.535) and these are converted into plotter units after adding the offset. 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 be able to plot D and E-size drawings, the full integer range must be used. 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 \(-\frac{1}{2}\) 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 to the lower left corner of the paper.
Some laser printers have a graphics printing limit. The Schematic Design Tools drivers place page breaks after a given number of graphics lines. To change the number of graphics lines between page breaks, change the word value in the driver at the offset specified in table E-5. The driver values were chosen to be a 10 inch graphics printing area.

<table>
<thead>
<tr>
<th>Driver</th>
<th>Offset</th>
<th>Old value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPLASER1.DRV</td>
<td>401h</td>
<td>02EEh (750)</td>
</tr>
<tr>
<td>HPLASER2.DRV</td>
<td>41Ch</td>
<td>03EBh (1000)</td>
</tr>
<tr>
<td>HPLASER3.DRV</td>
<td>44Eh</td>
<td>05DCh (1500)</td>
</tr>
<tr>
<td>HPLASER4.DRV</td>
<td>4E4h</td>
<td>0BB8h (3000)</td>
</tr>
</tbody>
</table>

Table E-5. Offsets for page breaks using laser printers.

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: Dashed lines will always be layer 0 due to line definition restrictions.
Appendix E: Plotter information

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 application programs such as word processing and page layout programs, 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.

Like 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.
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. The last section, File extensions by tool set gives a table showing which file extension each tool set uses.

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 any of three tools: Edit File, INET in Create Netlist, and INET in Create Hierarchical Netlist. 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.

.AST  Stimulus file created by any of three tools: Edit File, INET in Create Netlist, and INET in Create Hierarchical Netlist. 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.


.ATR  Trace file created by any of three tools: Edit File, INET in Create Netlist, and INET in Create Hierarchical Netlist. 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.


.BAK  Backup of a schematic file created by Cleanup Schematic 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.

### Files and file extensions

<table>
<thead>
<tr>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.CCF</td>
<td>Compiled version of the netlist format source file (.CF). IFORM extracts the netlist format guidelines from this binary file when creating a flat netlist. Compiled netlist format files are typically found in the <code>\ORCADESP\SDT\NETFORMS</code> directory.</td>
</tr>
<tr>
<td>.CCH</td>
<td>Compiled version of the hierarchical netlist format source file (.CH). HFORM extracts the netlist format guidelines from this binary file when creating a hierarchical netlist. Compiled netlist format files are typically found in the <code>\ORCADESP\SDT\NETFORMS</code> directory.</td>
</tr>
<tr>
<td>.CF</td>
<td>Flat netlist format source file that OrCAD provides or you create with a text editor. This text file contains formatting information for a netlist in one of over thirty formats. Although IFORM can use the information in this file when it creates a netlist, the process takes less time if you configure IFORM to use the corresponding .CCF file. Netlist format source files are typically found in the <code>\ORCADESP\SDT\NETFORMS\SOURCE</code> directory.</td>
</tr>
<tr>
<td>.CH</td>
<td>Hierarchical netlist format source file that OrCAD provides or you create with a text editor. This text file contains formatting information for a hierarchical netlist in one of several formats. Although HFORM can use the information in this file when it creates a hierarchical netlist, the process takes less time if you configure HFORM to use the corresponding .CCH file. Netlist format source files are typically found in the <code>\ORCADESP\SDT\NETFORMS\SOURCE</code> directory.</td>
</tr>
<tr>
<td>.ERC</td>
<td>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.</td>
</tr>
</tbody>
</table>
**.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.


**.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.

**.LNF** Linked connectivity database file created by ILINK in Create Netlist. ILINK creates a text file containing information about all the devices, connections, pipe commands, and title blocks in a design. PC Board Layout Tools uses .LNF files.
### Appendix F: Files and file extensions

<table>
<thead>
<tr>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.MAC</td>
<td>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.</td>
</tr>
<tr>
<td>.MAP</td>
<td>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.</td>
</tr>
<tr>
<td>.NET</td>
<td>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.</td>
</tr>
<tr>
<td>.PIP</td>
<td>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.</td>
</tr>
<tr>
<td>.PLT</td>
<td>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.</td>
</tr>
<tr>
<td>.PRN</td>
<td>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.</td>
</tr>
<tr>
<td>.RES</td>
<td>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.</td>
</tr>
<tr>
<td>.SCH</td>
<td>Schematic worksheet file created by Draft in binary format.</td>
</tr>
</tbody>
</table>
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.

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.

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.

Tree list created by Show Design Structure. This text file lists the sheets in a design and shows the relationships between the sheets.

Cross-reference file created by Cross Reference Parts. This text file tells you where each part is located on a worksheet.
## Other files

The sections lists files that don’t follow the naming convention of design name and default extension.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP_OUT.TXT</td>
<td>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.</td>
</tr>
<tr>
<td>HARDCOPY.PRN</td>
<td>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.</td>
</tr>
<tr>
<td>ORCADESCP.DAT</td>
<td>Data file that stores configuration information for each design. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.</td>
</tr>
<tr>
<td></td>
<td>See Chapter 5: Design environment technical information in the OrCAD/ESP Design Environment User’s Guide for more information about data files.</td>
</tr>
<tr>
<td>SDT.BCF</td>
<td>Binary configuration file for Schematic Design Tools. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.</td>
</tr>
<tr>
<td></td>
<td>See Chapter 5: Design environment technical information in the OrCAD/ESP Design Environment User’s Guide for more information about configuration files.</td>
</tr>
<tr>
<td>SDT.CFG</td>
<td>Configuration file for Schematic Design Tools. This file is located in the \ORCAD\TEMPLATE directory, and in each design directory.</td>
</tr>
<tr>
<td></td>
<td>See Chapter 5: Design environment technical information in the OrCAD/ESP Design Environment User’s Guide for more information about configuration files.</td>
</tr>
</tbody>
</table>

## 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 \ORCADESCP\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 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:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'74LS00'</td>
<td>'14DIP300'</td>
</tr>
<tr>
<td>'74LS138'</td>
<td>'16DIP300'</td>
</tr>
<tr>
<td>'74LS163'</td>
<td>'16DIP300'</td>
</tr>
<tr>
<td>'74LS373'</td>
<td>'20DIP300'</td>
</tr>
<tr>
<td>'74LS245'</td>
<td>'20DIP300'</td>
</tr>
<tr>
<td>'8259A'</td>
<td>'24DIP600'</td>
</tr>
</tbody>
</table>

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 13: Update Field Contents for more information about update files.
Appendix F: Files and file extensions

**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 7: Back Annotate for more information about Was/Is files.

**File extensions by tool set**

Table F-1 lists the file extensions used or created in each OrCAD tool set.
<table>
<thead>
<tr>
<th>File extension</th>
<th>Schematic Design Tools</th>
<th>Programmable Logic Design Tools</th>
<th>Digital Simulation Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ABR</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.AST</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.ATR</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.BAK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>.BCF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.BOM</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.BRK</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.CCF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>.CCH</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.CF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.CFG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>.CH</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.DBA</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.DEF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.DSF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.ERC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>.ERR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>.HEX</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.IGS</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.INF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.INS</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.INX</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.JED</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.LIB</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.LNF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.LOG</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.LST</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.MAC</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.MAP</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.NET</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.PIP</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.PLA</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.PLD</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.PLT</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.PRN</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.RES</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.SCH</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.SRC</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.STM</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TDF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TRC</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TVD</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TVO</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TVS</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.TWG</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.VEC</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>.XRF</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table F-1. File extensions in each tool set.
analogue 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.

CAE ■ An acronym for computer aided engineering.

check box ■ A small square button: O. 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 sheet path 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.

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*.

**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:

**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*.

**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.

**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 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 ■ A unit of measurement. 1K byte is equal to 1024 bytes. The "K" is taken from the metric system, where it stands for "kilo," or 1000. 1024 is \(2^{10}\) and is close to 1000.

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.

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.

linked connectivity database ■ ILINK can optionally be configured to create the linked connectivity database. This ASCII file has an extension of .LNF and is used to transfer to PC Board Layout Tools.

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.

megabyte ■ Slightly more than one million bytes; 10 megabytes equals 10 million bytes. A megabyte is equal to \(2^{20}\) bytes (1,048,576). "Mega" is taken from the metric system, where it is a prefix meaning one million.

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 picture element. 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.
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.

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.

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.

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.

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.
Special Characters

3-state pins see also pins defined 311
<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 201

A
ABR files 661, 670
AccessChild function in netlist format files 625
AccessKeyWord function in netlist format files 626
AccessPart function in netlist format files 626
access_constant type definition in netlist format files 623
active libraries see libraries
Active library size 18
AddressLine1-AddressLine4 symbol in netlist format files 617
AddSignalName function in netlist format files 626
AddSymbol function in netlist format files 626
AGAIN command
Draft 62
Edit Library 271
Algorex netlist format 487
Allegro netlist format 489
alphanumeric pin numbers 246
AlteraADF netlist format 490
including equations in the netlist 491
analog, defined 671

ANNOTATE see also Annotate Schematic, 435
command line controls 462
Annotate Schematic 183-188
before and after annotation 185
execution 183
Key Fields 40-42, 184
Local Configuration 186
File Options 186
Processing Options 187
part designation 38
preparing for simulation 187
processing complex hierarchies 187
reference designators 40-41
updating all reference designators 188
updating unannotated reference designators 188
annotation, defined 671
ANSI
grid references 20
title block 19
AppliconBRAVO netlist format 494
AppliconLEAP netlist format 495
Archive Parts in Schematic 240
COMPOSER 254
creating a library source file 256
execution 253
LIBARCH 254
Local Configuration 254
Configure LIBARCH 255
File Options 255
Processing Options 257
sheetpath parts 257
string files 257
string files as destination 256
ASCII, defined 671
AST files 662, 670
ATR files 662, 670
Available Display Drivers 8
Available Libraries 14
Available Plotter Drivers 10
Available Printer Drivers 9

679
B

Back Annotate 189-191
  execution 189
  Local Configuration 190
  File Options 190
  Processing Options 191
  selectively updates reference
designators 183
Was/Is files 189
back up files in Draft 149
BACKANNO see also Back Annotate
  command line controls 463
BAK files 193, 662, 670
batch files, running from Draft 146
baud rate 11
BCF files 670
bidirectional pins see also pins
defined 311
bill of materials see also Create Bill of
  Materials
  include file format 426
  including information from a text file
  425
BIOS and printing and plotting 11
bitmap definition see also Edit Library, 333
  creating 345-346
  graphic symbols 345
  maximum number of bits 348
  screen representation 275
bitmap images, Edit Library CONDITIONS
  292
bitmap, defined 260, 671
BLOCK command in Draft 63-71
  ASCII Import 70
  Drag 65
  Export 69
  Fixup 65
  Get 66
  Import 68
  Move 64
  Save 67
  Text Export 71
block parts see also parts
  introduction 244
  restrictions 272
block symbol definitions 334-344
  comments in 335
BODY command in Edit Library 272-290
  <Block> 276
    Kind of Part 276
    Size of Body 276
  <Graphic> 277
    Arc 279
    Circle 278
    Circle Center 278, 279
    Delete 283
    Erase Body 284
    Fill 283
    IEEE Symbol 281
    Kind of Part 284
    Line 277
    Size of Body 284
    Text 280
  <IEEE> 285
    Circle 286
    Delete 289
    Erase Body 289
    IEEE Symbol 288
    Kind of Part 290
    Line 285
    Size of Body 289
    Text 287
Does Graphic Part have CONVERT? 274
Is Part a GRID ARRAY? 274
  Kind of Part? 273
  Number of Parts per Package 274
  Place 275
BOM files 662, 670
  Border Text, character height 34
  border, worksheet see worksheets
Index

boxes
  command line xxxii
  entry xxxii
  menu xxxii
  prompt xxxii
brackets xxxii
branch parameter stimulus statements in .INF and .LNF files 592
BRK files 662, 670
buffers see also CONDITIONS command
  Draft
    getting and placing objects 66
    hierarchy 25, 72, 73
    macro 23, 72, 73
    saving objects 67
Edit Library
  examining buffer conditions 291
  Macro Buffer 292
bus stimulus statements in .INF and .LNF files 594
bus trace statements in .inf and .Inf files 586, 588
bus vector statements in .INF and .LNF files 591
buses
  checking for overlapping or duplicate objects 193
  combining labels 162
  connecting to module ports 165
  controlling stretching 150
  dragging 150
  drawing orthogonal 151
  hotpoint 161
  labeling 160, 162
  multiple labels 162
  naming 160
  placing entries in Draft 124
  placing in Draft 122
  short-cut for aligning 65
  splitting 166
buttons
  check box defined 671
  defined 671
  introduction xxv
  mouse xxx
  radio button defined 675
  scroll 5
  scroll button defined 675
  selecting xxv, xxxi
  user defined 676
byte, defined 671
C
  Cadnetix netlist format 496
  CAE, defined 671
  Calay netlist format 498, 500
  Calcomp plotters 654
  CALCOMP1.DRV 655
  CALCOMP2.DRV 655
  Case netlist format 501
  cautions xxxii
  CBDS netlist format 503
  CCF files 480, 663, 670
  CCH files 480, 663, 670
  CF files 480, 663, 670
  CFG files 670
  CH files 480, 663, 670
character height
  Comment Text 34
  Label 34
  Module Text 34
  Part Field 34
  Part Reference 33
  Part Value 34
  Pin Name 33
  Pin Number 33
  Power Text 34
  Sheet Name 34
  Sheet Net 34
character in .INF and .LNF files defined 570
character strings in Symbol Description Language 358, 360
check box, defined 671
Check Electrical Rules 387-395
  configuration matrix 51
  error messages 389-390
  execution 387
  forcing to process all sheets 395
Local Configuration 393
  File Options 393
  Processing Options 394
  matrix explained 389
  option under Create Netlist 212
  restoring default settings 390
  type mismatches 401
child data structure in netlist format files 612
child instance statements in .INF and .LNF files 579
child, defined 671
ChildPinCount function in netlist format files 626
CLEANUP see also Cleanup Schematic
  command line controls 463
Cleanup Schematic 193-196
  checking for errors 193
  execution 193
  Local Configuration 195
    File Options 195
    Processing Options 196
  reporting off-grid parts 196
ClearSymbolicStrings function in netlist format files 626
clock, computer see system clock
Color and Pen Plotter Table 26-29
  Pen speed 28
  Pen width 28
colors, screen 26
command in .INF and .LNF files defined 570
command line controls 461-478
command reference
  Draft 59-158
  Edit Library 321
Comment Text, character height 34
CompareSymbol function in netlist format files 627
Compile Library 240, 381-384
  creating custom libraries 242, 243
  creating custom libraries with 381
  execution 381
  Local Configuration 382
    File Options 382
    Processing Options 384
Compile Simulation Specification File 451
  configuring 451
compiled library file see also libraries
  .LIB extension 240
  defined 240
compiler, netlist see INET
complex hierarchies see also hierarchical
  Annotate Schematic 187
  converting to flat hierarchies 199
  creating netlists 199
  defined 671
  prerequisite for creating netlists 209
Complex to Simple
  flattening complex hierarchies 199
  prerequisite for Create Netlist 209
COMPOSER
  command line controls 464
ComputerVision netlist format 504
ConcatFile function in netlist format files 627
CONDITIONS command
  Draft 72
  Edit Library 291
configuration, defined 671
Configure Annotate Schematic 186-188
Configure ASCTOVST 450
Configure Back Annotate 190-191
Configure Check Electrical Rules 393-395
Configure Cleanup Schematic 195-196
Configure Compile Library 382-384
Configure Convert Plot to IGES 405
Configure Create Bill of Materials 424-428
Configure Cross Reference Parts 398-402
Configure Decompile Library 324
Configure Draft 56-58
Configure Edit Library 265-267
Configure Hierarchical Netlist Format 223
Configure Incremental Netlist 211, 223
  File Options 211
  Processing Options 212
Configure Library Archive 255-257
Configure List Library 250-252
Configure Netlist Format 217-219
Configure Netlist Linker 216
Configure Plot Schematic 410-416
Configure Print Schematic 419, 420
Configure Schematic Design Tools 52
  Check Electrical Rules matrix 51
  Color and Pen Plotter Table 26-29
  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 25
  buffer size 25
Key Fields 37-50
  Annotate Part Value Combine 40
  Annotate Schematic 38
  Create Bill of Materials 38
    Include File Combine 47
  Create Netlist 38
  Extract PLD 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

Configure Schematic Design Tools (continued)
  Library Options 12-18
    Active library size 18
    Available Libraries 14
    Configured Libraries 12, 14-15, 16
    inserting a library 14
    Library Prefix 13
    Name Table Location 15
    removing a library 14
    Symbolic Data Location 15-17
Macro Options 23-24
  Part Fields 29
  Printer/Plotter Output Options 11
    Template Table 30-36
  Worksheet Options 19-22
    Worksheet Prefix 21
Configure Select Field View 225-227
Configure Show Design Structure 430-431
Configured Libraries 12, 14-15, 16
connectivity database
  .INF format specification 571
  character defined 571
  child instance statements 579
  command defined 571
  creating 199
  defined 671
delimiter defined 570
deexternal statements 576
connectivity database (continued)
  file structure
    branch parameter stimulus statements 592
    bus stimulus statements 594
    bus trace statements 586, 588
    bus vector statements 591
    differences between .INF and .LNF files 600
    module port
      joined statements in .INF and .LNF files 581
    pin statements 582
    pin stimulus statements 593
    pin trace statements 585, 587
    pin vector statements 590
    sample .INF file 598
    set parameter stimulus statements 592
    sheet net statements 585
    sheet net trace statements 586, 588
    signal stimulus statements 593
    signal trace statements 584, 587
    signal vector statements 590
    signal, joined statements 581
  header 573
  instance statements in .INF and .LNF files 576
  joined statements 581
  link statements 574
  linked 202, 203
  module port statements 575
  number defined 571
  overview 567-568
  parameter defined 571
  part instance statements 576
  pipe statements 596
  quoted token defined 570
  signal statements 576
  statement defined 571
  stimulus statements 592
  string defined 570
connectivity database (continued)
  sub-part code defined 571
  title block statements 573
  token defined 570
  trace statements 587
  vector statements 590
  white space defined 570
connectors
  physical 128, 175-176
  placing, in Draft 175-176
conventions, notation xxxii
convert parts 348-351
  Convert command in Draft 88
  defined 274
  defining 274
  DeMorgan equivalent 348
Convert Plot to IGES 403-405
  Local Configuration 405
  sample output 404
coordinates
  block symbol definitions 335
  displaying pointer in Draft 153
  in vectors for graphic parts 260
  JUMP command in Draft 103
  JUMP command in Edit Library 298
  SET command in Draft 148
  Symbol Description Language 376-377
CopySymbol function in netlist format files 627
Create Bill of Materials 421-428
  configuring Key Fields 423
  execution 421
  Include File Combine 47
  include file format 426
  including information from a text file 425
  Local Configuration 424
  File Options 424
  Processing Options 427
Part Value Combine 47
  sample output 422
Index

Create Hierarchical Netlist 159, 221
  execution 222
  HFORM 221
  INET 221
  Local Configuration 223
    Configure HFORM 223
    Configure INET 223
Create Netlist 159, 207-219
  execution 209
  Key Fields 38
  Local Configuration 209-219
    Configure IFORM 217-219
    Configure ILINK 216
    Configure INET 211
  reporting connected objects 212
  reporting off-grid parts 212
  reporting unconnected objects 212
CreatePartDatabase function in netlist format files 627
Cross Reference Parts 397-402
  execution 397
  Local Configuration 398
    File Options 399
    Processing Options 400
    sample output 398
cursor, defined 671
custom drivers, configuring 8
custom libraries 13, see libraries
custom netlist formats see also netlists
  creating 601-644
customer-contributed netlist formats 602

D
dashed lines
  as guide lines in Draft 120
  part bodies in Edit Library 275
  placing in Draft 134
data bits 11
data functions in netlist format files 611-614
  instance files 613
data structures in netlist format files
  child 612
  IFORM and HFORM 611
  net-oriented 612
  part 611
  part-oriented 613
DateString symbol in netlist format files 617
DECOMP see also Decompile Library
  command line controls 464
Decompile Library 240
  creating custom libraries 243
  execution 323
  Local Configuration 324
    File Options 324
    Processing Options 325
DEF files 670
default, defined 671
DELETE command in Draft 74-75
  Block 75
  Object 74
  Undo 75
delimiter in .INF and .LNF files defined 570
design cycle, defined 672
design environment see also Design Management Tools
  introduction xxv-xxviii
Design Management Tools
  Complex to Simple 199
    prerequisite for Create Netlist 209
    defined 672
design structure
  differences in types 201
  flat 201
  hierarchical 201
  pipe link commands 201
  !LINK commands 201
Schematic Design Tools Reference Guide

Designs
- Checking electrical rules see Check Electrical Rules
- Checking for duplicate reference designators 400
- Checking for type mismatches 401
- Checking for unused parts 400
- Guidelines 159
- Hierarchical 143
- Listing of all parts 397
- Reporting coordinates of parts 400
- Reporting design structure 429-431

Digital Simulation Tools xxv
- Annotating schematics for simulation 187
- Creating hierarchical netlists 223
- Creating netlists for 210
- Model netlist format 529
- Netlist from INET 200
- Unlinked connectivity database 203

Digital, defined 672
Display drivers see drivers
DocumentNumber symbol in netlist format files 617
Double-click, defined xxx
Draft 55-158
- Active library 73
- Adding nets to sheet symbols 80
-Backing up worksheets 149
- Changing label size 77
- Changing module port type 78
- Changing objects 82-92
- Changing orientation 79
  - Labels 77
  - Parts 83-89
- Changing part orientations 88
- Changing reference designators 84
- Changing sheet symbol size 82
- Changing size
  - Labels 77
  - Text 76, 77

Draft (continued)
- Changing style
  - Labels 77
  - Module ports 78
  - Parts 83-89
  - Power objects 79
  - Text 76, 77
- Changing worksheets 143, 144
- Command line controls 465
- Commands see Draft commands
- Configuring initial macros 24
- Configuring locally 56-58
- Connecting buses to module ports 165
- Connecting power 167
- Connecting signals without wires or buses 159
- Controlling panning across the screen 149
- Deleting nets from sheet symbols 80
- Displaying convert form 88
- Displaying pointer coordinates 153
- Displaying text 156
- Displaying worksheets with more detail 158

Editing
- Label names 77
- Labels 77
- Layout objects 92
- Module ports 78
- Nets on sheet symbols 81
- Objects 76-92
- Part fields 86
- Part orientations 88
- Part reference designators 83
- Part values 85, 87
- Power objects 79
- Reference designator locations 84
- Reference designators 84
- Sheet symbol filenames 81
- Sheet symbols 80
- Stimulus objects 92
- Text 76, 77
- Title blocks 90-91
Index

Draft (continued)

  trace objects 92, 135
  vector objects 92
  erasing objects 74-75
  execution 55
  exporting objects to a file 69
  exporting text to a file 71
  finding and reporting errors 388
  fixing wires and buses 65
  getting and placing objects from a buffer 66
  hierarchy buffer 72, 73
  importing objects from a file 68
  importing text from a file 70
  isolating power 170-174
  labeling buses 160
  left mouse button 150
  loading macros automatically 24
  loading parts 95-98
  Local Configuration 56
    File Options 56
      Processing Options 58
  locating commands 59
  locating objects 93-94
  macros
    buffer 72, 73, 113
    calling 113
    creating 109-119
    debugging 111
    initial 112
    macro buffer 23
      nesting 111
    pause 111
    syntax 114
    text files 114
      using three-button mouse 117
    valid macro keys 109
  memory 72
  moving objects 64, 65
  moving reference designators 84
  name table 73
  naming nets on sheet symbols 81
  on-line library 73

Draft (continued)

  orthogonal wires and buses 65
  placing
    bus entries 124
    buses 122
    connectors 175-176
    dashed lines 134
    junctions 123
    labels 125
    Layout objects 140
    module ports 127
    multiple objects 147
    no-connects 139
    pipe link commands 201
    power objects 129
    sheet symbols 131
    stimulus objects 137
    text 133
    trace objects 135
    vector objects 136
    wires 120
  power objects 167
    changing name 79
    changing type 79
    creating different 168
    isolating 170
  power supplies, creating different 169
  power, isolating 174
  quitting without saving changes 145
  repeating commands 62
  reporting design structure 429-431
  restoring deletions 75
  rotating parts 88-89
  saving objects in a buffer 67
  saving worksheets 144
  screen colors, setting 26
  setting error bell 150
  setting macro prompts 151
  showing pin numbers 152
  status information, displaying 72
  stretching buses 150
  symbol table 73
  tasks quick reference 61

687
Draft (continued)

- updating reference designators 183
- worksheet memory 72

Draft commands 59, 74, 75-158

- AGAIN 62
- BLOCK 63-71
  - ASCII Import 70
  - Drag 65
  - Export 69
  - Fixup 65
  - Get 66
  - Import 68
  - Move 64
  - Save 67
- Text Export 71
- CONDITIONS 72
- DELETE 74-75
- EDIT 37, 76-92
  - Add-Net 80
  - Delete 80
  - Filename 81
  - Label 77
  - Label Larger 77
  - Label Name 77
  - Label Orientation 77
  - Label Smaller 77
  - Module Port 78
  - Module Port Name 78
  - Module Port Style 78
  - Module Port Type 78
  - Name 81
  - Net 81
  - Orientation 88
  - Part 41, 83
  - Part Fields 86
  - Part Value 85, 87
  - Power 79
  - Power Name 79
  - Power Orientation 79
  - Power Type 79
  - Reference 83
  - Size 82
- FIND 93-94

Draft commands (continued)

- GET 16, 73, 95-98
  - Convert 97, 274
  - Down 98
  - Mirror 98
  - Over 98
  - Place 97
  - Rotate 97
  - Up 98
- HARDCOPY 99-101
  - Destination 100
  - File Mode 101
  - Make Hardcopy 101
  - Width of Paper 101
- INQUIRE 102, 388
- JUMP 103-105
  - Reference 103
  - X-Location 104
  - Y-Location 105
- LIBRARY 106-107
  - Browse 16, 107
  - Directory 106
- MACRO 108-119
  - Capture 109-112
  - Delete 112
  - Initialize 112
  - List 112
  - Read 112
  - Write 113
- PLACE 120-139
  - Bus 122
  - Dashed Line 134
  - Entry (bus) 124
  - Junction 123
  - Label 125
  - Layout 140
  - Module Port 127-128
  - No-Connect 139
  - Power 129-130
  - Sheet 131-132
  - Stimulus 137-138
  - Text 133
  - Trace 135

688
Draft commands (continued)
  Vector 136
  Wire 120
quick reference 60
QUIT 143-146
  Abandon Edits 145
  Enter Sheet 143
  Initialize 144
  Leave Sheet 143
  Run User Commands 146
  Suspend to System 145
  Update File 144
  Write to File 144
REPEAT 147
SET 148-156
  Auto Pan 149
  Backup File 149
  Drag Buses 150
  Error Bell 150
  Grid Parameters 154
  Left Button 150
  Macro Prompts 151
  Orthogonal 151
  Repeat Parameters 155
  Show Pin Numbers 152
  Title Block 152
  Visible Lettering 156
  Worksheet Size 153
  X,Y Display 153
TAG 157
ZOOM 158
  Center 158
  In 158
  Out 158
  Select 158
DRAFTUSR file 146
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 179
  DSF files 670
  DUMP netlist format 505
  DXF.DRV 658

E
  EDA, defined 672
  EDIF hierarchical netlist format 558
  EDIF netlist format 506
  EDIT command in Draft 37, 76-92
    Add-Net 80
    Delete 80
    Filename 81
    Label 77
    Label Larger 77
    Label Name 77
    Label Orientation 77
    Label Smaller 77
    Module Port 78
    Module Port Name 78
    Module Port Style 78
    Module Port Type 78
    Name 81
    Net 81
    Orientation 88
    Part 41, 83
    Part Fields 86
    Part Value 85, 87
    Power 79
    Power Name 79
    Power Orientation 79
    Power Type 79
    Reference 83
    Size 82
Edit File 177
  configuring an editor 177
  execution 177
  M2EDIT 177
Edit Library 259, 320-321
  bit map images 292
  changing the definition of a part 300
  command reference 321
  configuring initial macros 24
  creating custom libraries 242
  creating parts 242
  editing parts 242
  execution 264
  introduction 259
  library objects 292
  loading macros automatically 24
Local Configuration 265
  File Options 265
    Processing Options 267
  macro buffer 23
  part suffix 296
  placing objects off-grid on parts 282
  status information, displaying 291
  system memory available 291
  system memory, free 292
Edit Library commands
  AGAIN 271
  BODY 272, 273, 276-290
    <Block> 276
      Kind of Part 276
      Size of Body 276
    <Graphic> 277
      Arc 279
      Circle 278
      Circle Center 278, 279
      Delete 283
      Erase Body 284
      Fill 283
      IEEE Symbol 281
      Kind of Part 284
      Line 277
      Size of Body 284
      Text 280
Edit Library commands (continued)
  <IEEE> 285
    Circle 286
    Delete 289
    Erase Body 289
    IEEE Symbol 288
    Kind of Part 290
    Line 285
    Size of Body 289
    Text 287
  Place 275
  CONDITIONS 291
  EXPORT 294
  GET PART 262, 296
  IMPORT 297
  JUMP 298-299
    X-Location 298
    Y-Location 299
  LIBRARY 300
    Browse 302
    Delete Part 303
    List Directory 301
    Prefix 304
    Update Current 262, 300
  MACRO 306
    Initial Macro 306
  NAME 307-308
    Add 308
    Delete 308
    Edit 308
    Prefix 308
  ORIGIN 309
  PIN 310-312
    Add 310
    Delete 310
    Move 312
    Name 310
    Pin-Number 310
    Shape 312
    Type 311
  quick reference 269
Index

Edit Library commands (continued)
  QUIT 313-315
    Abandon Edits 315
    Initialize 314
    Suspend to System 314
    Update File 262, 313
    Write to File 262, 314
  REFERENCE 316
    repeating 271
    selecting 268
  SET 317-319
    Auto Pan 317
    Backup File 317
    Error Bell 318
    Left Button 318
    Macro Prompts 318
    Power Pins Visible 319
    Show Body 275
    Show Body Outline 287, 319
    Visible Grid Dots 319
    tasks 270
  ZOOM 321
    Center 321
    In 321
    Out 321
    Select 321
  editors 53-179
    defined 672
    introduction xxvi
  EEDesigner netlist format 510
  electrical rules see Check Electrical Rules
  EMS see system EMS
    defined 672
  Encapsulated PostScript drivers 659
  EndNode function in netlist format files 627
  enter, defined xxxi
  entry box, defined 672
  EPS1.DRV 659
  EPS2.DRV 659
  EPS3.DRV 659
  EPS4.DRV 659

  equations included in the netlist
    AlteraADF netlist format 491
    IntelADF netlist format 519
  ERC see also Check Electrical Rules
  ERC files 663, 670
  ERR files 670
  errors see also Check Electrical Rules
    Already at the Root Level 143
    checking for on schematics 193, 389-390
    correcting errors in Draft 102
    in multi-sheet designs 205
    messages reported by IFOR and
      HFORM 643-644
    preventing errors when placing parts 153
    reported by HP.DRV 657
    setting error bell in Draft 150
    setting error bell in Edit Library 318
    Tag does not exist 103, 298
    update files 232
  ESP xxv-xxviii
  ExceptionsFor function in netlist format
    files 627
  ExitType symbol in netlist format files 617
  EXPORT command in Edit Library 294
  extensions
    .ABR 670
    .AST 670
    .ATR 670
    .BAK 193, 670
    .BCF 670
    .BOM 670
    .BRK 670
    .CCF 480, 670
    .CCH 480, 670
    .CF 480, 670
    .CFG 670
    .CH 480, 670
    .DBA 670
    .DEF 670
    .DSF 670
    .ERC 670
    .ERR 670
extensions (continued)

.HEX 670
.IGS 670
.INF 200, 568, 670
.INS 202, 613, 670
.INX 200, 568, 670
.JED 670
.LIB 240, 670
.LNF 202, 670
.LOG 670
.LST 670
.MAC 670
.MAP 670
.NET 670
.PIP 202, 614, 670
.PLA 670
.PLD 670
.PLT 670
.PRN 670
.RES 202, 670
.SCH 670
.SRC 240, 670
.STM 670
.TDF 670
.TRC 670
.TVD 670
.TVO 670
.TVS 670
.TWG 430, 670
.VEC 670
.XRF 670

external statements in .INF and .LNF files 576

EXTRACT 435
command line controls 466
Programmable Logic Device Tools 434
To PLD 434, 437

Extract PLD
Key Fields 38

PLD Part Combine 48
PLD Type Combine 48

F

Field
Key Fields 37
Part Field
tips 45
Part Fields 37

Configur Schematic Design Tools 29

editing and moving 86
Part Value 37

Fields
Key Fields 230
Part Fields
configuring character height 34

FieldString1-FieldString8 symbol in netlist format files 618

file extensions  see  extensions

file stack, rebuilding in INET 213

file structure  see design structure

FileName symbol in netlist format files 618

file_index type definition in netlist format files 623

FIND command in Draft 93-94

FindSymbolChar function in netlist format files 628

FirstChild function in netlist format files 628

FirstChildPin function in netlist format files 628

FirstNet function in netlist format files 628

FirstNode function in netlist format files 628

FirstPart function in netlist format files 629

FirstPin function in netlist format files 629

FirstPipe function in netlist format files 629

flat design, defined 672

flat file structure see design structure

flat netlists  see also netlists

creating 207-219

creating from hierarchical designs 209

formats 602

formatting 204

FLDATTRB see also Select Field View
command line controls 466
FLDSTUFF  see also  Update Field Contents, 435
  command line controls 467
format file  see netlists
format_constant type definition in netlist
  format files 623
functions in netlist format files
  AccessChild 625
  AccessKeyWord 626
  AccessPart 626
  AddSignalName 626
  AddSymbol 626
  C-language 616
  ChildPinCount 626
  ClearSymbolicStrings 626
  CompareSymbol 627
  ConcatFile 627
  CopySymbol 627
  CreatePartDatabase 627
  EndNode 627
  ExceptionsFor 627
  FindSymbolChar 628
 FirstChild 628
  FirstChildPin 628
  FirstNet 628
  FirstNode 628
  FirstPart 629
  FirstPin 629
  FirstPipe 629
  general 615
  getche 629
  GetIndex 629
  getnum 629
  GetStandardSymbol 630
  GetSymbolChar 630
  HandleNodeName 630
  Initialize 630
  IsKeyWord 630
  LoadFieldString 631
  LoadFirstPin 631
  LoadInstance 631
  LoadPin 631
  MakeInstanceFile 631
  functions in netlist format files (continued)
  MakeLocalSignal 632
  NextAccessType 632
  NextChild 632
  NextChildPin 632
  NextInstance 632
  NextKeyWord 633
  NextNet 633
  NextNode 633
  NextPart 633
  NextPin 633
  NextPipe 633
  OrCAD-defined 615
  PackString 634
  PadSpaces 634
  PinCount 634
  PostFile 634
  PostProcess 634
  PreFile 635
  PreviousNode 635
  print 635
  ProcessFieldStrings 635
  putch 635
  puts 636
  PutSymbolChar 636
  RecordNode 636
  RewindInstanceFile 636
  SetAccessType 637
  SetCharSet 637
  SetFirst 637
  SetIndexByKeyRef 637
  SetNext 638
  SetNumberWidth 638
  SetPartIndex 638
  SetPinMap 638
  SetPrevious 639
  SetSignal 639
  SetSymbol 639
  SetToIndex 639
  SetTraversal 639
  SortByNumber 640
  SwitchIsSet 640
  SymbolInCharSet 640
functions in netlist format files (continued)
  SymbolLength 640
  ToUpper 640
  WriteCrlf 640
  WriteHeader 641
  WriteInteger 641
  WriteMap 641
  WriteNet 641
  WriteNetEnding 641
  WriteNetListEnd 641
  WriteStdSymbol 642
  WriteString 642
  WriteSymbol 642
FutureNet netlist format 511

G
GENDRIVE 179
GET command in Draft 16, 73, 95-98
  Convert 97
  Down 98
  Mirror 98
  Over 98
  Place 97
  Rotate 97
  Up 98
GET PART command in Edit Library 262, 296
getche function in netlist format files 629
GetIndex function in netlist format files 629
getnum function in netlist format files 629
GetStandardSymbol function in netlist format files 630
GetSymbolChar function in netlist format files 630
GRIDARRAY
  part definition 332
  parts 337

H
HandleNodeName function in netlist format files 630
HARDCOPY command in Draft 99-101
  Destination 100
  File Mode 101
  Make Hardcopy 101
  Width of Paper 101
HARDCOPY.PRN 667
HDUMP netlist format 564
header in .INF and .LNF files 573
helpful files, viewing 179
HEX files 670
HFORM see also Create Hierarchical
Netlist, 204, 601
command line controls 468
configuring Create Hierarchical Netlist
223
Create Hierarchical Netlist 221
data structures in netlist format files
611
incremental connectivity database 204
output from INET 200
required functions in netlist format files
610
traversal functions in netlist format
files 614
HI plotters 653
hierarchical
designs
changing parts into sheets 87
converting to flat 209
defined 672
guidelines for creating 159
inter-sheet connections 164
leaving subsheets 143
memory allocated to hierarchy
buffer 25
moving between worksheets 143
numbering reference designators
187
pipe link commands 201
plotting sheets with multiple
references 417
power considerations 171-174
printing sheets with multiple
references 417
sheet symbols 80, 131
file structure see design structure
netlists see also netlists
creating 221
formatting 204, 554-565
interpreting .INF files 572
viewing buffer status 73
hierarchical netlist formats 602
Hierarchy Options 25
buffer size 25
HiLo netlist format 516
hotpoint
buses 161
defined 126
HP plotters 652
HP.DRV 657
HPLASER1.DRV 658
HPLASER2.DRV 658
HPLASER3.DRV 658
HPLASER4.DRV 658
I
IBUILD
To Digital Simulation 449
IEEE parts see also parts
body outline 355
defining 352-356
introduction 245
pin placement 355
size limits 354
IFORM see also Create Netlist, 204, 601
configuring Create Netlist 209, 217-219
data structures in netlist format files
611
intermediate netlist structure 204
output from ILINK 202, 204
overriding incremental processing 218
required functions in netlist format files
610
IGES format, converting from plot file 403-405
IGS files 664, 670
ILINK see also Create Netlist, To Layout, 202
  checking for single-node nets 216
  command line controls 470
  configuring Create Netlist 209, 216
  INS files 202, 203
  netlists for PC Board Layout Tools 202
  output from !NET 200
  overriding incremental processing 216
  PIP files 202, 203
  RES files 202, 203
IMPORT command in Edit Library 297
include file format for Create Bill of Materials 426
incremental annotation
  configuration option 188
  defined 188
incremental connectivity database see also connection database, netlists
  contents 568
  defined 198, 672
  files comprising 200
  INF format 567
incremental design 197-199
incremental netlist process, defined 197
incremental netlisting, defined 672
INET see also Create Netlist
  assigning net names to unconnected pins 212
  checking electrical rules 212
  command line controls 472
  comparing time stamp 200
  configuring Create Hierarchical Netlist 223
  configuring Create Netlist 209, 211
  Create Hierarchical Netlist 221
  creates .INF and .INX files 568
  creating only reports 214
  incremental nature 205
  INF file defined 200
  INF files 200
  introduction 200-201
  INX files 200, 201
  netlists for simulation 200
  overriding incremental netlisting 213
  reporting off-grid parts 212
  speeding processing 213
  to HFORM 200
  to ILINK 200
INF files 568, 664, 670
  defined 200
  differences from .LNF files 600
  extension 200
  sample 598
INF format specification 572
initial macro
  configuring 24
  defined 672
Initialize function in netlist format files 630
input pins see also pins
  defined 311
INQUIRE command in Draft 102
  viewing error messages 388
INS files 202, 203, 613, 664, 670
  data functions in netlist format files 613
  extension 202
instance file see INS files
instance statements in .INF and .LNF files 576
integer_constant type definition in netlist format files 624
IntelADF netlist format 518
   header information 518
   including equations in the netlist 519
Intergraph netlist format 521
intermediate netlist structure see also connectivity database, netlists defined 202, 673
INX files 568, 664, 670
defined 201
extension 200
using 205
IsKeyWord function in netlist format files 630

J
JED files 670
joined statements in .INF and .LNF files 581
JUMP command
   Draft 103-105
      Reference 103
      X-Location 104
      Y-Location 105
   Edit Library 298
      X-Location 298
      Y-Location 299
junctions
   checking for overlapping or duplicate objects 193
   placing in Draft 123

K
K, defined 673
Key Fields 37-50, see also Configure Schematic Design Tools
   Annotate Part Value Combine 40
   Annotate Schematic 38, 184
   Create Bill of Materials 38, 423
      Include File Combine 47
      Part Value Combine 47
   Create Netlist 38
   Extract PLD 38
      PLD Part Combine 48
      PLD Type Combine 48
   key field, defined 673
Netlist
   Module Value Combine 46
   Part Value Combine 46
   netlist formats 481
   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 618
keywords in Symbol Description Language 358, 360

L
labels
   checking for overlapping or duplicate objects 193
   configuring character height 34
   connecting signals without wires 159
   editing in Draft 77
   placing in Draft 125
   reporting connected 212
LastFile symbol in netlist format files 631
libraries (continued)
  inserting a library 14
  introduction to 239
Library Options
  Active library size 18
  Available Libraries 14
  Configured Libraries 12, 14-15, 16
  inserting a library 14
  Library Prefix 13
  Name Table Location 15-17
  removing a library 14
  Symbolic Data Location 15-17
library source file defined 240
  listing parts 241, 249, 251, 301
  listing parts in Draft 106
  loading order 12
  objects 292
  on-line 73
  order searched 241
  part names, duplicate 13
  parts see parts
  parts found in each library 179
  removing a library 14
  saving 300, 313
  saving memory 253
  sheetpath designator 247
  size limits 264
  status of active library in Edit Library 73
  status of current library 292
  system memory 240
  viewing parts in Draft 107
  writing to files 314
LIBRARY command
  Draft 106-107
    Browse 16, 107
    Directory 106
  Edit Library 300
    Browse 302
    Delete Part 303
    List Directory 301
    Prefix 304
    Update Current 262, 300
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 240
  creating 327
  creating and modifying 240
  Decompile Library 323
  defined 240
  editing 242
LibraryNameString symbol in netlist format files 618
lines see dashed lines, wires
link statements in .INF and .LNF files 574
linked connectivity database see also
  connectivity database, netlists
  contents 569
  defined 198, 673
  INF format 567
list box 5
List Library 249-252
  execution 249
    Local Configuration 250
      File Options 251
        Processing Options 252
      sample report 250
      string files 252
      use 241
LNF files 202, 203, 664, 670
  differences from .INF files 600
  sub-parts 600
LoadFieldString function in netlist format files 631
LoadFirstPin function in netlist format files 631
loading order, libraries 12
LoadInstance function in netlist format files 631
LoadPin function in netlist format files 631
Local Configuration
  Compile Simulation Specification 451
LocalSignal symbol in netlist format files 618
LOG files 670
LookupNameString symbol in netlist format files 619
LST files 670
M
M2EDIT 177, 179
  Edit File 177
    View Reference 179
MAC files 665, 670
MACRO command
  Draft 108-119
    Capture 109-112
    Delete 112
    Initialize 112
    List 112
    Read 112
    Write 113
  Edit Library 306
    Initial Macro 306
macros
  <Ctrl><Break> 111
  about 108
  comments 116
  configuring for macros 113
  defined 108, 673
  deleting 112
Draft
  buffer 72, 73, 113
  calling 113
  creating 109-119
  debugging 111
  initial 112
  nesting 111
  pause 111
  syntax 114
  text files 114
  using three-button mouse 117
  valid macro keys 109
Schematic Design Tools Reference Guide

macros (continued)
   Edit Library 306
      buffer 292
      initial 306
   efficient 119
   enabling prompts 151
   initial 24
   interrupting macros 111
   listing 112
   loading 112
   loading automatically 24
   macro buffer 23
   Macro Options 23-24
   using 113
main menu commands  xxxii
MakeInstanceFile function in netlist format files 631
MakeLocalSignal function in netlist format files 632
MAP files 665, 670
match string 230, see also Update Field Contents
mB, defined 391, 673
megabyte, defined 673
memory see system memory
Mentor netlist format 522
mI, defined 391
mO, defined 391
module port statements 575
   in .INF and .LNF files 582
module ports
   as connection between worksheets 175-
      176
   checking connections 213, 395
   checking for overlapping objects 193
   configuring character height 34
   connecting to buses 165
   connecting worksheets 164
   defined 673
   editing in Draft 78
   placing in Draft 127
   reporting unconnected 395
   unconnected, reporting 212
ModuleName symbol in netlist format files 619
mouse
   setting left button in Draft 150
   three-button and macros 117
moving see mouse, keyboard commands, JUMP command
mU, defined 391
MultiWire netlist format 524

N
NAME command in Edit Library 307-308
   Add 308
   Delete 308
   Edit 308
   Prefix 308
name table
   configuring location 15-17
   Draft 73
NC, defined 391
NET files 665, 670
net, defined 673
net-oriented data structure in netlist format files 612
NetCode symbol in netlist format files 619
netlist format files see netlists
netlists
   compiler
      INET 200
   compiling
      compiling several times 205
   creating 197-205
   defined 674
   flat
      creating 207-219
      multiple sheets 200
      one-sheet 200
      sheetpath parts 212
      uses for 207
   flat formats 481-552
   for PC boards 207, 210
   for simulation 207, 210
netlists (continued)

format files

creating custom 601, 644
required functions 610

formats 479-565

Algorex 487
Allegro 489
AlteraADF 490
AppliconBRAVO 494
AppliconLEAP 495
Cadnetix 496
Calay 498, 500
Case 501
CBDS 503
ComputerVision 504
creating custom 601-644
customer-contributed formats 602
DUMP 505
EDIF 506
EDIF hierarchical 558
EEDesigner 510
file extensions 480
flat 481-552, 602
FutureNet 511
HDUMP 564
hierarchical 554-565, 602
HiLo 516
IntelADF 518
Intergraph 521
introduction 479
Key Fields 481
Mentor 522
MultiWire 524
OrCAD Digital Simulation Tools
  Model 529
OrCAD Programmable Logic
  Design Tools 527
OrCAD-supplied 602
OrCAD/PCB II 525
PADS ASCII 531, 533
part and net orientations 602
PCAD 534
PCADnlt 536

netlists (continued)

PDUMP 538
RacalRedac 539
Scicards 541
selecting 218
SPICE 543
SPICE hierarchical 564
Tango 547
Telesis 549
types of format files 479
Vectron 550
WireList 552

formatting 204

hierarchical

complex 200
creating 221
formats 554-565
formatting 204
simple 200

incremental design 197-199

incremental netlist process

defined 197
HFORM 198
IFORM 198
ILINK 198
INET 198
time stamp 198

intermediate netlist structure 203

defined 202

linking 202

overriding incremental processing 213

selecting formats 218

speeding processing 213

SPICE format 203

NetNameString symbol in netlist format files 619

NetNumber symbol in netlist format files 619

NetType symbol in netlist format files 620

NextAccessType function in netlist format files 632

NextChild function in netlist format files 632
NextChildPin function in netlist format files 632
NextInstance function in netlist format files 632
NextKeyword function in netlist format files 633
NextNet function in netlist format files 633
NextNode function in netlist format files 633
NextPart function in netlist format files 633
NextPin function in netlist format files 633
NextPipe function in netlist format files 633
no-connect objects
 defined 139
 placing in Draft 139
notation conventions xxxii
notes xxxii
number in .INF and .LNF files defined 571
numeric constants in Symbol Description Language 358, 360

O
objects
 in libraries 292
 overlapping or duplicate, checking for 193
 placing in Draft 120
off-grid parts
 preventing 154
 reporting 196, 212, 395
on-line library in Draft 73
open collector see also pins.ii.pins:open collector
defined 312
open emitter see also pins
defined 312
OrCAD Digital Simulation Tools Model
 netlist format 529
OrCAD Programmable Logic Design Tools
 netlist format 527
OrCAD-supplied netlist formats 602
OrCAD/PCB II netlist format 525

ORCADESP.DAT file 667
 order of libraries 12
Organization symbol in netlist format files 620
orienting parts see parts
ORIGIN command in Edit Library 309
origin on plotted schematics 35
orthogonal wires and buses 65
drawing 151
orthogonal, defined 65
output pins see also pins
 defined 311

P
package, defined 674
PackString function in netlist format files 634
PADS ASCII netlist format 531, 533
PadSpaces function in netlist format files 634
pan, defined 674
parallel port 11
parameter in .INF and .LNF files defined 571
parent, defined 674
parity 11
part data structure in netlist format files 611
part definition 331-333
 components of 331
 construction 331
 sheetpath parts 331
types of 331
Part Fields 37
 abbreviations in key fields 39
changing visibility globally 226
Configure Schematic Design Tools 29
configuring character height 34
defined 674
editing and moving 86
tips 45
part instance statements in .INF and .LNF files 576
part name string 352
part reference see reference designator
Part Reference, character height 33
Part Value field 37
part values
abbreviation in key fields 39
changing visibility globally 226
configuring character height 34
ingesting and moving 85
part-oriented data structure in netlist
format files 613
PartIndex symbol in netlist format files 620
PARTLIST see also Create Bill of Materials
command line controls 475
PartName symbol in netlist format files 620
parts
adding pins 310
bitmap 260
block 272
body
block parts 244
graphic parts 245
IEEE parts 245
body outline display 319
body types 244, 260
changing kind of part 276
checking for duplicate reference
designators 400
checking for overlapping or duplicate
objects 193
checking for type mismatches 401
checking for unused parts 400
components 244
constraints 272
convert defined 274
convert, defining 274
converted forms 97
creating 262
creating a bill of materials 421-428
cross reference listing 397
defined 674
defining pins 311
deleting pins 310
parts (continued)
displaying convert form 88
drawing block parts 276
drawing graphic parts 277-284
drawing IEEE parts 285-290
duplicate names 13
ingesting 83, 262
editing prefix definitions 304
graphic 272
size limits 327
guidelines for creating 275
IEEE 261, 272, 273, 274, 337
defining 352-356
IEEE, defining 274
limits 263
limits on complexity 263
locating 106, 107
moving pins 312
names 247
naming 307
naming pins 310
numbering pins 310
object for unconnected pins 139
part fields, changing visibility globally 226
part name string 334
part values, changing visibility globally 226
parts per package, swapping in Draft 89
pin names 247
pin number
alphanumeric 246
pin numbers 246
pin shapes 246
pin types 246
pin-grid array 342
pins see pins
placing on grid 115
power objects 167, 169
preventing off-grid 154
reference designators 247, 316
reference designators, changing
visibility globally 226
703
parts (continued)

- reporting connected 212
- reporting coordinates 400
- reporting off-grid 212
- reporting off-grid parts 196, 395
- rotating and mirroring in Draft 88
- screen representation 275
- sheetpath 247
- sheetpath parts 335
- showing pin numbers in Draft 152
- sizing bodies 275, 276
- specifying pin shapes 312
- status of the current part 293
- suffixes 296
- types 272
- updating reference designators 183
- vector 260

- parts per package 334
  - checking for unused parts 400
  - choosing 274
  - specifying zero 340
  - updating reference designators 184

- part_symbol type definition in netlist format files 624

- passive pins see also pins
  - defined 311

- PC Board Layout Tools 140, 159
  - creating linked connectivity databases 199
  - creating netlists for 210
  - linked connectivity database 203
  - LNF files 202
  - output from ILINK 202
  - producing linked connectivity database 216

- PCAD netlist format 534
- PCADnlt netlist format 536
- PCB II netlist format 525
- PCB, defined 674
- PDUMP netlist format 538
- Pen speed, configuring 28
- Pen width, configuring 28
- pin-grid array parts 274, see also parts

- PIN command in Edit Library 310
  - Add 310
  - Delete 310
  - Move 312
  - Name 310
  - Pin-Number 310
  - Shape 312
  - Type 311

- pin definition 332, see also pins-333, 338
- Pin Name, character height 33
- Pin Number, character height 33
- pin statements in .INF and .LNF files 582
- pin stimulus statements in .INF and .LNF files 593
- pin trace statements in .inf and .Inf files 585, 587
- pin vector statements in .INF and .LNF files 590
- Pin-to-Pin distance 33
- PinCount function in netlist format files 634
- PINIndex symbol in netlist format files 620
- PINNameString symbol in netlist format files 621
- PINNumberString symbol in netlist format files 621

- pins see also pin definition
  - bidirectional 311
  - displaying pins 340
  - input 311
  - names 247
  - numbers 246
  - alphanumeric 246
  - object for unconnected 139
  - open emitter 312
  - output 311
  - passive 311
  - pin definitions 338
  - pin-grid array 342
  - power 167, 311
  - power objects 169
  - power pin visibility 319
  - reporting unconnected 395
  - shapes 246
pins (continued)
    showing pin numbers in Draft 152
three-state 311
types 246
    unconnected, reporting 212
PinType symbol in netlist format files 621
PIP files 202, 203, 614, 665, 670
    extension 202
pipe commands
    in INF files 200
    using in flat designs 201
pipe file see PIP files
pipe file functions 614
pipe link commands see pipe commands, PIP files
pipe statements in .INF and .LNF files 596
PipeLine symbol in netlist format files 621
PLA files 670
PLACE command in Draft 120-142
    Bus 122
    Dashed Line 134
    Entry (bus) 124
    Junction 123
    Label 125
    Layout 140
    Module Port 127-128
    No-Connect 139
    Power 129-130
    Sheet 131-132
    Stimulus 137-138
    Text 133
    Trace 135
    Vector 136
    Wire 120
PLD files 670
Plot Schematic 407-416
    Execution 407
    Local Configuration 410
        File Options 410
            Processing Options 413
    Plot X and Y Offset 35
    plotting to a file 412
    plotting to a printer 411
    sample output 409
    scaling output 413-415
    setting offsets 415
    suppressing title block, border, and text 408
PLOTALL see also Plot Schematic
    command line controls 476
plotter drivers 10
plotter pens, configuring 26
plotter, defined 407
plotters see printing and plotting
PLT files 665, 670
pointer, defined 674
PostFile function in netlist format files 634
PostProcess function in netlist format files 634
PostScript plotter drivers 659
power
    connecting 167
    isolating 170-174
    objects 167
        connecting 167, 169
        defined 167
        editing in Draft 79
        global objects 167
        names 169
    placing in Draft 129
    supplies, creating different 168-169
power pins see also pins
    configuring visibility 319
    defined 311
Power Text, character height 34
PreFile function in netlist format files 635
prefix definition see also parts
convention 329
Symbol Description Language 361
PreviousNode function in netlist format
files 635
print function in netlist format files 635
Print Schematic 417-420
execution 417
Local Configuration 419
File Options 419
Processing Options 420
printing to a file 420
sample output 418
suppressing pin numbers 420
PRINTALL see also Print Schematic
command line controls 477
Printed Circuit Board Layout Tools xxv
printer, defined 407
printing and plotting 11, see also
HARDCOPY command
common sheet dimensions 32
configuring plotter pens 26
creating an IGES file 403-405
Plot Schematic 407-416
Plotter drivers 10
plotter 11
moving origin 35
plotting 645-660
cabling 646
Calcomp plotters 654
HI plotters 653
hints 651
HP driver 657
HP LaserJet driver 658
HP plotters 652
MODE command 649
PC/AT cabling 647
PC/XT cabling 646
PC/XT cabling for IOline plotters
647
plotting
DXF driver for AutoCAD 658
problems 648
printing and plotting (continued)
plotting to a file 412
plotting to a printer 411, 650
Print Schematic 417-420
printer drivers 9
Printer/Plotter Output Options 11
printers 11
printing grid references and pin numbers 420
plotting to a file 420
scaling output 413
setting offsets 415
suppressing title block, border, and text
408
PRN files 665, 670
ProcessFieldStrings function in netlist format files 635
processors 181-236
defined 674
introduction xxvii
Programmable Logic Design Tools xxv, 433
netlist format 527
Programmable Logic Device Tools
To PLD
EXTRACT 434
work screen 435
programmable logic device, defined 674
prompt, defined 674
PSCRIPT.DRV 659
putch function in netlist format files 635
puts function in netlist format files 636
PutSymbolChar function in netlist format files 636
PWOR.DRV 660
Q
quiet mode, defined 675
QUIT command
Draft 143-146
  Abandon Edits 145
  Enter Sheet 143
  Initialize 144
  Leave Sheet 143
  Run User Commands 146
  Suspend to System 145
  Update File 144
  Write to File 144
Edit Library 313-315
  Abandon Edits 315
  Initialize 314
  Suspend to System 314
  Update File 262, 313
  Write to File 262, 314
quotation marks xxxii
quoted token in .INF and .LNF files defined 570

R
RacalRedac netlist format 539
radio button, defined 675
raster commands in printing 407
READ.ME files, viewing 179
RecordNode function in netlist format files 636
REFERENCE command in Edit Library 316
reference designators 247, 334, 336
  changing visibility globally 226
  checking for duplicates 400
  editing and moving 83
  location of 336
  modifying 189
  updating automatically 183
Reference field 37, see also Key Fields
  abbreviation for reference in key fields 39
reference files 667
  viewing 179
ReferenceString symbol in netlist format files 621
REPEAT command in Draft 147
reporters 385-431
  defined 675
  introduction xxvii
RES files 202, 203, 665, 670
  extension 202
resolved file see RES files
Revision symbol in netlist format files 621
RewindInstanceFile function in netlist format files 636
root directory, defined 675
root sheet, defined 675
Rotate command in Draft 88
Run User Commands command in Draft 146

S
sB, defined 391
SCH files 665, 670
Schematic Design Tools xxv
schematic structure see design structure
schematics see also Draft, Draft commands, worksheets
defined 675
Scicards netlist format 541
screen colors in Draft, configuring 26
scroll buttons 5
  defined 675
SDL see Symbol Description Language
SDT.BCF file 667
SDT.CFG file 667
Select Field View 37, 225-227
  execution 225
  Local Configuration 225
    File Options 226
    Processing Options 226
    Part Fields 226
serial port 11
SET command

Draft 148-156
Auto Pan 149
Backup File 149
Drag Buses 150
Error Bell 150
Grid Parameters 154
Left Button 150
Macro Prompts 151
Orthogonal 151
Repeat Parameters 155
Show Pin Numbers 152
Title Block 152
Visible Lettering 156
Worksheet Size 153
X,Y Display 153

Edit Library 317
Auto Pan 317
Backup File 317
Error Bell 318
Left Button 318
Macro Prompts 318
Power Pins Visible 319
Show Body 275
Show Body Outline 287,319
Visible Grid Dots 319

set parameter stimulus statements in .INF and .LNF files 592
SetAccessType function in netlist format files 637
SetCharSet function in netlist format files 637
SetFirst function in netlist format files 637
SetIndexByRef function in netlist format files 637
SetNext function in netlist format files 638
SetNumberWidth function in netlist format files 638
SetPartIndex function in netlist format files 638
SetPinMap function in netlist format files 638
SetPrevious function in netlist format files 639
SetSignal function in netlist format files 639
SetSymbol function in netlist format files 639
SetToIndex function in netlist format files 639
SetTraversal function in netlist format files 639

sheet net statements in .INF and .LNF files 585
sheet net trace statements in .inf and .Inf files 585, 588

sheet nets
adding to sheet symbol 131
configuring character height 34
defined 675
deleting 131
editing 132

sheet part names
editing and moving 87

sheet parts
defined 675

sheet symbols
adding sheet nets 131
changing filename 132
changing size 132
deleting sheet nets 131
Draft
adding nets 80
changing size 82
deleting nets 80
naming nets 81
editing filenames 81
editing in Draft 80
editing in Draft nets 81
editing sheet nets 132
naming 132
placing in Draft 131
viewing represented worksheet 143

SheetNumber symbol in netlist format files 622

sheetpath designator 247
sheetpath parts
  Archive Parts in Schematic 257
  creating netlists 212
  defined 675
  part definitions 331
  referencing 335
sheets see also worksheet
  configuring border width 35
  configuring character height of name 34
  configuring size 22
  defined 675
  size
    American 30
    built-in 31
    International Standards
      Organization 30
  units of measure 31
SheetSize symbol in netlist format files 622
shelling to system see system
short-cuts
  aligning wires and buses 65
  changing visibility of part information globally 226
  checking for unconnected nodes 216
  cutting wires 115
  debugging flat netlists 214
  efficient macros 119
  placing multiple objects 147
  placing parts on grid 115
  repeating commands in Draft 62
  repeating commands in Edit Library 271
  using macros 108-119
Show Design Structure 429-431
  execution 429
  Local Configuration 430
  sample output 429
si, defined 391
signal statements in .INF and .LNF files 576, 581, 584, 587
signal stimulus statements in .INF and .LNF files 593
signal vector statements in .INF and .LNF files 590
SignalNameString symbol in netlist format files 622
signals
  connecting buses to module ports 165
  connecting with buses 162
  connecting with labels 125, 159
  connecting with module ports 127
  connecting with sheet symbols 131
  global objects 167
  splitting multiple signals off buses 166
SignalType symbol in netlist format files 622
SIMPLE see also Complex to Simple,
  Design Management Tools
  command line controls 478
simple hierarchy, defined 675
simulation
  creating connectivity databases 199
  single-node nets, checking for 216
so, defined 391
SortByNumber function in netlist format files 640
spacebar xxxi
Spacing Ratio 35
SPICE hierarchical netlist format 564
SPICE netlist format 203, 543
  node names 544
SRC files 240, 666, 670
standard_symbol type definition in netlist format files 625
statement in .INF and .LNF files defined 571
stimulus objects
  editing in Draft 92
  INQUIRE command in Draft 102
  placing in Draft 137
stimulus statements in .INF and .LNF files 592
STM files 666, 670
stop bits 11
Schematic Design Tools Reference Guide

string files
created by List Library 252
creating with Archive Parts in
Schematic 256, 257
defined 257
source for Archive Parts in Schematic
256
string in .INF and .LNF files defined 570
string_constant type definition in netlist
format files 625
sU, defined 391
sub-part code in .INF and .LNF files
defined 571
sub-parts
defined 676
in .LNF files 600
subsheets
exiting 143
viewing 143
switches see command line controls
switches for netlist format files 616
SwitchIsSet function in netlist format files
640
Symbol Description Language
character strings 358, 360
comments in 364
creating a library source file with
Archive Parts in Schematic 256
defining bitmaps in 373-375
maximum size 374
defining converts in 378-379
defining parts in 363-366
defining pins in 367-372
defining vectors in 376-377
keywords 358, 360
numeric constants 358, 360
prefix definition 361
specification 357-379
symbol size 332, see also parts
symbol table in Draft 73
symbol type definition in netlist format files
625
Symbolic Data Location 15-17
SymbolInCharSet function in netlist format
files 640
SymbolLength function in netlist format
files 640
symbols in netlist format files
AddressLine1-AddressLine4 617
DateString 617
DocumentNumber 617
ExitType 617
FieldString1-FieldString8 618
FileName 618
KeyWordin 618
LastFile 631
LibraryNameString 618
LocalSignal 618
LookupNameString 619
ModuleName 619
NetCode 619
NetNameString 619
NetNumber 619
NetType 620
Organization 620
PartIndex 620
PartName 620
PinIndex 620
PinNameString 621
PinNumberString 621
PinType 621
PipeLine 621
ReferenceString 621
Revision 621
SheetNumber 622
SheetSize 622
SignalNameString 622
SignalType 622
TimeStamp 622
TitleString 622
TotalSheets 622
TypeSheets 622
typeCode 623
typology diagrams
how to read 357-359
type, defined 676
system
clock, importance when creating netlists 201
EMS 16
advantages and disadvantages in Draft 73
configuring 17
Draft 72
memory
available in Edit Library 291, 292
hierarchy buffer 25
saving 253
status in Draft 72
running system commands from Draft 146
suspending to from Draft 145
suspending to from Edit Library 314
time stamp
defined 622
importance in incremental netlist process 198
TimeStamp symbol in netlist format files 622
title block statements in .INF and .LNF files 573
title blocks
ANSI 19
configuring character height 34
creating custom 152
displaying standard on worksheets 152
editing ANSI 91
editing in Draft 90
OrCAD 19
suppressing during printing and plotting 408
suppressing lines 27
suppressing lines and text 28
suppressing text 27
types 19
using pre-printed paper 28
TitleString symbol in netlist format files 622
To Digital Simulation 449
IBUILD 449
Local Configuration 447
To Layout 453
Execute 454
Local Configuration 454
To Main 457
To PLD 433-444
Execution 434
EXTRACT 434, 437
Local Configuration 435
token in .INF and .LNF files defined 570
tool set xxv
defined 676
tool, defined 676
TotalSheets symbol in netlist format files 622
ToUpper function in netlist format files 640

T
TAG command
Draft 157
Edit Library 320
tag, defined 676
Tango netlist format 547
TDF files 670
Telesis netlist format 549
Template Table 30-36
text
configuring spacing ratio for Draft 35
displaying in Draft 156
editing before placing in Draft 133
placing in Draft 133
rotating in Draft 133
sizing in Draft 133
View Reference 179
text editor
creating custom libraries 242
Edit File 177
text export, defined 676
text import, defined 676
three-state pins see also pins
defined 311
trace objects
  editing in Draft 92, 135
  INQUIRE command in Draft 102
  placing in Draft 135
trace statements in .inf and .Inf files 587
transfers 432-457
  defined 676
  introduction xxvii
  To Digital Simulation 449
  To Layout 453
  To Main 457
traversal functions in netlist format files 614
traversal_constant type definition in netlist format files 625
TRC files 666, 670
TREELIST see also Show Design Structure command line controls 478
troubleshooting schematics 505, 538, 563
TTL, defined 676
tutorial files 668
TVD files 670
TVO files 670
TVS files 670
TWG files 666, 670
type definitions in netlist format files
  access_constant 623
  file_index 623
  format_constant 623
  integer_constant 624
  part_symbol 624
  standard_symbol 625
  string_constant 625
  symbol 625
  traversal_constant 625
  user_constant 625
  variable 625
type, defined xxxi
TypeCode symbol in netlist format files 623

U
unconditional annotation
  configuration option 188
  defined 188
units of measure on sheets 31
Update Field Contents 229-236
  creating update reports 235
  execution 233
Key Fields 38
  Combine for Fields 1 through 8 43
  Combine for Value 43
Local Configuration 234, 236
  File Options 235
  Processing Options 235
Part Fields 235
  update file 230, 235
  update file strings, length 231
  update files 668
update file see also Update Field Contents
  configuring 235
  creating 230
  errors 232
update string 231, see also Update Field Contents
upload, defined 676
user buttons xxviii
  defined 676
  introduction xxviii
User commands in Draft 146
user_constant type definition in netlist format files 625
Index

V
variable type definition in netlist format files 625
VCC see power objects
VDD see power objects
VEC files 670
vector commands in plotting 407
vector definition
    creating 346, 353
    defined 333, 346
    graphic symbols 345
    screen representation 275
vector objects
    editing in Draft 92
    INQUIRE command in Draft 102
    placing in Draft 136
vector statements in .INF and .LNF files 590
vector, defined 260, 676
Vectron netlist format 550
View Reference 179
    execution 179
    M2EDIT 179

W
Was/Is file 669, see also Back Annotate
    defined 189
    format 189
white space in .INF and .LNF files defined 570
wildcard, defined 676
WireList netlist format 552
wires
    checking for overlapping or duplicate objects 193
    connecting signals without wires 159
    drawing orthogonal 151
    labels 159
    macro for cutting 115
    placing in Draft 120
    reporting unconnected 212, 395
    short-cut for aligning 65
Worksheet Options 19-22
Worksheet Prefix 21
worksheets see also Draft
    back annotating reference designators 189
    checking for overlapping or duplicate objects 193
    cleaning up large worksheets 193
    configuring default extension 22
    configuring worksheet prefix 21
    connecting signals without wires 159
    connections between sheets 164
    default document number 22
    default organization
        name.ii.Organization name 22
    default revision number 22
    default size 22
    default title 22
    defined 677
    global power 167
    inter-sheet connections 164
    maximum dimensions 30
    setting size 153
    size 31-33
    ANSI 31
    ISO 31
    OrCAD default, inches 33
    OrCAD default, millimeters 33
    scale factors for Plot Schematic 414
    suppressing border during printing and plotting 408
    suppressing borders during printing and plotting 408
    tagged locations 157
    updating reference designators 184
WriteCrLf function in netlist format files 640
WriteHeader function in netlist format files 641
WriteInteger function in netlist format files 641
WriteMap function in netlist format files 641
WriteNet function in netlist format files 641

713
WriteNetEnding function in netlist format
files 641
WriteNetListEnd function in netlist format
files 641
WriteStdSymbol function in netlist format
files 642
WriteString function in netlist format files
642
WriteSymbol function in netlist format files
642

X
X and Y Border Width 35
XRF files 666, 670

Z
ZOOM command
  Draft 158
  Center 158
  In 158
  Out 158
  Select 158
  Edit Library 321
    Center 321
    In 321
    Out 321
    Select 321
  zoom, defined 677